Lecture 4: Monte Carlo Methods

Friday, November 19, 2021

Reinforcement Learning, Winter Term 2021/22

Joschka Boedecker, Gabriel Kalweit, Jasper Hoffmann

Neurorobotics Lab
University of Freiburg

Lecture Overview

Lecture Overview

Recap: Dynamic Programming

Last lecture: Planning by dynamic programming, solve a *known* MDP.

Policy Iteration

Alternate **evaluating** the value function v_π and **improving** the policy π to convergence.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

Value Iteration

Evaluate just once and combine it with the policy improvement step.

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r | s, a) \left[r + \gamma v_k(s')\right] \end{aligned}$$

Monte Carlo Reinforcement Learning

This lecture: Model-free prediction and control. Estimate/ Optimize the value function of an *unknown* MDP.

- MC methods learn from episodes of *experiences*
  experiences = sequences of states, actions, and rewards
- MC is model-free: no knowledge required about MDP dynamics
- MC learns from complete episodes (no bootstrapping), based on averaging sample returns

Lecture Overview

Monte Carlo Prediction

- Goal: learn the state-value function v_π for a given policy π

$$S_0, A_0, R_1, ..., S_T \sim \pi$$

- Idea: estimate it from experience by average the returns observed after visits to that state
- Recall: the *return* is the total discounted reward

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

- Recall: the value function is the expected return

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

- Monte-Carlo policy prediction uses the empirical mean return instead of expected return

Incremental and Running Mean

- We can compute the mean of a sequence $x_1, x_2, \ldots$ incrementally:

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

Incremental and Running Mean

- Thus, we can update $V(s)$ incrementally by:

$$V(s) \leftarrow V(s) + \frac{1}{N(s)}(G_t - V(s)),$$

  where $\frac{1}{N(s)}$ is the state-visitation counter
- Instead $\frac{1}{k}$, we can use step size α to calculate a running mean:

$$V(s) \leftarrow V(s) + \alpha(G_t - V(s))$$

Monte Carlo Prediction

- First-visit MC method: Estimates $v_\pi(s)$ as the average of the returns following first visits to s.
- Every-visit MC method: Estimates $v_\pi(s)$ as the average of the returns following all visits to s.

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow \text{average}(Returns(S_t))$

Example: Blackjack

- States (200 of them) :
  - Current sum (12 - 21)
  - Dealer's showing card (ace-10)
  - Do I have a *usable* ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 sum of cards $<$ sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Example: Blackjack

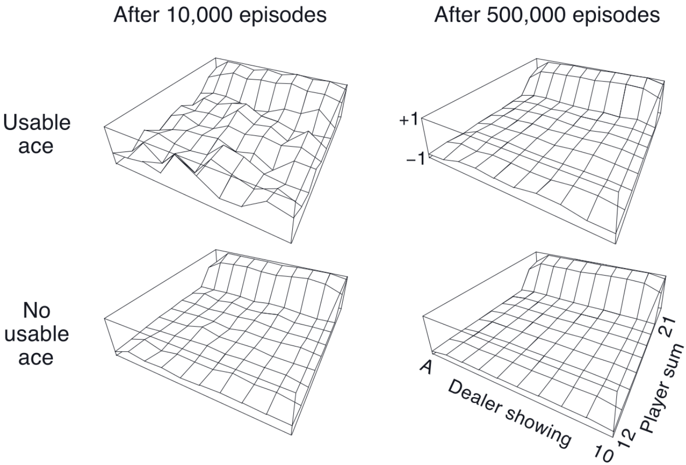
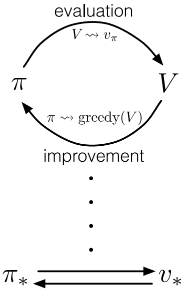
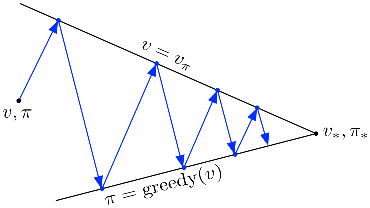


Figure 5.1: Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation. ■

Lecture Overview

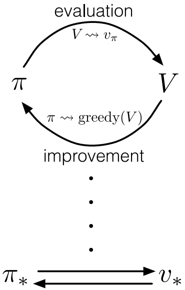
Generalized Policy Iteration

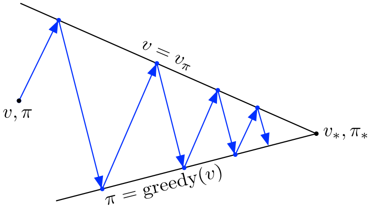




- Policy Evaluation: estimate v_π
- Policy Improvement: greedy

Generalized Policy Iteration with MC Evaluation





- Monte Carlo Policy Evaluation: $V \approx v_\pi$
- Policy Improvement: greedy?

Monte Carlo Estimation of Action Values

- Greedy policy improvement over $V(s)$ requires a model of the MDP

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

- Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$$

Generalized Policy Iteration with action-value function:

- Monte Carlo Policy Evaluation: $Q \approx q_\pi$
- Policy Improvement: greedy?

ϵ-greedy Policy Improvement

- We have to ensure that each state-action pair is visited a sufficient (infinite) number of times
- Simple idea: ϵ-greedy
- All actions have non-zero probability
- With probability ϵ choose a random action, with probability $1-\epsilon$ take the greedy action.

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}|} + 1 - \epsilon & \text{if } a = \arg\max_{a' \in \mathcal{A}} Q(s, a') \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

ϵ-greedy Policy Improvement

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s, a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_a q_\pi(s, a) + (1 - \epsilon) \max_a q_\pi(s, a) \\
&\geq \frac{\epsilon}{|\mathcal{A}|} \sum_a q_\pi(s, a) + (1 - \epsilon) \sum_a \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}|}}{1 - \epsilon} q_\pi(s, a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_a q_\pi(s, a) - \frac{\epsilon}{|\mathcal{A}|} \sum_a q_\pi(s, a) + \sum_a \pi(a|s) q_\pi(s, a) \\
&= v_\pi(s)
\end{aligned}
$$

On-policy First-visit MC Control

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

Off-policy Learning

- We want to learn the optimal policy, but we have to account for the problem of *maintaining exploration*
- We call the (optimal) policy to be learned the *target policy* π and the policy used to generate behaviour the *behaviour policy* b
- We say that learning is from data *off* the target policy – thus, those methods are referred to as *off-policy learning*

Importance Sampling

- Weight returns according to the relative probability of target and behaviour policy
- Define state-transition probabilities $p(s'|s,a)$ as $p(s'|s,a) = \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r\in\mathcal{R}} p(s',r|s,a)$
- The probability of the subsequent trajectory under any policy π, starting in S_t, then is:

$$
\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots S_T | S_t, A_{t:T-1} \sim \pi\} \\
&= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&= \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)
\end{aligned}
$$

Importance Sampling

The relative probability therefore is:

Definition: Importance Sampling Ratio

$$\rho_{t:T-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)}$$

The expectation of the returns by b is $\mathbb{E}[G_t|S_t = s] = v_b(s)$. However, we want to estimate the expectation under π. Given the importance sampling ratio, we can transform the returns by b to yield the expectation under π:

$$\mathbb{E}[\rho_{t:T-1} G_t|S_t = s] = v_\pi(s).$$

Importance Sampling can come with a vast increase in variance.

Off-policy MC Control

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

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

$Q(s,a) \in \mathbb{R}$ (arbitrarily)

$C(s,a) \leftarrow 0$

$\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):

$b \leftarrow$ any soft policy

Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Lecture Overview

Summary by Learning Goals

Having heard this lecture, you can now...

- describe how to evaluate a given policy with Monte Carlo rollouts
- explain policy improvement by on-policy Monte Carlo control