Mini-Course 2, Module 1
Monte Carlo Methods for Prediction & Control

CMPUT 365
Fall 2021

October 4, 2021

- Friday (8th) at noon: Graded quiz DUE
  - Don’t forget Lab on Wednesday! Don’t use up all your attempts before getting help!
- In Lab Andy will also review solution to last week’s DP notebook
- Any questions about course admin?

Planning vs learning from interaction

- Recall: the universe consists of you, a chess board and pieces AND me (but I only play chess and I never talk, never respond to you)
  - The only thing you can do in life is play chess against me!!
- There are two ways you could figure out how to beat me:
  - Play me over and over and figure out how the game works; figure out my play. **Trial and error learning** (like in a bandit)
  - **Dynamic Programming:** If you had a **book** describing the rules & how I play (Adam is part of the environment)

Using Monte Carlo to beat me in chess

- You start with some policy: say move the piece forward closest to my nearest piece

1. Play a full game till the end against me:

    1.1. Policy/strategy is **frozen** during the game

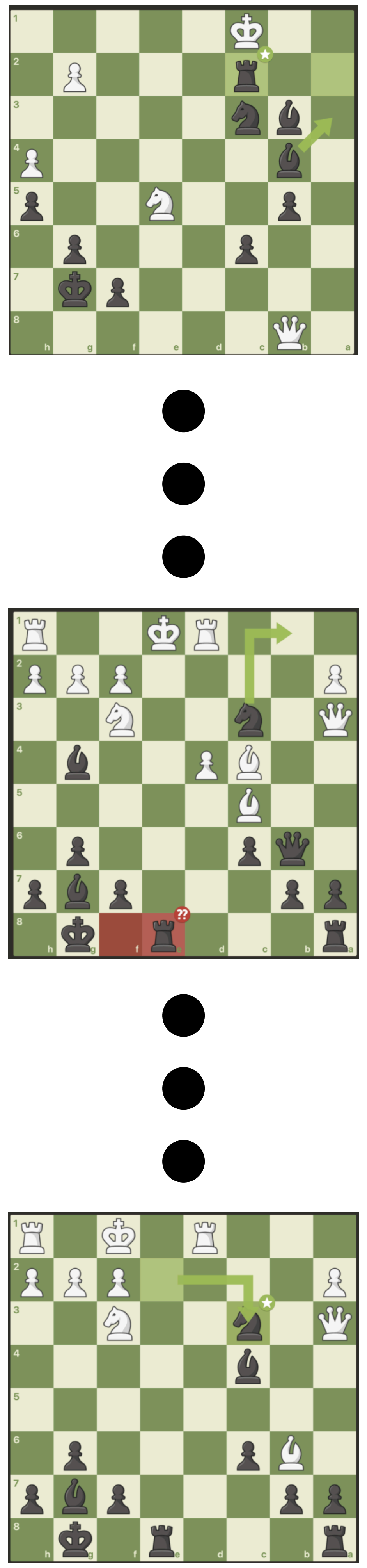
    1.2. **Record** the states of the game you see (board) and the rewards

2. After the game:

    2.1. **Update** your **value function** for all the board configurations you saw

    2.2. **Update** your **policy**/strategy

    2.3. Goto 1



Why is MC a good idea here?

- **Model free**: Don't need p(s',r|s,a) —the book explaining the rules AND how Adam plays
- It's **adaptive**:
  - What if Adam changes how he plays?? **The book would be wrong!!**
    - *Then the MC agent can change its policy in response*
  - What if Adam disappears and Martha becomes your new opponent? *MC can adapt!*
- **Scalable**: if $|\mathcal{S}|$ is big, then p(s',r|s,a) is big
- **Focused** on relevant data: DP learns the optimal policy. Even in states Adam never plays in, MC does not! It *specializes to its opponent!*

Monte Carlo is a first principles algorithm

- Consider policy evaluation: estimating v_\pi given some policy \pi
- What is the definition of v_\pi? $v_\pi(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \ldots \mid S_t = s] = \mathbb{E}_\pi[G_t \mid S_t = s]$
- v_\pi is equal to the **expected return**
    - It’s the expected value of the *random variable G_t*
- How do we estimate an expected value?
    - We can use a sample average!
    - Generate some samples of G_t, then compute the average of them and that’s it!

Monte Carlo is just a sample average

- Let's use MC to estimate the value function for the start state S_0:
  - **Generate** 30 episodes starting in S_0 and taking actions according to \pi
    - Episode 1: S_0, A_0~\pi, R_1, S_1, A_1~\pi, R_2, S_2, …, **R_365, S_365**
      Episode 1: S_0, A_0~\pi, R_1, S_1, A_1~\pi, R_2, S_2, …, **R_12 S_12**
      …
      Episode 30: S_0, A_0~\pi, R_1, S_1, A_1~\pi, R_2, S_2, …, **R_204 S_204**
  - **Compute**:
    - $G^{(1)}$ for episode 1 = R_1 + R_2 + … + R_365; lets say its 14.25
      …
      $G^{(30)}$ for episode 30 = R_1 + R_2 + … + R_204; lets say its 8.45
  - **Average**: V(S_0) = ($G^{(1)}$ + .. + $G^{(30)}$) = (14.25 + …+ 8.45) / 30 = 11.14

Each episode might be a different length & generate different rewards

The states and rewards observed during the episode will differ from episode to episode

Monte Carlo is just a sample average

- So if we wanted to use MC to estimate the value function for the start state S_0, we do the following:
  - **Generate; Compute; Average**

Review of C2M1
Monte Carlo

Video 1: What is Monte Carlo?

- The term “Monte Carlo” is often used more broadly for any estimation method that relies on **repeated random sampling**
- In RL, Monte-Carlo methods allow us to **estimate values** directly from experience: from **sequences of states, actions, and rewards**.
- Goals:
  - Understand how Monte-Carlo methods can be used to **estimate** value functions from **sample interaction**
  - **Identify problems** that can be solved using Monte-Carlo methods
- *If we only have the model—$p(s',r|s,a)$—can we still do Monte Carlo?*

Video 2: Using Monte Carlo for Prediction

- Discussed the **Monte Carlo Policy Evaluation algorithm.** We also looked at a **results** of using MC to evaluate one particular policy in Blackjack

- Goals:

  - Use Monte Carlo prediction to estimate the value function for a **given policy.**

- *How could policy evaluation be useful in the real world? **Hint**: think of an example like Ad Serving online.*

Every-Visit Monte Carlo prediction, for estimating V

Input: a policy π **to be evaluated**

Initialize:

  $V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

  $Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Loop forever (for each episode):

  Generate an episode following $\pi : S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

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

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

    Append G **to** $Returns(S_t)$

    $V(S_t) \leftarrow$ **average**$(Returns(S_t))$

Video 3: Using Monte Carlo to Estimate Action-Values

- How to estimate q_π instead of v_π with MC: Q(S_t, A_t) instead of V(S_t). We also tackled the exploration problem in MC.
- Goals:
  - Estimate **action-value functions** using Monte Carlo and
  - Understand the importance of **maintaining exploration** in Monte Carlo algorithms
- *Why do we need to explore when learning Q(S_t, .)? **Hint**: imagine policy \pi never chooses action 1 in state S_t?*
- *Why do we care that Q(S_t, .) is accurate for all actions in state S_t? Hint: What is the goal of RL?*

Video 4: Using Monte Carlo Methods for Generalized Policy Iteration

- Our first **control Monte Carlo** algorithm. Using **Exploring Starts** to handle the exploration problem
- Goals:
  - Understand how to use Monte Carlo methods to implement a **GPI algorithm**.
- *We can think of Dynamic programming algorithms as doing policy evaluation (recompute the value func) and improvement (greedify policy in all states) as two interacting processes*
- *We can think of bandit algorithms as doing policy evaluation (updating Q) and improvement (picking a greedy action)* ***on a step by step basis***
- *Monte Carlo methods are said to perform policy evaluation and policy improvement* ***on a ____ by ____ basis****. Fill in the blank.* ***Hint****: how often do you update the policy in a MC method?*

Video 5: Solving the Blackjack Example

- Using Monte Carlo Control with Exploring Starts to learn an optimal policy in Blackjack!
- **Goals**:
  - Apply Monte Carlo with exploring starts to solve an example MDP.
- *What is a major limitation of MC methods for control (the ones we study in this chapter)? **Hint**: think of using **On-policy MC control** to learn to beat Adam in a game of tennis, but Adam sometimes changes his policy during a match.*

Video 6: Epsilon-Soft Policies

- Exploring starts is not always the best idea. Think of estimating the value function for a car on a freeway. Turns out we can combine Monte-Carlo control with epsilon-greedy

- **Goals**:

  - Understand why **Exploring Starts can be problematic in real problems**

  - Describe an alternative exploration method for Monte Carlo control, using **Epsilon-soft policies**

- *Why would Epsilon-soft also be bad?*

Video 7: Why Does Off-Policy Learning Matter?

- Off-policy learning is **another way to handle exploration**. You have one policy called the **behavior policy** in charge of acting, and another policy, called the **target policy** that you want to learn the value function for.

- **Goals**:

  - Understand how off-policy learning can **help** deal with the **exploration problem.**

  - Examples of target policies

  - and examples of behavior policies.

- *Do people perform off-policy learning? Can you think of some examples?*

Video 8: Importance Sampling

- **Statistics review:** estimating the expected value of one random variable, with samples drawn according to a different distribution: estimate $E_{\pi}[X]$ with samples drawn according to distribution *b, where* π != b

- **Goals**:

  - use **importance sampling** to estimate the expected value of a target distribution using samples from a different distribution.

- *Imagine we use data generated by a person in a motion capture suite to learn a policy for a robot (say picking up boxes). The person is generating the training data. The distribution b is coming from the person! What is the challenge using importance sampling with this motion capture data?*

Video 9: Off-Policy MC Prediction

- Now that we know how to use importance sampling, we can use it with Monte Carlo to estimate v_π off-policy. We will do off-policy control later. We keep it simple for now!
- **Goals**:
  - Understand how to **use importance sampling to correct returns**
  - And you will understand how to modify the **Monte Carlo prediction** algorithm for off-policy learning.
- *The importance sampling correction is:*
  prob_episode_according_to_pi / prob_episode_according_to_b
  When could this number be really huge? Should we worry about that?

Practice Question

. (*Exercise 5.5 S&B*) Consider an MDP with a single nonterminal state s and a single action that transitions back to s with probability p and transitions to the terminal state with probability $1 - p$. Let the rewards be $+1$ on all transitions, and let $\gamma = 1$. Suppose you observe one episode that lasts 10 steps, with return of 10. What is the (every-visit) Monte-carlo estimator of the value of the nonterminal state s?

Generate an episode following $\pi : S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

. (*Exercise 5.5 S&B*) Consider an MDP with a single nonterminal state s and a single action that transitions back to s with probability p and transitions to the terminal state with probability $1 - p$. Let the rewards be $+1$ on all transitions, and let $\gamma = 1$. Suppose you observe one episode that lasts 10 steps, with return of 10. What is the (every-visit) Monte-carlo estimator of the value of the nonterminal state s?

- What are the non-terminal states?
  - Just 's'
  - So we only need to compute V(s)
- Let's look at the trajectory:
  s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, T
  - How many times did we visit 's'? 10 times
  - What was the return from the first visit to 's'? 10
  - How many samples of G from 's' do we have? How many returns? 10

. (*Exercise 5.5 S&B*) Consider an MDP with a single nonterminal state s and a single action that transitions back to s with probability p and transitions to the terminal state with probability $1 - p$. Let the rewards be $+1$ on all transitions, and let $\gamma = 1$. Suppose you observe one episode that lasts 10 steps, with return of 10. What is the (every-visit) Monte-carlo estimator of the value of the nonterminal state s?

- Let's look at the trajectory:
  s, a, 1, $\boxed{\text{s,}}$ a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, s, a, 1, $\boxed{\text{s,}}$ a, 1, T
  - What was the return from the first visit to 's'? **10**
  - What was the return from the second visit to 's'? **9**
  - What was the return from the last visit to 's'? **1**
- Since it is every-visit MC, we simply average the 10 returns

Terminology Review

- In Monte Carlo there are **no models, and no bootstrapping**
- **Experience**: data generate by the agent taking actions and getting reward feedback for the action it selected.
  - different from what Dynamic Programming does. DP updates the value of states using $p(s',r|s,a)$. DP knows all the rewards in each state via p
- **Sample episodes**: starting in the start state, run policy pi (select actions according to pi) until termination, recording the states, actions, and rewards observed
- MC methods update the value estimates on an **episode-by-episode** basis. Must wait until the end of an episode to update the values of each state the agent observed

Terminology Review (2)

- **Maintaining exploration:** Why we need exploration in MC. Assume pi never takes action b in state S. If we want to estimate q(S,b) we will have no data about the reward you get from state S when pi chooses action b
- **Exploring starts:** every episode must begin in a random state, and the first action must be randomly selected, even if that action is not what pi would do
    - guarantees we visit every state-action pair
- **Epsilon-soft policies:** a stochastic policy. A policy where each action is selected with at least epsilon probability. (e.g., epsilon-greedy)

Terminology Review (3)

- **Off-policy:** learning about one policy, while following another
  - e.g., learning the value function for the optimal policy (q*) while following some exploration policy b (i.e. b=random_policy)
- **Target policy:** the policy you want to learn *about. W*e always call it pi. We either want to learn v_π or (q* and pi*)
- **Behavior policy:** the policy used to select actions, to generate the data. We always call it *b.* It is usually an exploratory policy (e.g., epsilon-greedy with respect to Q)
- **Importance sampling:** a statistical technique for estimating the expected value when the samples used to compute the average don't match the distribution you want.