CSC415: Introduction to Reinforcement Learning

Lecture 3: Model-Free Policy Evaluation and Control

Dr. Amey Pore

Winter 2026

January 21, 2026

Material taken from Sutton and Barto: Chapter 5 and Chapter 6. Structure adapted from David Silver's and Emma Brunskill's course on Introduction to RL.

Think Pair-wise 1

Question 1: In a tabular MDP asymptotically, value iteration will always yield a policy with the same value as the policy returned by policy iteration.

- True
- False
- Not sure

Question 2: Can value iteration require more iterations than $|\mathcal{A}|^{|\mathcal{S}|}$ to compute the optimal value function? (Assume $|\mathcal{A}|$ and $|\mathcal{S}|$ are small enough that each round of value iteration can be done exactly).

- True
- False
- Not sure

Today's Plan

Last Time:

- Markov reward / decision processes
- Policy evaluation & control when have true model (of how the world works)

Today:

- Policy evaluation without known dynamics & reward models
- Control when don't have a model of how the world works

Evaluation through Direct Experience

- Estimate expected return of policy π
- Only using data from environment (direct experience)[1]
- Why is this important?
- What properties do we want from policy evaluation algorithms?

This Lecture: Policy Evaluation

Estimating the expected return of a particular policy if we don't have access to true MDP models

- Monte Carlo policy evaluation
- Temporal Difference (TD)
- Course logistics

Recall

- Definition of Return, G_t (for a MRP)
  - Discounted sum of rewards from time step t to horizon

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

- Definition of State Value Function, $V^\pi(s)$
  - Expected return starting in state s under policy π

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s]$$

- Definition of State-Action Value Function, $Q^\pi(s, a)$
  - Expected return starting in state s, taking action a and following policy π

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \\ &= \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s, a_t = a] \end{aligned}$$

Recall: Dynamic Programming for Policy Evaluation

- In a Markov decision process

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[G_t | s_t = s] \\ &= \mathbb{E}_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s] \\ &= R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s' | s, \pi(s)) V^{\pi}(s') \end{aligned}$$

- If given dynamics and reward models, can do policy evaluation through dynamic programming

$$V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s' | s, \pi(s)) V_{k-1}^{\pi}(s') \tag{1}$$

- **Note**: before convergence, V_k^{π} is an estimate of V^{π}
- In Equation 1 we are substituting $\sum_{s' \in S} p(s' | s, \pi(s)) V_{k-1}^{\pi}(s')$ for $\mathbb{E}_{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s]$.
- This substitution is an instance of **bootstrapping**

This Lecture: Policy Evaluation

Estimating the expected return of a particular policy if we don't have access to true MDP models

- **Monte-Carlo policy evaluation**
- Temporal Difference (TD)
- Course logistics

Monte-Carlo Policy Evaluation

- Goal: learn V^π from episodes of experience under policy π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots + \gamma^{T_i - t} r_{T_i}$ in MDP M under policy π
- $V^\pi(s) = \mathbb{E}_{\tau \sim \pi}[G_t | s_t = s]$
  - Expectation over trajectories τ generated by following π
- Monte-Carlo policy evaluation uses **empirical mean** return instead of *expected* return

Monte-Carlo Policy Evaluation

- If trajectories are all finite, sample set of trajectories & average returns
- Does not require MDP dynamics/rewards
- Does not assume state is Markov
- Can be applied to episodic MDPs
  - Averaging over returns from a complete episode
  - Requires each episode to terminate

First-Visit Monte-Carlo On Policy Evaluation

Initialize $N(s) = 0, G(s) = 0 \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \ldots \gamma^{T_i - 1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For each time step t until T_i (the end of the episode i)
  - If this is the **first** time t that state s is visited in episode i
    - Increment counter of total first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $V^{\pi}(s) = G(s)/N(s)$

Every-Visit Monte-Carlo On Policy Evaluation

Initialize $N(s) = 0, G(s) = 0, \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \ldots \gamma^{T_i - 1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For each time step t until T_i (the end of the episode i)
  - state s is the state visited at time step t in episodes i
  - Increment counter of total visits: $N(s) = N(s) + 1$
  - Increment total return $G(s) = G(s) + G_{i,t}$
  - Update estimate $V^{\pi}(s) = G(s)/N(s)$

Worked Example: Monte-Carlo On Policy Evaluation

Initialize $N(s) = 0, G(s) = 0, \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \ldots \gamma^{T_i - 1} r_{i,T_i}$
- For each time step t until T_i (the end of the episode i)
  - If this is the **first** time t that state s is visited in episode i (for first visit MC)
    - Increment counter of total first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $V^{\pi}(s) = G(s)/N(s)$
- Mars rover: R(s) = [1 0 0 0 0 0 +10]
- Trajectory = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$
- Let $\gamma < 1$. Compute the first visit & every visit MC estimates of s_2.
- See solutions at the end of the slides

Incremental Mean

The mean $V_1, V_2, \ldots$ of a sequence $G_1, G_2, \ldots$ can be computed incrementally,

$$\begin{aligned} V_k &= \frac{1}{k} \sum_{j=1}^{k} G_j \\ &= \frac{1}{k} \left(G_k + \sum_{j=1}^{k-1} G_j \right) \\ &= \frac{1}{k} (G_k + (k-1) V_{k-1}) \\ &= V_{k-1} + \frac{1}{k} (G_k - V_{k-1}) \end{aligned}$$

Incremental Monte-Carlo Updates

- Update $V^{\pi}(s)$ incrementally after episode $s_1, a_1, r_2, \ldots, s_T$
- For each state s_t with return $G_{i,t}$

$$N(s_t) \leftarrow N(s_t) + 1$$
$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \frac{1}{N(s_t)}(G_{i,t} - V^{\pi}(s_t))$$

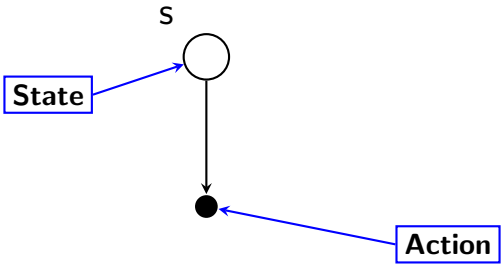
- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(G_{i,t} - V^{\pi}(s_t))$$

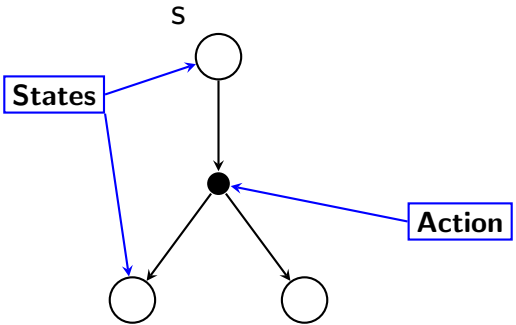
Incremental Monte Carlo (MC) On Policy Evaluation

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \ldots \gamma^{T_i - 1} r_{i,T_i}$
- for $t = 1 : T_i$ where T_i is the length of the i-th episode
  - $V^\pi(s_{it}) = V^\pi(s_{it}) + \alpha(G_{i,t} - V^\pi(s_{it}))$
- We will see many algorithms of this form with a learning rate, target, and incremental update

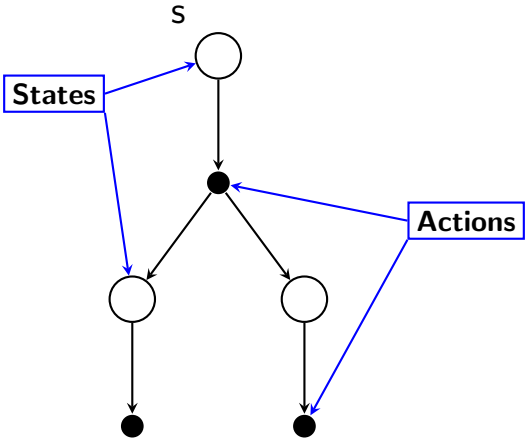
Policy Evaluation Diagram



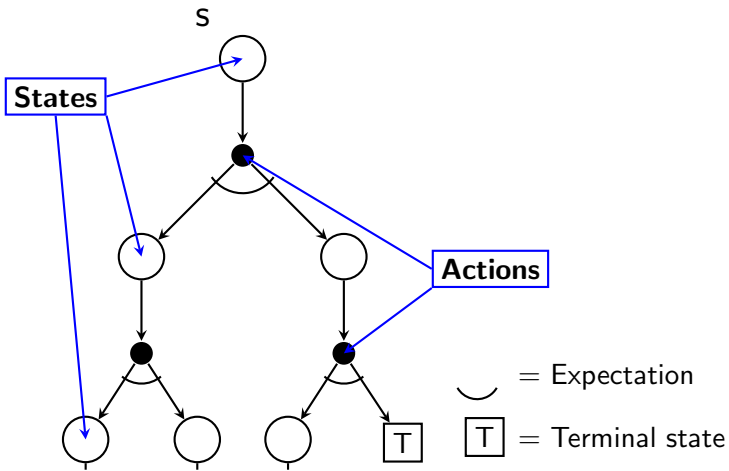
Policy Evaluation Diagram



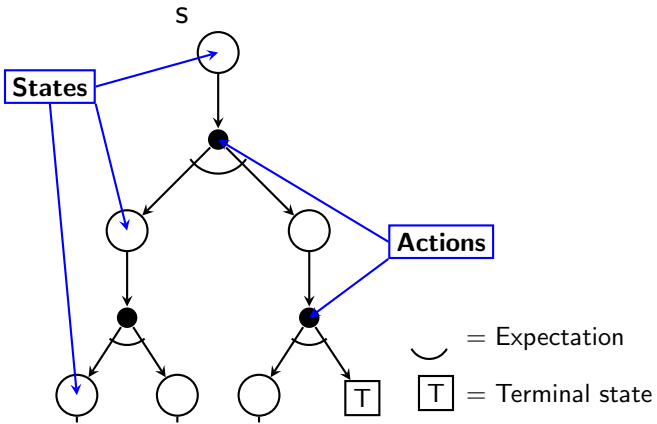
Policy Evaluation Diagram



Policy Evaluation Diagram

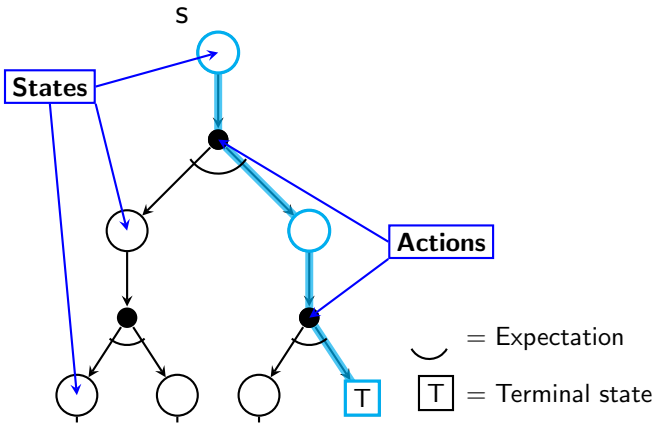


Monte-Carlo Policy Evaluation



$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$

Monte-Carlo Policy Evaluation



$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$

MC updates the value estimate using a **sample** of the return to approximate an expectation

Evaluation of the Quality of a Policy Estimation Approach

- **Consistency**: with enough data, does the estimate converge to the true value of the policy?
- **Computational complexity**: as we get more data, computational cost of updating estimate
- **Memory requirements**
- **Statistical efficiency** (intuitively, how does the accuracy of the estimate change with the amount of data)
- **Empirical accuracy**, often evaluated by mean squared error

Evaluation of the Quality of a Policy Estimation: Bias, Variance and MSE

- Consider a statistical model that is parameterized by θ and that determines a probability distribution over observed data $P(x|\theta)$
- Consider a statistic $\hat{\theta}$ that provides an estimate of θ and is a function of observed data x
  - E.g. for a Gaussian distribution with known variance, the average of a set of i.i.d data points is an estimate of the mean of the Gaussian
- Definition: the bias of an estimator $\hat{\theta}$ is:

$$Bias_\theta(\hat{\theta}) = \mathbb{E}_{x|\theta}[\hat{\theta}] - \theta$$

- Definition: the variance of an estimator $\hat{\theta}$ is:

$$Var(\hat{\theta}) = \mathbb{E}_{x|\theta}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2]$$

- Definition: mean squared error (MSE) of an estimator $\hat{\theta}$ is:

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias_\theta(\hat{\theta})^2$$

Evaluation of the Quality of a Policy Estimation: Consistent Estimator

- Let n be the number of data points x used to estimate the parameter θ and call the resulting estimate of θ using that data $\hat{\theta}_n$

Consistency

Then the estimator $\hat{\theta}_n$ is **consistent** if, for all $\epsilon > 0$:

$$\lim_{n \to \infty} Pr(|\hat{\theta}_n - \theta| > \epsilon) = 0$$

- If an estimator is unbiased (bias $= 0$) is it consistent?

Properties of Monte-Carlo On Policy Evaluators

Properties:

- First-visit Monte Carlo
  - V^π estimator is an unbiased estimator of true $\mathbb{E}_\pi[G_t|s_t = s]$
  - By law of large numbers, as $N(s) \to \infty$, $V^\pi(s) \to \mathbb{E}_\pi[G_t|s_t = s]$
- Every-visit Monte Carlo
  - V^π every-visit MC estimator is a **biased** estimator of V^π
  - But consistent estimator and often has better MSE
- Incremental Monte Carlo
  - Properties depends on the learning rate α

Monte-Carlo (MC) Policy Evaluation Key Limitations

- Generally high variance estimator
  - Reducing variance can require a lot of data
  - In cases where data is very hard or expensive to acquire, or the stakes are high, MC may be impractical
- Requires episodic settings
  - Episode must end before data from episode can be used to update V

Monte Carlo (MC) Policy Evaluation Summary

- Aim: estimate $V^{\pi}(s)$ given episodes generated under policy π
  - $s_1, a_1, r_1, s_2, a_2, r_2, \ldots$ where the actions are sampled from π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots$ under policy π
- $V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s]$
- Simple: Estimates expectation by empirical average (given episodes sampled from policy of interest)
- Updates V estimate using **sample** of return to approximate the expectation
- Does not assume Markov process
- Converges to true value under some (generally mild) assumptions
- **Note**: Sometimes is preferred over dynamic programming for policy evaluation *even if know the true dynamics model and reward*

This Lecture: Policy Evaluation

Estimating the expected return of a particular policy if we don't have access to true MDP models

- Monte Carlo policy evaluation
- **Temporal Difference (TD)**
- Course logistics

Temporal Difference Learning

- "If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning." – Sutton and Barto 2017
- Combination of Monte Carlo & dynamic programming methods
- Model-free
- Can be used in episodic or infinite-horizon non-episodic settings
- Immediately updates estimate of V after each (s_t, a_t, r_t, s_{t+1}) tuple

Temporal Difference Learning for Estimating V

- Aim: estimate $V^\pi(s)$ online from experience under policy π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$ in MDP M under policy π
- In incremental every-visit MC, update towards *actual* return G_t

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(G_t - V^\pi(s_t))$$

- **Idea**: update value $V^\pi(s_t)$ toward *estimated* return using $r_t + \gamma V^\pi(s_{t+1})$

$$V^\pi(s_t) = V^\pi(s_t) + \alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s_t))$$

Temporal Difference [$TD(0)$] Learning

- Aim: estimate $V^{\pi}(s)$ online from experience under policy π
  - $s_1, a_1, r_1, s_2, a_2, r_2, \ldots$ where the actions are sampled from π
- TD(0) learning / 1-step TD learning: update estimate towards target

$$V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha(\underbrace{[r_t + \gamma V^{\pi}(s_{t+1})]}_{\text{TD target}} - V^{\pi}(s_t))$$

- TD(0) error:

$$\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

- Update can be done after each step! (online, model-free) $(S_t, A_t, R_{t+1}, S_{t+1})$ tuple
- Don't need episodic setting

Temporal Difference [$TD(0)$] Learning Algorithm

- Input: α
- Initialize $V^{\pi}(s) = 0, \forall s \in S$
- Loop
  - Sample **tuple** (s_t, a_t, r_t, s_{t+1})
  - $V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha(\underbrace{[r_t + \gamma V^{\pi}(s_{t+1})]}_{\text{TD target}} - V^{\pi}(s_t))$

Worked Example TD Learning

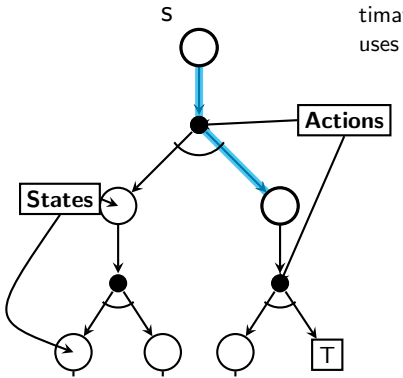
- **Input:** α
- **Initialize** $V^\pi(s) = 0, \forall s \in S$
- **Loop**
  - Sample **tuple** (s_t, a_t, r_t, s_{t+1})
  - $V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$
- **Example:**
  - Mars rover: $R = [1\ 0\ 0\ 0\ 0\ 0\ +10]$ for any action
  - $\pi(s) = a_1 \forall s, \gamma = 1$. any action from s_1 and s_7 terminates episode
  - Trajectory $= (s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$
  - TD estimate of all states (init at 0) with $\alpha = 1, \gamma < 1$ at end of this episode?
- First visit MC estimate of V of each state? $[1\ \gamma\ \gamma^2\ 0\ 0\ 0\ 0]$

Temporal Difference (TD) Policy Evaluation

$$V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha([r_t + \gamma V^{\pi}(s_{t+1})] - V^{\pi}(s_t))$$

TD updates the value estimate using a **sample** of s_{t+1} to approximate an expectation

TD updates the value estimate by **bootstrapping**, uses estimate of $V(s_{t+1})$



= Expectation

T = Terminal state

Think Pair wise 2: Temporal Difference [$TD(0)$] Learning Algorithm

- Input: α
- Initialize $V^\pi(s) = 0, \forall s \in S$
- Loop
  - Sample **tuple** (s_t, a_t, r_t, s_{t+1})
  - $V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$

Select all that are true

1. If $\alpha = 0$ TD will weigh the TD target more than the past V estimate
2. If $\alpha = 1$ TD will update the V estimate to the TD target
3. If $\alpha = 1$ TD in MDPs where the policy goes through states with multiple possible next states, V may oscillate forever
4. There exist deterministic MDPs where $\alpha = 1$ TD will converge

Summary: Temporal Difference Learning

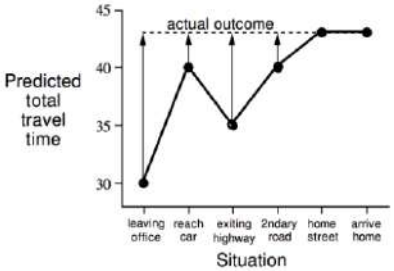
- Combination of Monte Carlo & dynamic programming methods
- Model-free
- **Bootstraps and samples**
- Can be used in episodic or infinite-horizon non-episodic settings
- Immediately updates estimate of V after each $(S_t, A_t, R_{t+1}, S_{t+1})$ tuple
- Biased estimator (early on will be influenced by initialization, and won't be unbiased estimator)
- Generally lower variance than Monte Carlo policy evaluation
- Consistent estimator if learning rate α satisfies same conditions specified for incremental MC policy evaluation to converge
- **Note**: algorithm I introduced is TD(0). In general can have approaches that interpolate between TD(0) and Monte Carlo approach

Driving Home Example

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Driving Home Example: MC vs. TD

Changes recommended by Monte Carlo methods (α=1)



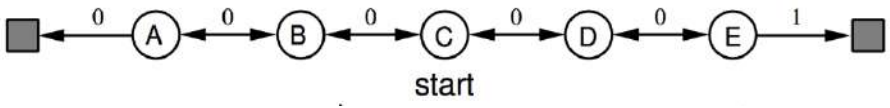
Changes recommended by TD methods (α=1)

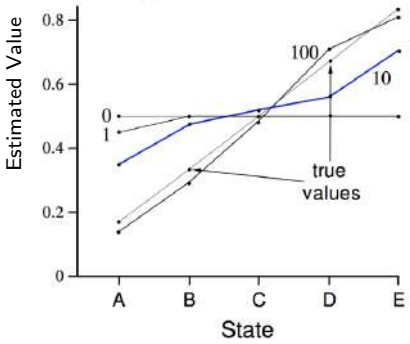
Comparison: DP vs MC vs TD

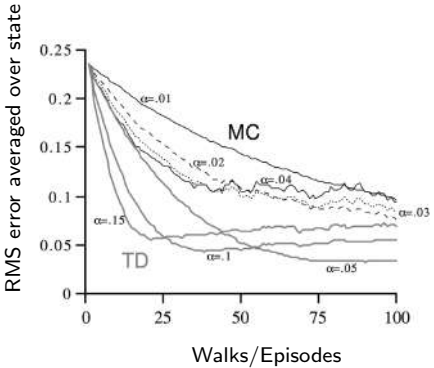
	DP	**MC**	**TD**
Data usage	All transitions	Complete episodes	One step
Bootstrapping	Yes	No	Yes
Sampling	No	Yes	Yes
Model required	Yes	No	No
Computational cost	High	Low	Low
Bias	Biased	Unbiased (First-Visit)	Biased
Variance	None	High	Low
Works online[2]	Yes	No	Yes
Assumes Markov property	Yes	No	Yes

[2]Online learning refers to updating estimates step-by-step during an episode, rather than waiting until the end.

Random Walk: MC vs. TD







Batch MC and TD

- Batch (Offline) solution for finite dataset
  - Given set of K episodes
  - Repeatedly sample an episode from K
  - Apply MC or TD(0) to the sampled episode
- What do MC and TD(0) converge to?

AB Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

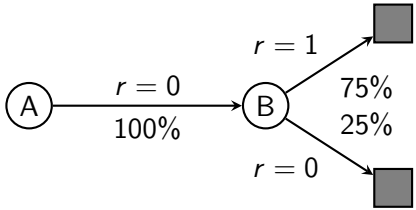
What is $V(A), V(B)$?

AB Example

Two states A, B; no discounting; 8 episodes of experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

What is $V(A), V(B)$?



Batch MC and TD: Convergence

- Monte Carlo in batch setting converges to min MSE (mean squared error)
  - Minimize loss with respect to observed returns
  - In AB example, $V(A) = 0$
- TD(0) converges to DP policy V^{π} for the MDP with the maximum likelihood model estimates
- **Aka same as dynamic programming with certainty equivalence!**
  - Maximum likelihood Markov decision process model

$$\hat{P}(S_{t+1}|S_t, A_t) = \frac{1}{N(S_t, A_t)} \sum_{k=1}^{i} \mathbb{I}(S_k = S_t, A_k = A_t, S_{k+1} = S_{t+1})$$

$$\hat{R}(S_t, A_t) = \frac{1}{N(S_t, A_t)} \sum_{k=1}^{i} \mathbb{I}(S_k = S_t, A_k = A_t) R_{k+1}$$

  - Compute V^{π} using this model
  - In AB example, $V(A) = 0.75$

This Lecture: Policy Evaluation

Estimating the expected return of a particular policy if we don't have access to true MDP models

- Monte Carlo policy evaluation
- Temporal Difference (TD)
- **Course logistics**

Course Logistics

- Those not added on MarkUs: Send an email [CSC415].
- Sample Mid-term exam uploaded last week.
- Solutions will be out tomorrow.
- Lab 2 will be help tomorrow.
- Check Piazza for weekly updates.

Project Goals

Please read the project guidelines.

- Assignement 1: Literature review and baseline
  - Topics: Any topic covered in the course
  - We provided the ones since these are fundamental challenges
  - Available open-source repos.
  - Talk to TAs and instructor if outside the pdf
- Project proposal:
  - Based on challenges identified in assignment 1.
  - Often needs more literature survey
  - Form an hypothesis, which environments to use
- Project report:
  - Your Investigations

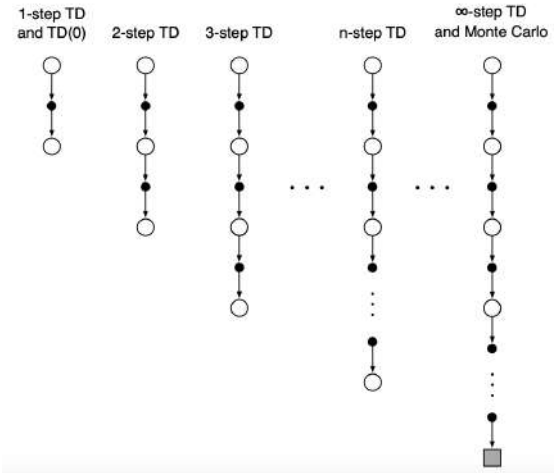
Project Ideas

Please read the project guidelines.

- New idea for RL
  - Introducing new regularization to make learning faster.
  - New exploration strategy
  - New architecture of NN
- New application of RL
  - Robotics: Generalization to new objects, colors.
  - LLMs: reasoning improvements for local LLMs
- Future works of existing papers
- More ideas here:
  `https://cs224r.stanford.edu/projects/cs224r_final_projects.html`

Thank you!

n-Step Prediction



n-Step Return

- Consider the following n-step returns for $n = 1, 2, \infty$:

$$\begin{aligned} n = 1 \quad \text{(TD)} \quad & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n = 2 \quad & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ \vdots \quad & \vdots \\ n = \infty \quad \text{(MC)} \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T \end{aligned}$$

- Define the n-step return

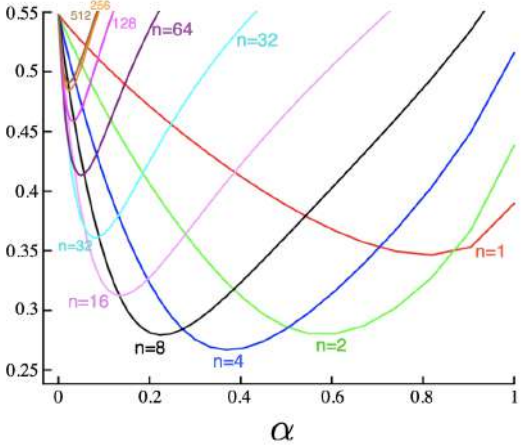
$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Large Random walk example

Average RMS error over 19 states and first 10 episodes



Think pair-wise 1 solution

- In a tabular MDP asymptotically value iteration will always yield a policy with the same value as the policy returned by policy iteration
  Answer. True. Both are guaranteed to converge to the optimal value function and a policy with an optimal value

- Can value iteration require more iterations than $|A|^{|S|}$ to compute the optimal value function? (Assume $|A|$ and $|S|$ are small enough that each round of value iteration can be done exactly).
  Answer: True. As an example, consider a single state, single action MDP where $r(s,a) = 1, \gamma = .9$ and initialize $V_0(s) = 0$. $V^*(s) = \frac{1}{1-\gamma}$ but after the first iteration of value iteration, $V_1(s) = 1$.

Example: Mars Rover - Monte Carlo

Mars rover: $R = [1\ 0\ 0\ 0\ 0\ 0\ +10]$ for any action
$\pi(s) = a_1\ \forall s$, $\gamma = 1$. Any action from s_1 and s_7 terminates episode

Trajectory: $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1,$ terminal$)$

First visit MC estimate of V of each state?

- s_1: First visit at end, $G = 1$, so $V(s_1) = 1$
- s_2: First visit at $t = 1$, $G_1 = 0 + 0 + 1 = 1$, so $V(s_2) = 1$ (or γ if $\gamma \neq 1$)
- s_3: First visit at $t = 0$, $G_0 = 0 + 0 + 0 + 1 = 1$, so $V(s_3) = 1$ (or γ^2 if $\gamma \neq 1$)
- Other states: Not visited, $V = 0$

Answer: $[1\ \gamma\ \gamma^2\ 0\ 0\ 0\ 0]$ (with $\gamma = 1$: $[1\ 1\ 1\ 0\ 0\ 0\ 0]$)

Example: Mars Rover TD Learning

- **Initialize** $V(s) = 0 \quad \forall s$
- Trajectory: $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$
- Update rule: $V(s_t) \leftarrow V(s_t) + \alpha(r_t + \gamma V(s_{t+1}) - V(s_t))$ with $\alpha = 1$
- Step 1: $(s_3 \xrightarrow{0} s_2)$. Target $0 + \gamma V(s_2) = 0$.
  - $V(s_3) \leftarrow 0 + 1(0 - 0) = 0$
- Step 2: $(s_2 \xrightarrow{0} s_2)$. Target $0 + \gamma V(s_2) = 0$.
  - $V(s_2) \leftarrow 0 + 1(0 - 0) = 0$
- Step 3: $(s_2 \xrightarrow{0} s_1)$. Target $0 + \gamma V(s_1) = 0$.
  - $V(s_2) \leftarrow 0 + 1(0 - 0) = 0$
- Step 4: $(s_1 \xrightarrow{1} T)$. Target $1 + \gamma(0) = 1$.
  - $V(s_1) \leftarrow 0 + 1(1 - 0) = 1$
- **Final estimates:** $V(s_1) = 1$, others 0.
- Note: $V(s_2)$ remains 0 despite being visited, because standard TD(0) bootstraps from the current estimate of the next state (which was 0).

Think Pair Wise 2 Solution

TD Update Rule: $V(S_t) \leftarrow V(S_t) + \alpha \underbrace{(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))}_{\text{TD Error } \delta_t}$

1. If $\alpha = 0$ TD will weigh the TD target more than the past V estimate.
   False. $V(S_t) \leftarrow V(S_t) + 0 \cdot \delta_t = V(S_t)$. It keeps the past estimate.
2. If $\alpha = 1$ TD will update the V estimate to the TD target.
   True. $V(S_t) \leftarrow V(S_t) + 1 \cdot (R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) = R_{t+1} + \gamma V(S_{t+1})$.
3. If $\alpha = 1$ and next states are stochastic, V may oscillate forever.
   True. The target $R_{t+1} + \gamma V(S_{t+1})$ depends on the sampled $S_{t+1} \sim P(\cdot|S_t, A_t)$. If $S_t \to S_A$ or $S_t \to S_B$, $V(S_t)$ will jump between targets.
4. There exist deterministic MDPs where $\alpha = 1$ TD will converge.
   True. In deterministic MDPs, R_{t+1} and S_{t+1} are fixed for (S_t, A_t). The update $V(S_t) \leftarrow R_{t+1} + \gamma V(S_{t+1})$ is equivalent to Value Iteration.

Solution: AB Example

Monte Carlo:

- Average return for state A
- State A only appears in one episode: `A, 0, B, 0`
- Return for this episode is $G = 0 + 0 = 0$
- Hence, $V(A) = 0$

Temporal Difference:

- For B, TD converges to the average reward:

$$V(B) = \frac{6/8 \times 1 + 2/8 \times 0}{1} = 0.75$$

- For A, TD bootstraps using $V(B)$:
- $V(A) = 0 + V(B) = 0.75$

Certainty Equivalence for Mars Rover example

- Mars rover: $R = [1\ 0\ 0\ 0\ 0\ 0\ +10]$ for any action
- $\pi(s) = a_1\ \forall s, \gamma = 1$. Any action from s_1 and s_7 terminates episode
- Trajectory $= (s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1,$ terminal$)$
- First visit MC estimate of V of each state? $[1\ \gamma\ \gamma^2\ 0\ 0\ 0\ 0]$
- TD estimate of all states (init at 0) with $\alpha = 1$ is $[1\ 0\ 0\ 0\ 0\ 0\ 0]$
- Optional exercise: What is the certainty equivalent estimate?
- $\hat{r} = [1\ 0\ 0\ 0\ 0\ 0\ 0], \hat{p}(\text{terminate}|s_1, a_1) = \hat{p}(s_2|s_3, a_1) = 1$

CSC415: Introduction to Reinforcement Learning

Lecture 3: Model-Free Policy Evaluation and Control

Dr. Amey Pore

Winter 2026

January 21, 2026

Material taken from Sutton and Barto: Chapter 5 and Chapter 6. Structure adapted from David Silver's and Emma Brunskill's course on Introduction to RL.

Today's Lecture

- Last part:
  - Model-free prediction
  - *Estimate* the value function of an *unknown* MDP
- This part:
  - Model-free control
  - *Optimise* the value function of an *unknown* MDP

Uses of Model-Free Control

Some example problems that can be modelled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

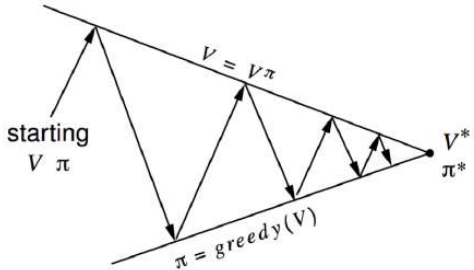
- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

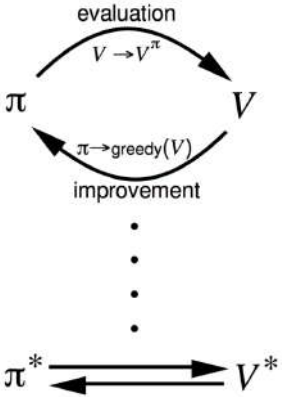
Model-free control can solve these problems

Outline

- **Monte-Carlo Control**
- Temporal Difference Methods for Control (SARSA, Q-Learning)

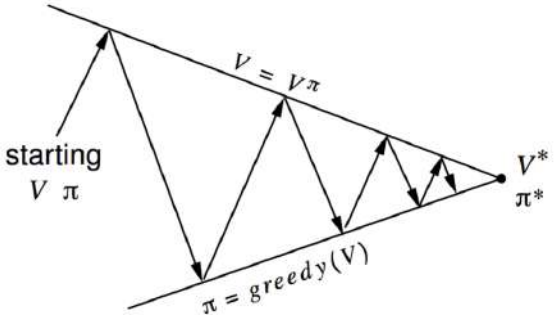
Recall: Generalized Policy Iteration





- Policy evaluation Estimate V^π
  Iterative policy evaluation
- Policy improvement Generate $\pi' \geq \pi$
  Greedy policy improvement

Generalized Policy Iteration with MC evaluation



- Policy evaluation: MC policy evaluation, V^{π}
- Policy improvement: Greedy policy improvement

Model-Free Policy Improvement with Q-Value

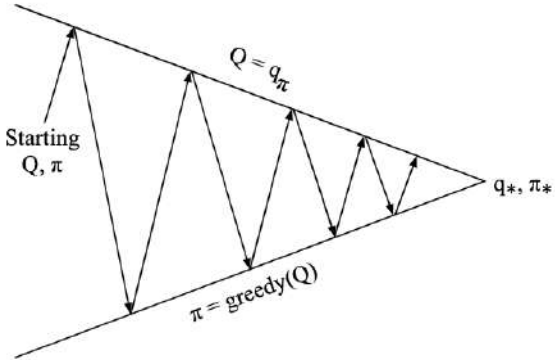
- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^{\pi_i}(s') \right)$$

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

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

Generalized Policy Iteration with Q value function

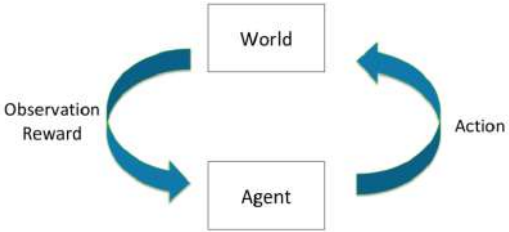


- Policy evaluation: MC policy evaluation, $Q = Q^\pi$
- Policy improvement: Greedy policy improvement?

Model-free Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement: update π given Q^π
- May need to modify policy evaluation:
  - If π is deterministic, can't compute $Q(s, a)$ for any $a \neq \pi(s)$
- How to interleave policy evaluation and improvement?
  - Policy improvement is now using an estimated Q

The Problem of Exploration



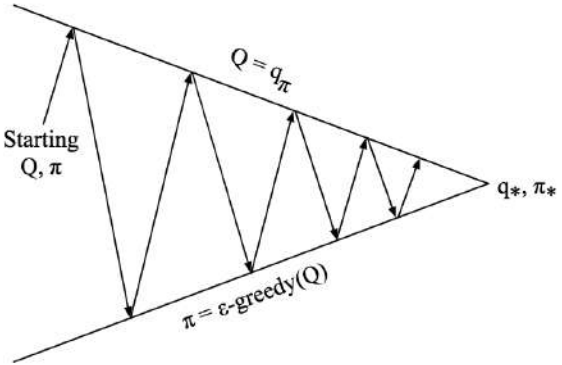
- Goal: Learn to select actions to maximize total expected future reward
- Problem: Can't learn about actions without trying them (need to *explore*)
- Problem: But if we try new actions, spending less time taking actions that our past experience suggests will yield high reward (need to *exploit* knowledge of domain to achieve high rewards)

ϵ-greedy Policies

- Simple idea to balance exploration and achieving rewards
- Let $|\mathcal{A}|$ be the number of actions
- with probability $1 - \epsilon$, choose the greedy action.
- with probability ϵ, choose an action uniformly at random

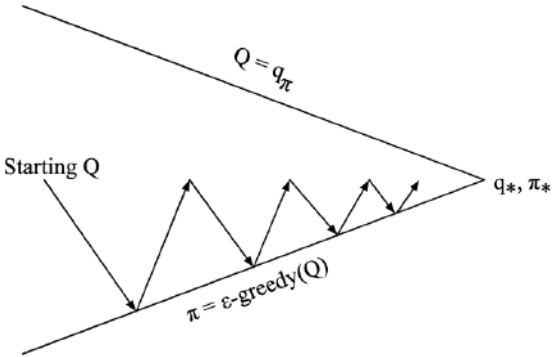
$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}|} + 1 - \epsilon & \text{if } a = \arg\max_{a' \in \mathcal{A}} Q(s, a') \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

Monte-Carlo Policy Iteration



- Policy evaluation: MC policy evaluation, $Q = Q^\pi$
- Policy improvement: ϵ-Greedy policy improvement.

Monte-Carlo Policy Iteration



- Policy evaluation: MC policy evaluation, $Q \approx Q^\pi$
- Policy improvement: ϵ-Greedy policy improvement.

ϵ-Greedy Policy Improvement

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to Q^π is an improvement, $V^{\pi'}(s) \geq V^\pi(s)$

- Proof at the end of the slides.

Recall Monte Carlo Policy Evaluation, Now for Q

1: Initialize $Q(s,a) = 0, N(s,a) = 0, \forall (s,a), k = 1$, Input $\epsilon = 1, \pi$
2: **loop**
3: Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π
4: Compute $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \cdots + \gamma^{T-1} r_{k,T}, \forall t$
5: **for** $t = 1, \ldots, T$ **do**
6: **if** First visit to (s,a) in episode k **then**
7: $N(s,a) = N(s,a) + 1$
8: $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
9: **end if**
10: **end for**
11: $k = k + 1$
12: **end loop**

Monte Carlo Online Control / On Policy Improvement

Initialize $Q(s,a) = 0, N(s,a) = 0, \forall (a,s)$, Set $\epsilon = 1, k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
    Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
    Compute $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \cdots + \gamma^{T-1} r_{k,T}$
    for $t = 1, \ldots, T$ **do**
        if First visit to (s,a) in episode k **then**
            $N(s,a) = N(s,a) + 1$
            $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
        end if
    end for
    $k = k + 1, \epsilon = 1/k$
    $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

Monte Carlo Online Control / On Policy Improvement

Initialize $Q(s,a) = 0, N(s,a) = 0, \forall (s,a)$, Set $\epsilon = 1, k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
  Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
  Compute $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \cdots + \gamma^{T-1} r_{k,T}$
  for $t = 1, \ldots, T$ **do**
    if First visit to (s,a) in episode k **then**
      $N(s,a) = N(s,a) + 1$
      $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
    end if
  end for
  $k = k + 1, \epsilon = 1/k$
  $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

- Is Q an estimate of Q^*? When might this procedure fail to compute the optimal Q^*?

Greedy in the Limit of Infinite Exploration (GLIE)

Definition of GLIE

- All state-action pairs are visited an infinite number of times: $\lim_{i \to \infty} N_i(s, a) \to \infty$
- Behavior policy (policy used to act in the world) converges to greedy policy

- A simple GLIE strategy is ϵ-greedy where ϵ is reduced to 0 with the following rate: $\epsilon_i = 1/i$

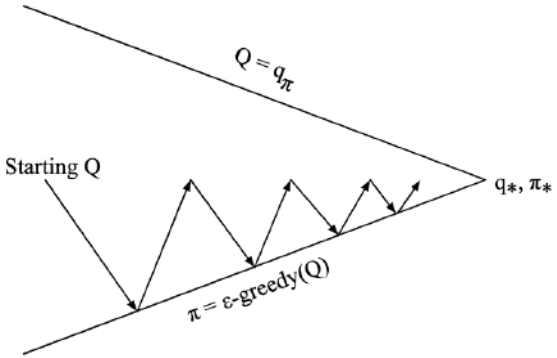
Theorem

GLIE Monte-Carlo control converges to the optimal state-action value function $Q(s, a) \to Q^*(s, a)$

Outline

- Monte-Carlo Control
- **Temporal Difference Methods for Control (SARSA, Q-Learning)**

On-Policy Control with SARSA



Every Time Step

- Policy evaluation: SARSA, $Q \approx Q^\pi$
- Policy improvement: ϵ-Greedy policy improvement.

General Form of SARSA Algorithm

1: Set initial ϵ-greedy policy π, $t = 0$, initial state $s = s_0$
2: Take $a_t \sim \pi(s_t)$ // Sample action from policy
3: Observe (r_t, s_{t+1})
4: **loop**
5: Take action $a_{t+1} \sim \pi(s_{t+1})$
6: Observe (r_t, s_{t+2})
7: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
8: $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
9: $t = t + 1$
10: **end loop**

- See worked example with Mars rover at end of slides

Convergence Properties of SARSA

Theorem

SARSA for finite-state and finite-action MDPs converges to the optimal action-value, $Q(s,a) \to Q^*(s,a)$, under the following conditions:

1. The policy sequence $\pi_t(a|s)$ satisfies the condition of GLIE
2. The step-sizes α_t satisfy the Robbins-Munro sequence such that

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

- For ex, $\alpha_t = \frac{1}{t}$ satisfies the above condition.

On and Off-Policy Learning

- On-policy learning
  - "Learn on the job"
  - Learn about policy π from experience sampled from π
- Off-policy learning
  - "Look over someone's shoulder"
  - Learn about policy π from experience sampled from μ

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, \ldots, S_T\} \sim \mu$$

- Why is this important?
  - Learn from observing humans or other agents
  - Re-use experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
  - Learn about *optimal* policy while following *exploratory* policy
  - Learn about *multiple* policies while following *one* policy

Q-Learning: Learning the Optimal State-Action Value

- SARSA is an **on-policy** learning algorithm
  - SARSA estimates the value of the current behavior policy (policy using to take actions in the world)
  - And then updates that (behavior) policy
- Alternatively, can we directly estimate the value of π^* while acting with another behavior policy π_b?
- Yes! Q-learning, an **off-policy** RL algorithm

Q-Learning

- Estimate the Q-value of pi^* while acting with another behavior policy π_b
- Key idea: Maintain Q estimates and use bootstrap for best future value.

Recall SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$$

Q-learning

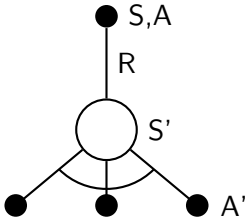
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

Q-Learning with ϵ-greedy Exploration

1: Initialize $Q(s,a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}, t = 0$, initial state $s_t = s_0$
2: Set π_b to be ϵ-greedy w.r.t. Q
3: **loop**
4: Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
5: Observe (r_t, s_{t+1})
6: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t))$
7: $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
8: $t = t+1$
9: **end loop**

- See optional worked example and optional understanding check at the end of the slides

Q-Learning Control Algorithm



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$

Theorem

Q-learning control converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$

Thank you!

Homework: Model-free Generalized Policy Improvement

- Consider policy iteration
- Repeat:
  - Policy evaluation: compute Q^{π}
  - Policy improvement $\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a)$
- **Question:** Is this π_{i+1} deterministic or stochastic? Assume for each state s there is a unique $\max_a Q^{\pi_i}(s, a)$.
- **Answer:** Deterministic, Stochastic, Not Sure
- Now consider evaluating the policy of this new π_{i+1}. Recall in model-free policy evaluation, we estimated V^{π}, using π to generate new trajectories
- **Question:** Can we compute $Q^{\pi_{i+1}}(s, a) \forall s, a$ by using this π_{i+1} to generate new trajectories?

Proof: ϵ-Greedy Policy Improvement

$$
\begin{aligned}
Q^{\pi}(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) Q^{\pi}(s, a) \\
&= \epsilon/|\mathcal{A}| \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} Q^{\pi}(s, a) \\
&\geq \epsilon/|\mathcal{A}| \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/|\mathcal{A}|}{1 - \epsilon} Q^{\pi}(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s, a) = V^{\pi}(s)
\end{aligned}
$$

Therefore from policy improvement theorem, $V^{\pi'}(s) \geq V^{\pi}(s)$

Optional Worked Example: MC for On Policy Control Solution

- Mars rover with new actions:
  - $r(-, a_1) = [1\ 0\ 0\ 0\ 0\ 0\ +10]$, $r(-, a_2) = [0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$.
- Assume current greedy $\pi(s) = a_1 \forall s, \epsilon = 0.5$. $Q(s, a) = 0$ for all (s, a)
- Sample trajectory from ϵ-greedy policy
- Trajectory $= (s_1, a_1, 0, s_2, 0, s_2, 0, s_2, 0, s_1, a_1, 1, \text{terminal})$
- First visit MC estimate of Q of each (s, a) pair?
- $Q^*_{-,a_1} = [1\ 0\ 0\ 0\ 0\ 0\ 0]$
- After this trajectory:
- $Q^*_{-,a_2} = [0\ 1\ 0\ 0\ 0\ 0\ 0]$
- The new greedy policy would be: $\pi = [1\ 2\ 1$ tie tie tie tie$]$
- If $\epsilon = 1/3$, prob of selecting a_1 in s_1 in the new ϵ-greedy policy is $5/6$.

Q-Learning with ϵ-greedy Exploration

- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal Q^*?
- What conditions are sufficient to ensure that Q-learning with ϵ-greedy exploration converges to optimal π^*?

Worked Example: SARSA for Mars Rover

1: Set initial ϵ-greedy policy π, $t = 0$, initial state $s = s_0$
2: Take $a_t \sim \pi(s_t)$ // Sample action from policy
3: Observe (r_t, s_{t+1})
4: **loop**
5: Take action $a_{t+1} \sim \pi(s_{t+1})$
6: Observe (r_t, s_{t+2})
7: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
8: $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
9: $t = t + 1$
10: **end loop**

- Initialize $\epsilon = 1/k, k = 1$, and $\alpha = 0.5$, $Q(-, a) = [1\ 0\ 0\ 0\ 0\ 0\ +10]$
- $Q(-, a_2) = [0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- Assume using same trajectory as example above

Worked Example: SARSA for Mars Rover

Set initial ϵ-greedy policy π, $t = 0$, initial state $s = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_t, s_{t+2})
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$
end loop

- Initialize $\epsilon = 1/k, k = 1$, and $\alpha = 0.5$, $Q(-, a) = [1\ 0\ 0\ 0\ 0\ 0\ +10]$
- $Q(-, a_2) = [0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- $Q(s_4, a_2) = 5.0 + 5 \cdot (0 + Q(s_2, a_1)) = 2.5$

Worked Example: ϵ-greedy Q-Learning Mars

1: Initialize $Q(s,a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}, t = 0$, initial state $s_t = s_0$
2: Set π_b to be ϵ-greedy w.r.t. Q
3: **loop**
4: Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
5: Observe (r_t, s_{t+1})
6: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t))$
7: $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
8: $t = t+1$
9: **end loop**

- Initialize $\epsilon = 1/k, k = 1$, and $\alpha = 0.5$, $Q(-,a) = [1\ 0\ 0\ 0\ 0\ 0\ +10]$
- $Q(-,a_2) = [0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- Like in SARSA example, start in s_6 and take a_1

Worked Example: ϵ-greedy Q-Learning Mars

1: Initialize $Q(s,a) \leftarrow 0, \forall s \in \mathcal{S}, a \in \mathcal{A}, t = 0$, initial state $s_t = s_0$
2: Set π_b to be ϵ-greedy w.r.t. Q
3: **loop**
4: Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
5: Observe (r_t, s_{t+1})
6: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t))$
7: $\pi(s_t) = \arg\max_a Q(s_t,a)$ w.prob $1-\epsilon$, else random
8: $t = t+1$
9: **end loop**

- Initialize $\epsilon = 1/k, k = 1$, and $\alpha = 0.5$, $Q(-,a) = [1\ 0\ 0\ 0\ 0\ 0\ +10]$
- $Q(-,a_2) = [0\ 0\ 0\ 0\ 0\ 0\ +5]$, $\gamma = 1$
- Like in SARSA example, start in s_4 and a_2: $L_t(s_4,a_2) = 0 - 5 \cdot 0.1 = -0.5$
- Recall that in the SARSA update we saw $Q(s_4,a_2) = 2.5$ because we used the actual action taken
- Does how Q is initialized matter (initially asymptotically)?

(Optional) Think pair wise: SARSA and Q-Learning

- SARSA: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
- Q-Learning: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

Select all that are true

- Both SARSA and Q-learning may update their policy after every step
- If $\epsilon = 0$ for all time steps, and Q is initialized randomly, a SARSA Q state update will be the same as a Q-learning Q state update
- Not sure

(Optional) Think pair wise: SARSA and Q-Learning Solutions

- SARSA: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
- Q-Learning: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

Select all that are true

- Both SARSA and Q-learning may update their policy after every step
- If $\epsilon = 0$ for all time steps, and Q is initialized randomly, a SARSA Q state update will be the same as a Q-learning Q state update
- Not sure