

Lecture 3: Model-Free Policy Evaluation: Policy Evaluation Without Knowing How the World Works

Emma Brunskill

CS234 Reinforcement Learning

Winter 2026

- Material builds on structure from David Silver's Lecture 4: Model-Free Prediction. Other resources: Sutton and Barto Jan 1 2018 draft Chapter/Sections: 5.1; 5.5; 6.1-6.3

- In a tabular MDP asymptotically value iteration will always yield a policy with the same value as the policy returned by policy iteration
 - ① True.
 - ② False
 - ③ Not sure
- 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).
 - ① True.
 - ② False
 - ③ Not sure

L3N1 Refresh Your Knowledge

- In a tabular MDP asymptotically value iteration will always yield a policy with the same value as the policy returned by policy iteration
- 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).

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**
- Next Time:
 - 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 environment¹ (direct experience)
- Why is this important?
- What properties do we want from policy evaluation algorithms?

¹Assume today this experience comes from executing the policy π . Later will consider how to do policy evaluation using data gathered from other policies.

This Lecture: Policy Evaluation

- Estimating the expected return of a particular policy if don't have access to true MDP models
- Monte Carlo policy evaluation
 - Policy evaluation when don't have a model of how the world works
 - Given on-policy samples
- Temporal Difference (TD)
- Certainty Equivalence with dynamic programming
- Batch policy evaluation

Recall

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

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

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

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

Recall: Dynamic Programming for Policy Evaluation

- In a Markov decision process

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

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

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

- **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 $\mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s]$.
- This substitution is an instance of **bootstrapping**

This Lecture: Policy Evaluation

- Estimating the expected return of a particular policy if don't have access to true MDP models
- **Monte Carlo policy evaluation**
 - Policy evaluation when don't have a model of how the world work
 - Given on-policy samples
 - Temporal Difference (TD)
 - Certainty Equivalence with dynamic programming
 - Batch policy evaluation

Monte Carlo (MC) Policy Evaluation

- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots + \gamma^{T_i-t} r_{T_i}$ 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
- Simple idea: Value = mean return
- If trajectories are all finite, sample set of trajectories & average returns
- Note: all trajectories may not be same length (e.g. consider MDP with terminal states)

Monte Carlo (MC) 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 (MC) On Policy Evaluation

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

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-1} r_{i,T_i}$ as return from time step t onwards in i th 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 (MC) On Policy Evaluation

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

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-1} r_{i,T_i}$ as return from time step t onwards in i th 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 $V^\pi(s) = G(s)/N(s)$

Optional Worked Example: MC On Policy Evaluation

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

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-1} r_{i,T_i}$
- 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 Monte Carlo (MC) On Policy Evaluation

After each episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots$

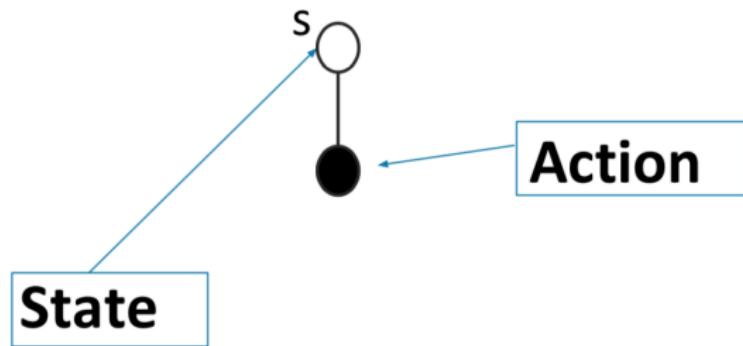
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots$ as return from time step t onwards in i th episode
- For state s visited at time step t in episode i
 - Increment counter of total visits: $N(s) = N(s) + 1$
 - Update estimate

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

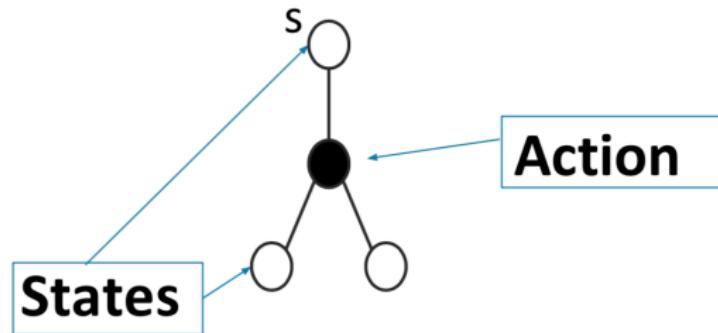
Incremental Monte Carlo (MC) On Policy Evaluation

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}, a_{i,T_i}, r_{i,T_i}$
- $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-1} r_{i,T_i}$
- 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

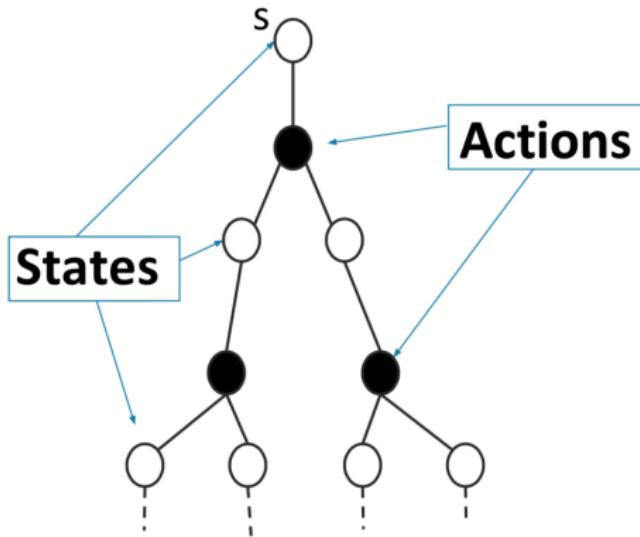
Policy Evaluation Diagram



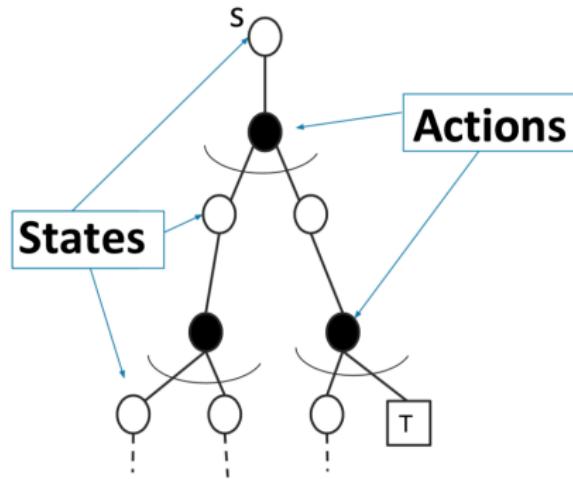
Policy Evaluation Diagram



Policy Evaluation Diagram



Policy Evaluation Diagram

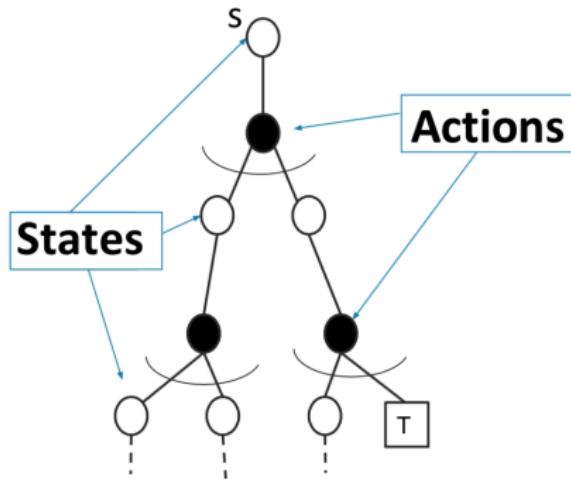


\curvearrowright = Expectation

T = Terminal state

MC Policy Evaluation

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



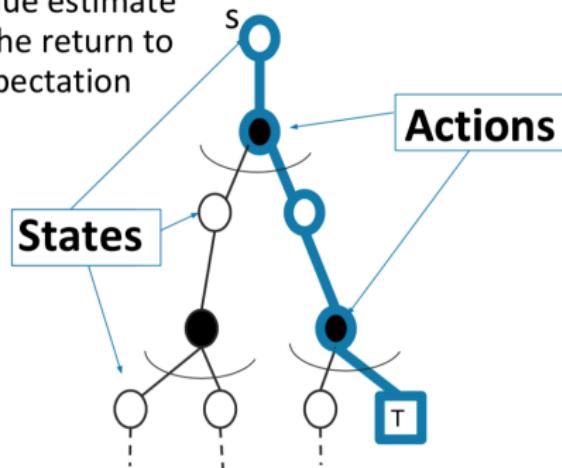
= Expectation

T = Terminal state

MC 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



\smile = Expectation

T = Terminal state

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 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 Approach: 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 Approach: Consistent Estimator

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

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

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

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

- 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 $N(s) \rightarrow \infty$, $V^\pi(s) \rightarrow \mathbb{E}_\pi[G_t|s_t = s]$
- 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

Properties of Monte Carlo On Policy Evaluators

- Update is: $V^\pi(s_{it}) = V^\pi(s_{it}) + \alpha_k(s_j)(G_{i,t} - V^\pi(s_{it}))$
- where we have allowed α to vary (let k be the total number of updates done so far, for state $s_{it} = s_j$)
- If

$$\sum_{n=1}^{\infty} \alpha_n(s_j) = \infty,$$

$$\sum_{n=1}^{\infty} \alpha_n^2(s_j) < \infty$$

- then incremental MC estimate will converge to true policy value $V^\pi(s_j)$

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
 - $s_1, a_1, r_1, s_2, a_2, r_2, \dots$ where the actions are sampled from π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$ 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 don't have access to true MDP models
- Monte Carlo policy evaluation
- **Temporal Difference (TD)**
- Certainty Equivalence with dynamic programming
- Batch policy evaluation

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, a, r, s') tuple

Temporal Difference Learning for Estimating V

- Aim: estimate $V^\pi(s)$ given episodes generated under policy π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$ in MDP M under policy π
- $V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$
- Recall Bellman operator (if know MDP models)

$$B^\pi V(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s))V(s')$$

- In incremental every-visit MC, update estimate using 1 sample of return (for the current i th episode)

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

- **Idea:** have an estimate of V^π , use to estimate expected return

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

Temporal Difference [$TD(0)$] Learning

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

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

- TD(0) error:
$$\delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$
- Can immediately update value estimate after (s, a, r, s') 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})
- $V^\pi(s_t) = V^\pi(s_t) + \underbrace{\alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s_t))}_{\text{TD target}}$

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 = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1, \text{terminal})$

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? $[1 \gamma \gamma^2 0 0 0 0]$

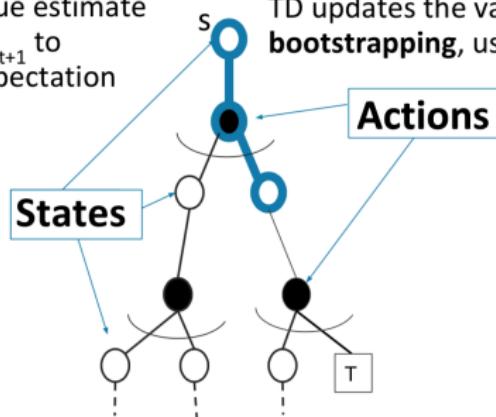
Temporal Difference (TD) Policy Evaluation

$$V^\pi(s_t) = r(s_t, \pi(s_t)) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, \pi(s_t)) V^\pi(s_{t+1})$$

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

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 $V(s_{t+1})$



\smile = Expectation

\square = Terminal state

Check Your Understanding L3N2: Polleverywhere Poll 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})
- $$V^\pi(s_t) = V^\pi(s_t) + \underbrace{\alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s_t))}_{\text{TD target}}$$

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

Check Your Understanding L3N2: Polleverywhere Poll 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})
- $V^\pi(s_t) = V^\pi(s_t) + \underbrace{\alpha([r_t + \gamma V^\pi(s_{t+1})] - V^\pi(s_t))}_{\text{TD target}}$

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, a, r, s') 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**

This Lecture: Policy Evaluation

- Estimating the expected return of a particular policy if don't have access to true MDP models
- Monte Carlo policy evaluation
- Temporal Difference (TD)
- **Certainty Equivalence with dynamic programming**
- Batch policy evaluation

Certainty Equivalence V^π MLE MDP Model Estimates

- Model-based option for policy evaluation without true models
- After each (s_i, a_i, r_i, s_{i+1}) tuple
 - Recompute maximum likelihood MDP model for (s, a)

$$\hat{P}(s'|s, a) = \frac{1}{N(s, a)} \sum_{k=1}^i \mathbb{1}(s_k = s, a_k = a, s_{k+1} = s')$$

$$\hat{r}(s, a) = \frac{1}{N(s, a)} \sum_{k=1}^i \mathbb{1}(s_k = s, a_k = a) r_k$$

- Compute V^π using MLE MDP ² (using any dynamic programming method from lecture 2))
- Optional worked example at end of slides for Mars rover domain.

²Requires initializing for all (s, a) pairs

Certainty Equivalence V^π MLE MDP Model Estimates

- Model-based option for policy evaluation without true models
- After each (s, a, r, s') tuple
 - Recompute maximum likelihood MDP model for (s, a)

$$\hat{P}(s' | s, a) = \frac{1}{N(s, a)} \sum_{k=1}^K \sum_{t=1}^{T_k-1} \mathbf{1}(s_{k,t} = s, a_{k,t} = a, s_{k,t+1} = s')$$

$$\hat{r}(s, a) = \frac{1}{N(s, a)} \sum_{k=1}^K \sum_{t=1}^{T_k-1} \mathbf{1}(s_{k,t} = s, a_{k,t} = a) r_{t,k}$$

- Compute V^π using MLE MDP
- Cost: Updating MLE model and MDP planning at each update ($O(|S|^3)$ for analytic matrix solution, $O(|S|^2|A|)$ for iterative methods)
- Very data efficient and very computationally expensive
- Consistent (will converge to right estimate for Markov models)
- Can also easily be used for off-policy evaluation (which we will shortly define and discuss)

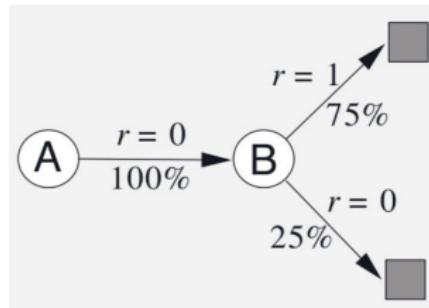
This Lecture: Policy Evaluation

- Estimating the expected return of a particular policy if don't have access to true MDP models
- Monte Carlo policy evaluation
 - Policy evaluation when don't have a model of how the world work
 - Given on-policy samples
- Temporal Difference (TD)
- Certainty Equivalence with dynamic programming
- **Batch policy evaluation**

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: (Ex. 6.4, Sutton & Barto, 2018)



- Two states A, B with $\gamma = 1$
- Given 8 episodes of experience:
 - $A, 0, B, 0$
 - $B, 1$ (observed 6 times)
 - $B, 0$
- Imagine running TD updates over data infinite number of times
- $V(B) =$

AB Example: (Ex. 6.4, Sutton & Barto, 2018)

- TD Update: $V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}}) - V^\pi(s_t))$
- Two states A, B with $\gamma = 1$
- Given 8 episodes of experience:
 - $A, 0, B, 0$
 - $B, 1$ (observed 6 times)
 - $B, 0$
- Imagine run TD updates over data infinite number of times
- $V(B) = 0.75$ by TD or MC
- What about $V(A)$?

Check Your Understanding L3N3: AB Example: (Ex. 6.4, Sutton & Barto, 2018)

- TD Update: $V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}}) - V^\pi(s_t))$
- Two states A, B with $\gamma = 1$
- Given 8 episodes of experience:
 - $A, 0, B, 0$
 - $B, 1$ (observed 6 times)
 - $B, 0$
- Imagine run TD updates over data infinite number of times
- $V(B) = 0.75$ by TD or MC
- What about $V(A)$?
- Respond in Poll

Check Your Understanding L3N3: AB Example: (Ex. 6.4, Sutton & Barto, 2018)

- TD Update: $V^\pi(s_t) = V^\pi(s_t) + \underbrace{\alpha([r_t + \gamma V^\pi(s_{t+1})])}_{\text{TD target}} - V^\pi(s_t)$
- Two states A, B with $\gamma = 1$
- Given 8 episodes of experience:
 - $A, 0, B, 0$
 - $B, 1$ (observed 6 times)
 - $B, 0$
- Imagine run TD updates over data infinite number of times
- $V(B) = 0.75$ by TD or MC
- What about $V(A)$?

Batch MC and TD: Convergence

Batch MC and TD: Convergence

Summary: Policy Evaluation

Estimating the expected return of a particular policy if don't have access to true MDP models. Ex. evaluating average purchases per session of new product recommendation system

- Monte Carlo policy evaluation
 - Policy evaluation when we don't have a model of how the world works
 - Given on policy samples
 - Given off policy samples
- Temporal Difference (TD)
- Dynamic Programming with certainty equivalence
- *Understand what MC vs TD methods compute in batch evaluations
- Metrics / Qualities to evaluate and compare algorithms
 - Uses Markov assumption
 - Accuracy / MSE / bias / variance
 - Data efficiency
 - Computational efficiency

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
- Next Time:
 - Control when don't have a model of how the world works