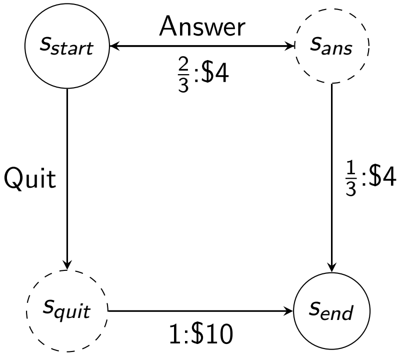
CS-541: Artificial Intelligence
Lecture 7

Abdul Rafae Khan
Department of Computer Science
Stevens Institute of Technology
akhan4@stevens.edu

March 28, 2022

Recap

States: s_{start} & s_{end}
Chance nodes: s_{ans} & s_{quit}
Policy (π) produces a path:
$s_0; a_1 r_1 s_1; a_2, r_2, s_2; a_3, r_3, s_3; \cdots$
Utility: $u_\pi = r_1 + \gamma r_2 + \gamma^2 r_3$
Value: $V_\pi(s)$, Expected Utility for policy π starting at state s
Q-Value: $A_\pi(s, a)$, Expected Utility for policy π after taking action a from state s



Recap

s_{start}: start state
Actions(s**)**: all possible actions from state s
T(s, a, s'**)**: probability of s' if action a is taken from state s
Reward(s, a, s'**)**: reward from the transition s to s'
IsEnd(s**)**: is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Unknown Transitions & Reward

s_{start}: start state
Actions(s): all possible actions from state s
~~**T(s, a, s')**: probability of s' if action a is taken from state s~~
~~**Reward(s, a, s')**: reward from the transition s to s'~~
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Unknown Transitions & Reward

s_{start}: start state
Actions(s): all possible actions from state s
~~**T(s, a, s')**: probability of s' if action a is taken from state s~~
~~**Reward(s, a, s')**: reward from the transition s to s'~~
IsEnd(s): is s a goal state
$0 \leq \gamma \leq 1$: discount factor (default: 1)

Reinforcement Learning!

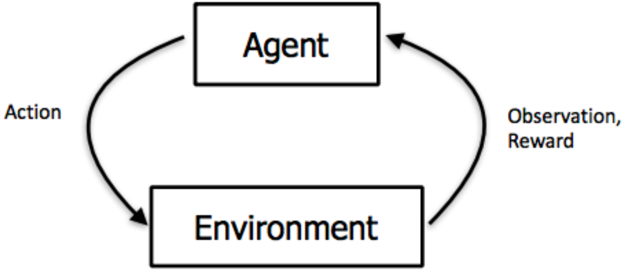
Unknown Transitions & Reward

MDPs:
Know how the word works: Environment is observable
Find a policy which maximizes the reward

Reinforcement learning:
Do not know about the world: Environment is not observable
Find a policy which maximizes the reward
Perform actions and collect the reward

Reinforcement Learning

The agent performs actions and observes the rewards
This feedback loop helps learn the missing values (transition probabilities and reward)



Reinforcement Learning

Overall algorithm

```
for t = 1, 2, 3, ···
    Choose action a_t = π_act(s_{t−1})
    Get reward r_t and new state s_t
    Update parameters
```

Model-based Monte Carlo

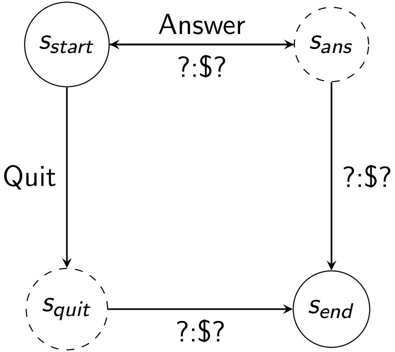
Data: $s_0; a_1 r_1 s_1; a_2, r_2, s_2; a_3, r_3, s_3; \cdots$
Estimate $T(s, a, s')$ & $R(s, a, s')$

$$\hat{T}(s, a, s') = \frac{\text{No. of times } s, a, s' \text{ occurs}}{\text{No. of times } s, a \text{ occurs}}$$

$$\hat{R}(s, a, s') = \text{reward observed by } s, a, s'$$

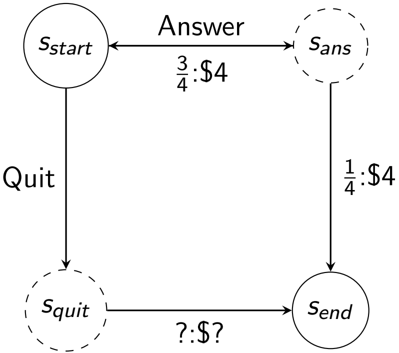
Model-based Monte Carlo

Iteration: 0



Model-based Monte Carlo

Policy π is Answer
Iteration: 1
Data:s_{start}; $Ans, 4, s_{start}$; $Ans, 4, s_{start}$; $Ans, 4, s_{start}$; $Ans, 4, s_{end}$

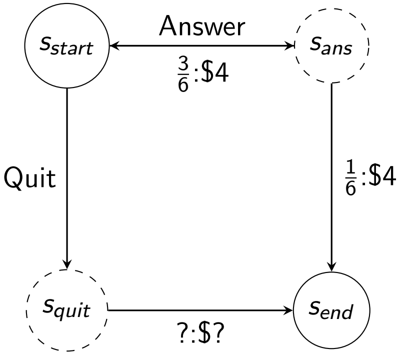


Model-based Monte Carlo

Policy π is Answer

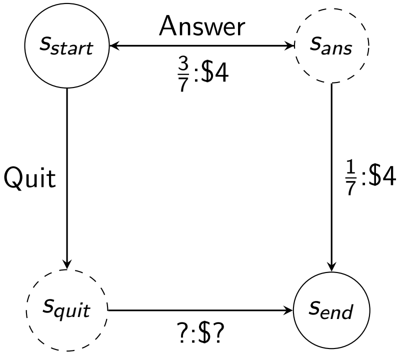
Iteration: 2

Data:s_{start}; $Ans, 4, s_{start}$; $Ans, 4, s_{end}$

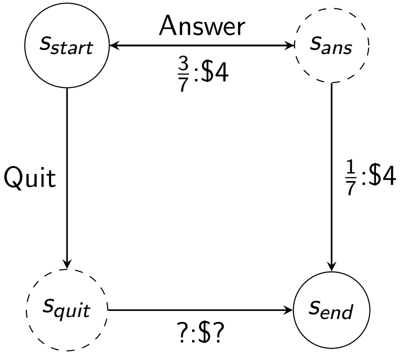


Model-based Monte Carlo

Policy π is Answer
Iteration: 3
Data: $s_{start}; Ans, 4, s_{end}$



Model-based Monte Carlo

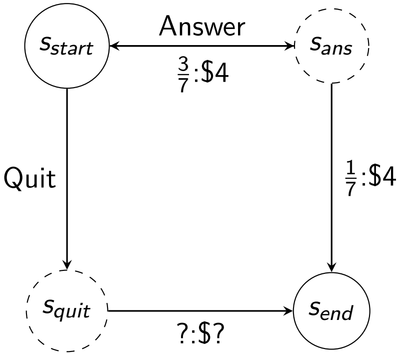


Can converge to true values

Compute policy using value Iteration for the estimated MDP (with $\hat{T}$ and $\hat{R}$)

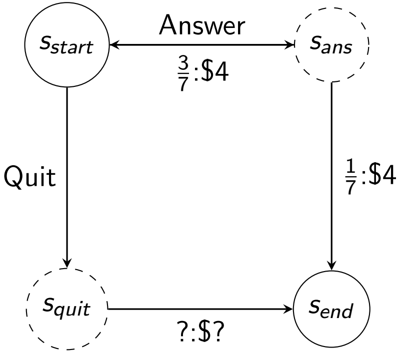
Model-based Monte Carlo

If $a \neq \pi(s)$ ($a = Quit$), s, a will not be seen



Model-based Monte Carlo

Exploration: try unknown actions to get information



Model-based Monte Carlo

We can use the computed transitions and rewards
And compute the optimal Value and Q-value

$$\hat{V}_{opt}(s) = E[\hat{V}_{opt}(s)] = \begin{cases} 0 \text{ if } isEnd(s) \\ \hat{Q}_{opt}(s) \text{ otherwise} \end{cases}$$

$$\hat{Q}_{opt}(s, a) = \sum_{s'} \hat{T}(s, a, s')[\hat{R}(s, a, s') + \gamma \hat{V}_{opt}(s')]$$

Model-based Monte Carlo

Pros:

- Makes efficient use of experiences

Cons:

- May not scale to large state spaces
  - Learns model one state-action pair at a time
  - Cannot solve MDP for very large $|S|$

Model-based vs Model-free

Goal: Compute the age of CS students

$P(A)$ is known

$$\begin{aligned} \mathbb{E}[A] &= \sum_a P(A) \cdot a \\ &= 0.35 \times 20 + \cdots \end{aligned}$$

Model-based vs Model-free

Without $P(A)$, collect samples $[a_1, a_2, \cdots, a_N]$

Unknown $P(A)$: *Model-based*

$$\hat{P}(A) = \frac{num(a)}{N}$$

$$\mathbb{E}[A] \approx \sum_a \hat{P}(A)$$

Because, eventually the correct model is learnt

Unknown $P(A)$: *Model-free*

$$\mathbb{E}[A] \approx \frac{1}{N} \sum_i a_i$$

Because, samples appear with right frequencies

Model-based vs Model-free

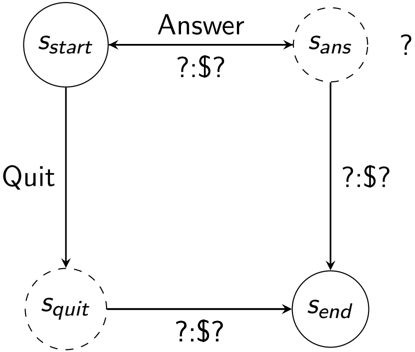
Model based vs. Model free:
Do we estimate $T(s, a, s')$ and $R(s, a, s')$, or just learn values/policy directly

Online vs Batch:
Learn while exploring the world, or learn from fixed batch of data

Active vs Passive:
Does the learner actively choose actions to gather experience? or, is a fixed policy provided?

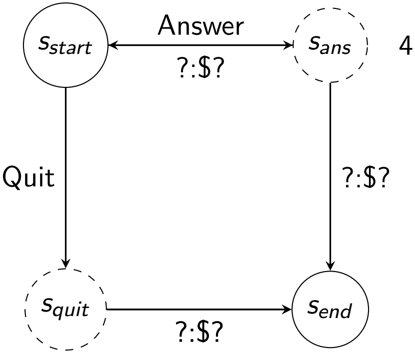
Model-free Monte Carlo

Policy π is Answer
Iteration: 0
Data:



Model-free Monte Carlo

Policy π is Answer
Iteration: 1
Data: $s_{start}; Ans, 4, s_{end}$

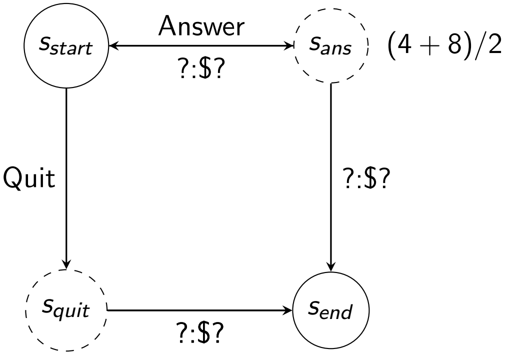


Model-free Value Iteration

Policy π is Answer

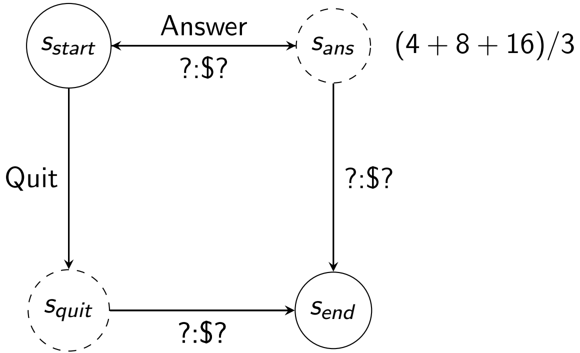
Iteration: 2

Data: $s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$



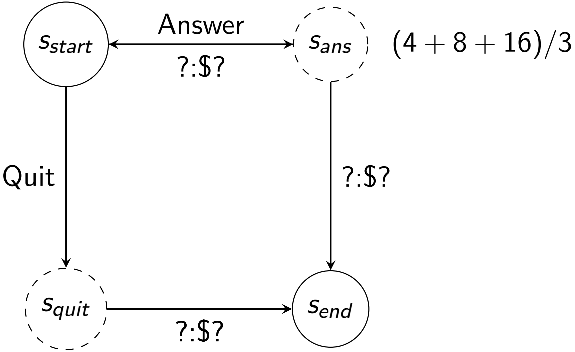
Model-free Value Iteration

Policy π is Answer
Iteration: 3
Data: $s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$



Model-free Value Iteration

We are estimating Q_π and not Q_{opt}



Model-free Value Iteration

Policy π is Answer
Data: $s_1; a_1, r_1, s_1; a_2, r_2, s_2; \cdots ; a_n, r_n, s_n$

$$\hat{Q}(s, a) = \text{average of } u_t \text{ where } s_{t-1} = s, a_t = a$$

Equivalent formulation (convex combination)

$$\text{for each } (s, a, u)$$

$$\eta = \frac{1}{1 + \text{No. of updates } (s, a)}$$

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta u$$

Model-free Value Iteration

Convex combination:

$$\begin{aligned}&\text{for each } (s, a, u)\\&\qquad \hat{Q}_\pi(s, a) \leftarrow (1-\eta)\hat{Q}_\pi(s, a) + \eta u\end{aligned}$$

Stochastic Gradient:

$$\begin{aligned}&\text{for each } (s, a, u)\\&\qquad \hat{Q}_\pi(s, a) \leftarrow \hat{Q}_\pi(s, a) - \eta[\underbrace{\hat{Q}_\pi(s, a)}_{\text{prediction}} - \underbrace{u}_{\text{target}}]\end{aligned}$$

Objective (Least squares): $(\hat{Q}_\pi(s, a) - u)^2$

Using the Utility

Policy π is Answer
Data:

$$s_{start}; Ans, 4, s_{end} \qquad u = 4$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 8$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 12$$

$$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end} \qquad u = 16$$

Model-free Monte Carlo:

$$\text{for each } (s, a, u)$$

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta \underbrace{u}_{data}]$$

Using the reward+Q-value

Current estimate: $Q_\pi(s, Ans) = 11$

Data:

$s_{start}; Ans, 4, s_{end}$	$4 + 0$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$
$s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{start}; Ans, 4, s_{end}$	$4 + 11$

SARSA:

for each (s, a, r, s', a')

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta[\underbrace{r}_{data} + \gamma \underbrace{\hat{Q}_\pi(s', a')}_{estimate}]$$

Model-free Monte Carlo vs SARSA

Model-free Monte Carlo:

$$\text{for each } (s, a, u)$$
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta \underbrace{u}_{data}]$$

SARSA:

$$\text{for each } (s, a, r, s', a')$$
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta[\underbrace{r}_{data} + \gamma \underbrace{\hat{Q}_\pi(s',a')}_{estimate}]$$

SARSA uses $\hat{Q}_\pi(s,a)$ instead of raw data u
SARSA doesn't have to wait till it reaches the terminal node to update

Model-free Monte Carlo vs SARSA

Output	MDP	Reinforcement Learning
Q_π	Policy Evaluation	Model-free Monte Carlo, SARSA
Q_{opt}	Value Iteration	*Q-Learning*

Table: Caption

Q-Learning

Recall (Bellman optimality equation):

$$Q_{opt}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_{opt}(s')]$$

Q-Learning:

$$\text{for each } (s,a,r,s')$$
$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{opt}(s,a)}_{prediction} + \eta\underbrace{(r + \gamma V_{opt}(s'))}_{target}$$

Q-Learning

Recall (Bellman optimality equation):

$$Q_{opt}(s, a) = \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V_{opt}(s')]$$

Q-Learning:

$$\text{for each } (s, a, r, s')$$

$$\hat{Q}_{opt}(s, a) \leftarrow (1 - \eta)\, \underbrace{\hat{Q}_{opt}(s, a)}_{prediction} + \eta(\underbrace{r + \gamma V_{opt}(s')}_{target})$$

$$\hat{V}_{opt}(s') = \max_{a' \in Actions(s')} \hat{Q}_{opt}(s', a')$$

SARSA vs Q-Learning

SARSA:

$$\text{for each } (s, a, r, s', a')$$
$$\hat{Q}_\pi(s,a) \leftarrow (1-\eta)\hat{Q}_\pi(s,a) + \eta\big[r + \gamma \hat{Q}_\pi(s',a')\big]$$

Q-Learning:

$$\text{for each } (s, a, r, s')$$
$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\hat{Q}_{opt}(s,a) + \eta\big(r + \gamma \max_{a' \in Actions(s')} \hat{Q}_{opt}(s',a')\big)$$

Reinforcement Learning

On-policy: evaluate or improve the data-generating policy
Off-policy: evaluate or learn using data from another policy

	On-Policy	**Off-Policy**
Policy Evaluation (Q_π)	Monte-Carlo, SARSA	
Policy Optimization (Q_{opt})		Q-Learning

Reinforcement Learning

Algorithm	Estimating	Based On
Model-Based Monte Carlo	$\hat{T}, \hat{R}$	$s_0, a_1, r_1, s_1, \cdots$
Model-Free Monte Carlo	$\hat{Q}_\pi$	u
SARSA	$\hat{Q}_\pi$	$r + \hat{Q}_\pi$
Q-Learning	$\hat{Q}_{opt}$	$r + \hat{Q}_{opt}$

Reinforcement Learning

Overall algorithm

for $t = 1, 2, 3, \cdots$

Choose action $a_t = \pi_{act}(s_{t-1})$

Get reward r_t and new state s_t

Update parameters

Reinforcement Learning

Overall algorithm

```
for t = 1, 2, 3, ...
    Choose action a_t = π_act(s_{t-1}) (how?)
    Get reward r_t and new state s_t
    Update parameters (how?)
```

$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3, \cdots ; a_n, r_n, s_n$
What policy π_{act} should be used?

Choosing the policy

Option1: Select the best policy
$\pi_{act}(s) = \arg\max_{a \in Actions(s)} \hat{Q}_\pi(s, a)$
Problem: $\hat{Q}_\pi(s, a)$ estimates are inaccurate. Too greedy

Option2: Select a random policy
$\pi_{act}(s) =$ random from $Actions(s)$
Problem: Exploration is not guided

Epsilon-Greedy Policy

$$\pi_{act}(s) = \begin{cases} \arg\max_{a \in Actions(s)} \hat{Q}_\pi(s, a) & \text{probability } 1 - \epsilon \\ \text{random from } Actions(s) & \text{probability } \epsilon \end{cases}$$

A balance between the two!

Function Approximation

Stochastic Gradient update:

$$\hat{Q}_{opt}(s, a) \leftarrow (1 - \eta)\hat{Q}_{opt}(s, a) + \eta\big[\underbrace{\hat{Q}_{opt}(s, a)}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s', a'))}_{target}\big]$$

How to generalize to unseen states/actions

Function Approximation

Linear Regression:

Use features $\phi(s, a)$ and weights $\mathbf{w}$

$$\hat{Q}_{opt}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$

Grid World:

$$\phi_1(s, a) = 1[a = Up]$$
$$\phi_2(s, a) = 1[a = Left]$$
$$\ldots$$

$$\phi_7(s, a) = 1[s = (1, *)]$$
$$\phi_8(s, a) = 1[s = (*, 2)]$$
$$\ldots$$

Function Approximation

Q-Learning with Function Approximation:

$$\text{for each } (s, a, r, s') :$$
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \big[\underbrace{\hat{Q}_{opt}(s, a; \mathbf{w})}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s'))}_{target}\big]\phi(s, a)$$

Objective Function:

$$\Big(\underbrace{\hat{Q}_{opt}(s, a; \mathbf{w})}_{prediction} - \underbrace{(r + \gamma \hat{V}_{opt}(s'))}_{target}\Big)^2$$

Recap

Reinforcement Learning
Model-based Monte Carlo Learning
Model-free Monte Carlo Learning
SARSA
Q-Learning
Epsilon-Greedy
Function Approximation

References

Stuart Russell and Xiaodong Song (2021)
CS 188 — Introduction to Artificial Intelligence
University of California, Berkeley

Chelsea Finn and Nima Anari (2021)
CS221 — Artificial Intelligence: Principles and Techniques
Stanford University

The End