Freie Universität Berlin

Lecture: Machine Learning for Data Science

Winter semester 2021/22

Lecture 18: Reinforcement Learning (Model-free learning)

Prof. Dr. Eirini Ntoutsi

Outline

- Model-free learning
- Direct policy evaluation
- TD-learning
- Q-learning
- Exploration-vs-Exploitation
- Things you should know from this lecture & reading material

Model-free learning

- Model-based learning: learns an approximate model of T, R based on experience and uses this model to solve a (conventional) MDP.
- Model-free learning: learns the v-values of states or q-values of state-action pairs directly, without constructing a model of the rewards and transitions in the MDP

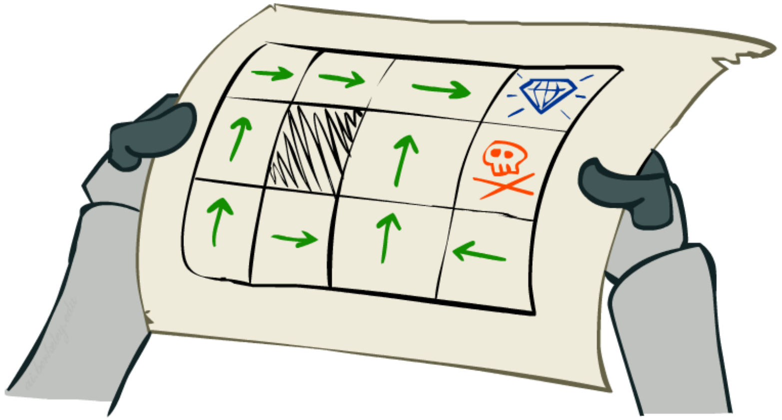
- We will cover
  - Direct evaluation
  - Temporal difference (TD) learning
  - Q-learning

Passive RL: An agent is given a policy to follow and learns the values of the states under that policy as its experience grows

Active RL: The agent can use the feedback it receives to iteratively update its policy

Passive Reinforcement Learning

- Passive RL: An agent is given a policy π to follow and learns the values of the states under that policy as its experience grows
- Direct policy evaluation
  - Input: a fixed policy $\pi(s)$
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - Goal: learn the state values



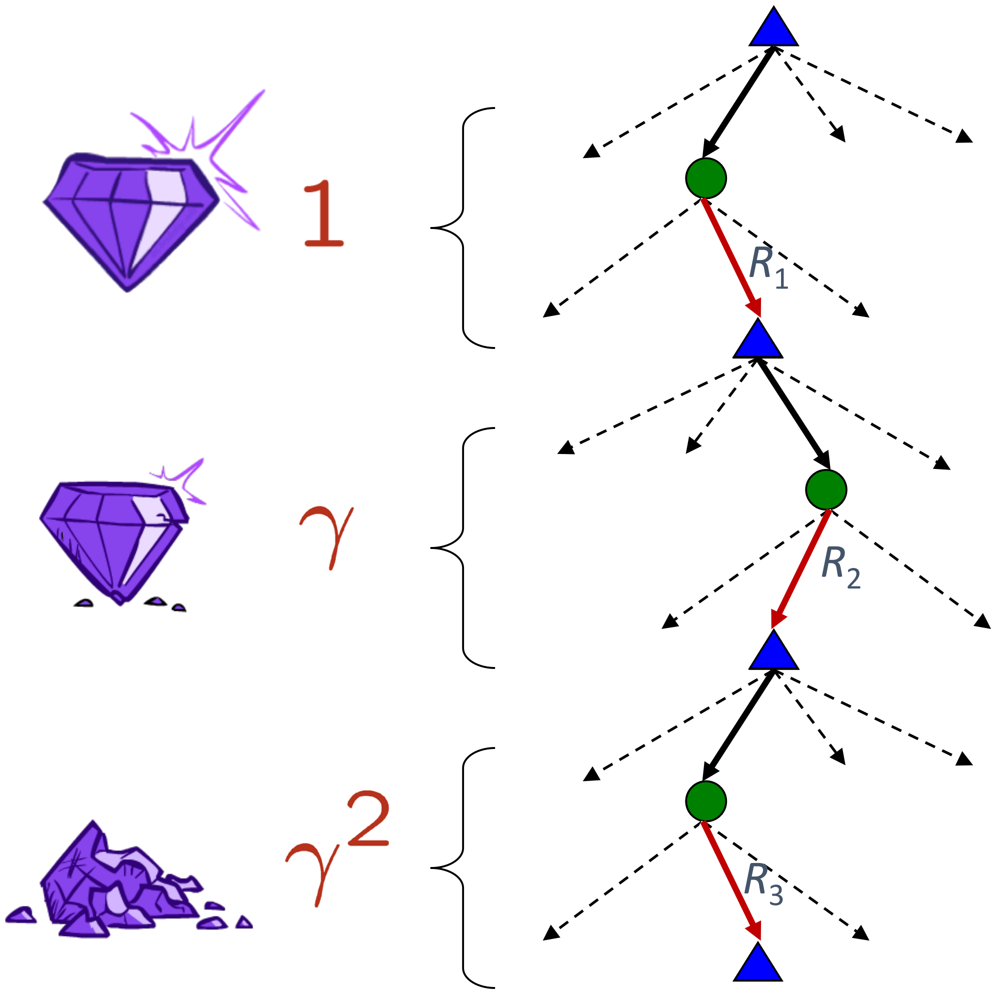
- In this case:
  - No choice about what actions to take
  - Just execute the policy and learn from experience

Outline

- Model-free learning
- Direct policy evaluation
- TD-learning
- Q-learning
- Exploration-vs-Exploitation
- Things you should know from this lecture & reading material

Direct evaluation

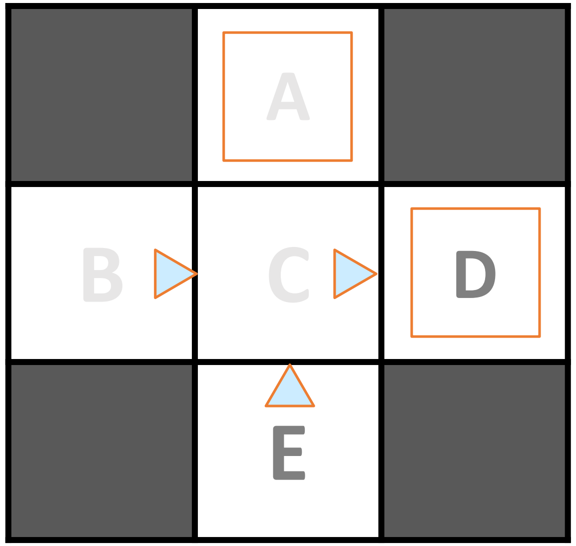
- **Goal**: Compute values for each state under a policy π
- **Idea**: Have the agent learn from experience while following π
- The experience comes in form of episodes
- More concretely:
  - Act according to π
  - Every time you visit a state s, keep track of its utility (sum of discounted rewards) as well as of the number of visits
    - $U[s_o, s_1, \ldots] = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^n R(s_n)$
  - Average those samples to get the estimated value of s
- This is called direct policy evaluation



Example: Direct evaluation

$$U[s_o, s_1, \ldots] = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^n R(s_n)$$

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

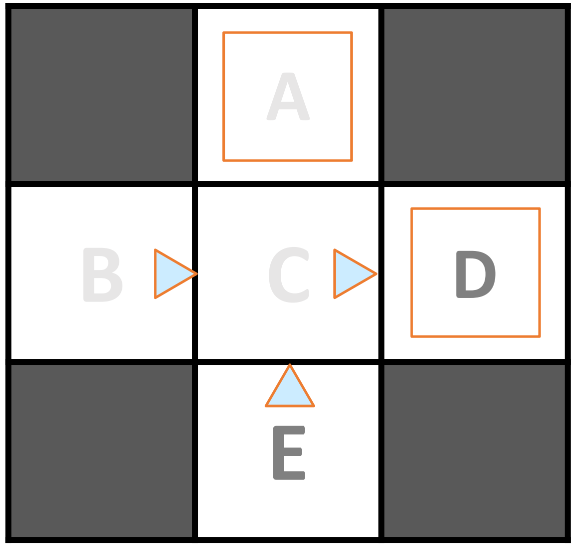
Output Values

Episodes	**1**
A	
B	-1-1+10=8
C	-1+10=9
D	10
E	

Example: Direct evaluation

$$U[s_o, s_1, \ldots] = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^n R(s_n)$$

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

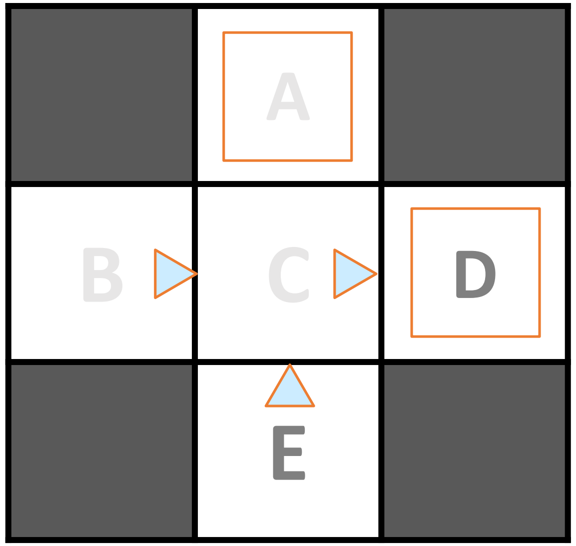
Output Values

Episodes	**1**	**2**
A		
B	8	-1-1+10=8
C	9	-1+10=9
D	10	10
E		

Example: Direct evaluation

$$U[s_o, s_1, \ldots] = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^n R(s_n)$$

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

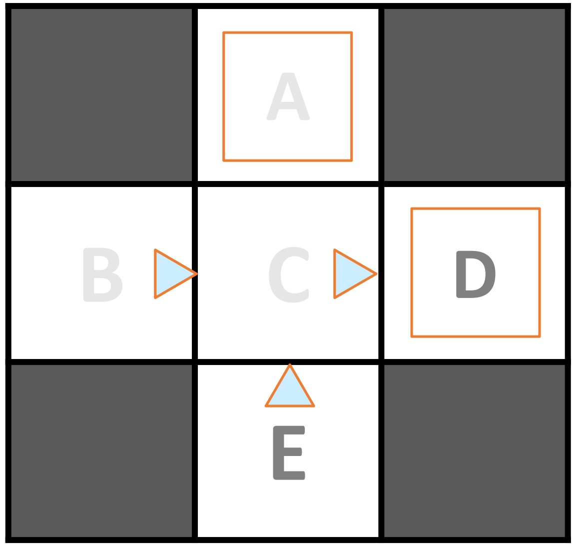
Output Values

Episodes	**1**	**2**	**3**
A			
B	8	8	
C	9	9	9
D	10	10	10
E			8

Example: Direct evaluation

$U[s_o, s_1, \ldots] = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots + \gamma^n R(s_n)$

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values

Episodes	**1**	**2**	**3**	**4**
A				-10
B	8	8		
C	9	9	9	-11
D	10	10	10	
E			8	-12

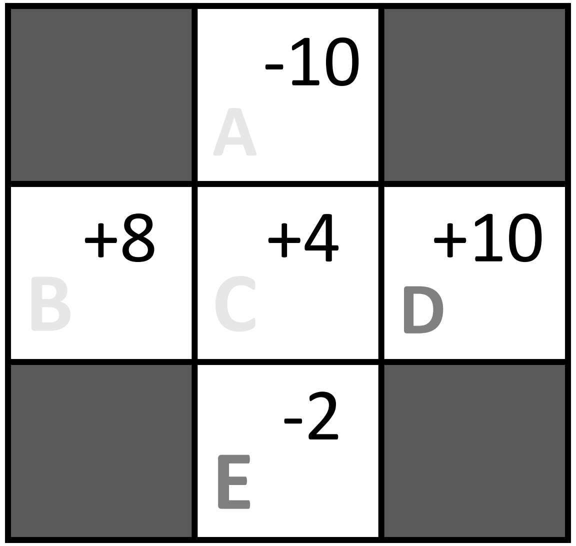
Example: Direct evaluation

- We can now estimate the values for each state s

Episodes	1	2	3	4
A				-10
B	8	8		
C	9	9	9	-11
D	10	10	10	
E			8	-12

s	Total Reward	Times Visited	$V^{\pi}(s)$
A	-10	1	-10
B	16	2	8
C	16	4	4
D	30	3	10
E	-4	2	-2

Output Values

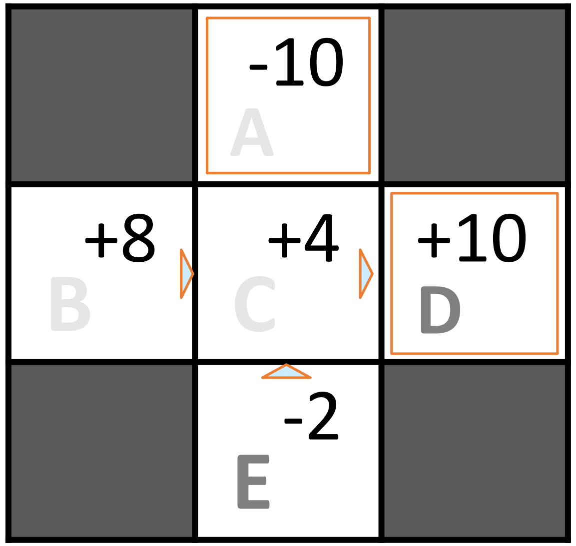


Direct evaluation: Discussion

- Batch approach: It collects data and learns the values at the end of training
- Passive approach: A fixed policy π is provided
- Model-free approach: learn directly the values of the states

- What is good about direct evaluation?
  - Easy to understand
  - Doesn't require any knowledge of T, R
  - It eventually computes the correct average values, using just sample transitions

Output Values

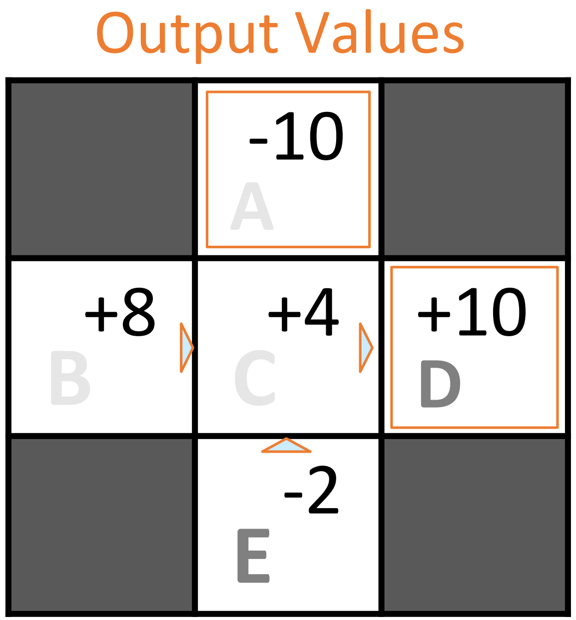


Direct evaluation: Discussion cont'

- What is bad about direct evaluation?
  - In our example: B and E only have C as a successor (based on training) and the transition to C gives -1 reward for both.
  - How can their values be different under π (recall Bellman equation)?

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

  - With enough training, it will eventually converge to the true state values, but it takes a long time to learn
  - Each state must be learned separately
    - No share of information between states, it wastes information about state connections (transitions between states)

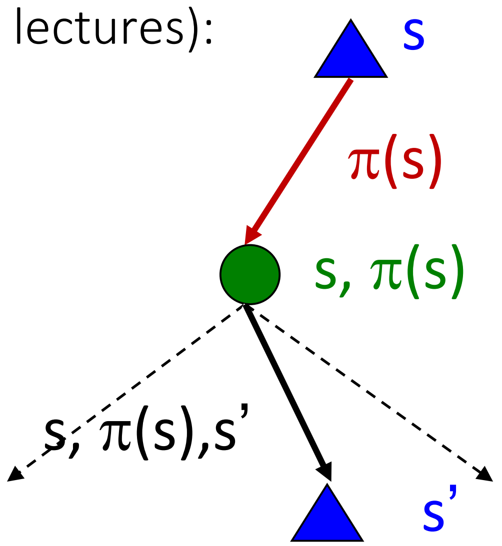


Why not use policy evaluation?

- Simplified Bellman updates calculate v-values for a fixed policy π (see previous lectures):
  - Each round, replace V with a one-step-look-ahead layer over V

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$



  - This approach fully exploited the connections between the states
  - Unfortunately, it requires T and R to do it!

- Key question: how can we do this update to V without knowing T and R?
  - In other words, how to we take a weighted average without knowing the weights?

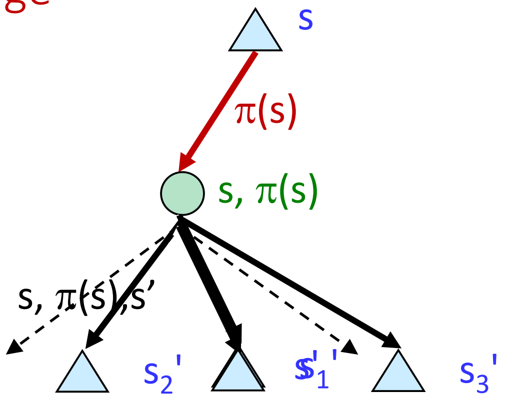
Sample-based policy evaluation?

- We want to improve our estimate of V by computing these averages but we don't have T, R:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

- **Idea:** Take samples of outcomes s' (by doing the action in the real world!) and average

$$sample_1 = R(s,\pi(s),s_1') + \gamma V_k^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_k^{\pi}(s_2')$$
$$\ldots$$
$$sample_n = R(s,\pi(s),s_n') + \gamma V_k^{\pi}(s_n')$$
$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$



Almost! But we can't rewind time to get sample after sample from state s.

  - If we could collect data from a given state s many times this could be a good idea
    - Nice idea but hard to achieve

Detour: sampling expectations via an example

Goal: Compute expected age of the class students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \qquad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

Estimate P(a) from samples and then expected age

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

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

Why does this work? Because eventually you learn the right model.

Unknown P(A): "Model Free"

Directly estimate expected age from samples

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

Why does this work? Because samples appear with the right frequencies.

From direct evaluation to temporal difference learning

- Direct evaluation: collect data (in form of episodes) and learn the values at the end of training
  - No share of information between states
- Bellman update allows for share of information between states however requires T, R knowledge
- Sample-based policy evaluation idea:
  - if we could collect data from a given state s many times (s_1, s_2, …, s_n) we could compute the average state value → not easy
- New idea: what if we learn from each sample as we get it? → temporal difference learning

Outline

- Model-free learning
- Direct policy evaluation
- TD-learning
- Q-learning
- Exploration-vs-Exploitation
- Things you should know from this lecture & reading material

Temporal Difference Learning

- **Main idea:** learn from every experience (sample)
  - Update V(s) each time we get a new experience/sample, i.e., transition (s, a, s', r)
  - Policy still fixed, so a=π(s), so we are still doing evaluation!
  - Likely outcomes s' will contribute updates more often
- How do we update V(s) based on just one sample?
  - **Main idea:** Correct the old estimate *V(s)* with the new sample value
    - Estimate value of *V(s) based on sample: (s, a, s', r)*

$$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

    - Update: Move current value estimate *V(s)* values toward value of whatever successor occurs (*s'*)

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$$

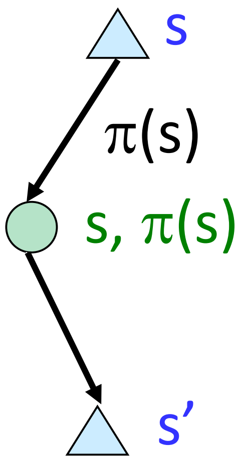
    - The learning rate *α* controls how to combine our current estimate with the new sampled estimate, 0≤*α*≤ 1

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

Same update :

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s, \pi(s), s') + \gamma V^{\pi}(s')\right]$$

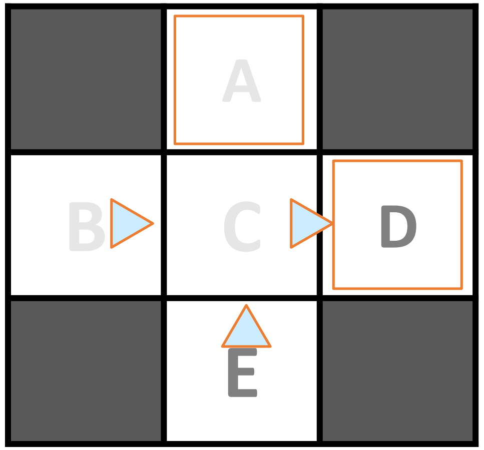
Ma



Example: Temporal Difference Learning

$V^{\pi}(C) = (1-\alpha)V^{\pi}(C)+\alpha[R(C,east,D)+\gamma V^{\pi}(D)]$ $=(1-0.5)*0+0.5[-2+8]=3$ Only C is updated

States

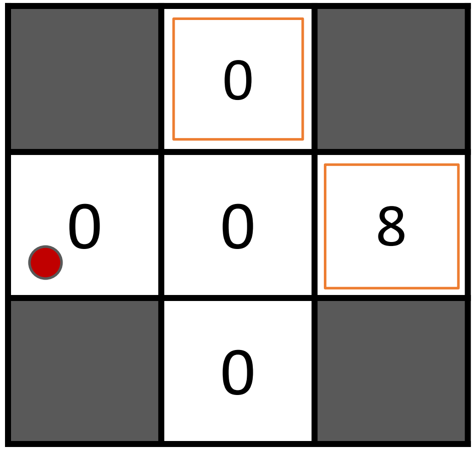


Assume: $\gamma = 1$, $\alpha = 1/2$

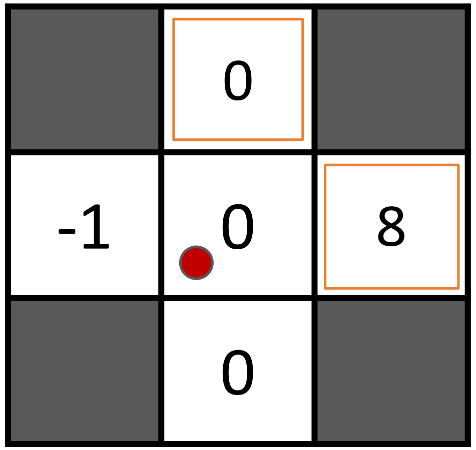
Observed Transitions (samples)

B, east, C, -2

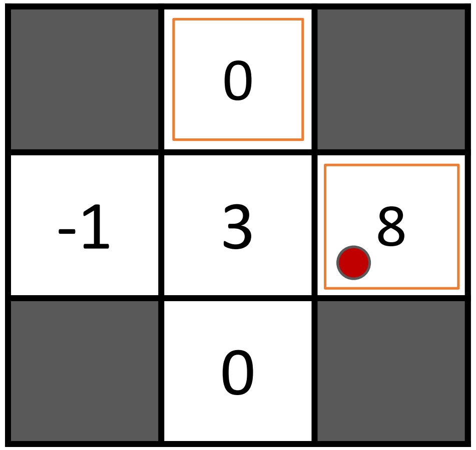
C, east, D, -2



Current estimates



Updated estimates



Updated estimates

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

$V^{\pi}(B) = (1-\alpha)V^{\pi}(B)+\alpha[R(B,east,C)+\gamma V^{\pi}(C)]$ $=(1-0.5)*0+0.5[-2+0]=-1$ Only B is updated

The update rule

$$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

- Update rule $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$
- Assuming k samples/updates of $V^{\pi}(s)$: $sample_1$, $sample_2$,…, $sample_{k-1}$, $sample_k$

$$V_k^{\pi}(s) \leftarrow (1-a)*V_{k-1}^{\pi}(s)+a*sample_k$$
$$V_{k-1}^{\pi}(s) \leftarrow (1-a)*V_{k-2}^{\pi}(s)+a*sample_{k-1}$$

$$V_k^{\pi}(s) \leftarrow (1-a)*[(1-a)*V_{k-2}^{\pi}(s)+a*sample_{k-1}]+a*sample_k$$
$$V_k^{\pi}(s) \leftarrow (1-a)^2*V_{k-2}^{\pi}(s) + (1-a)*a*sample_{k-1} +a*sample_k$$

$$\begin{aligned} V_k^{\pi}(s) &\leftarrow (1-\alpha)^k V_0^{\pi}(s) + \alpha \cdot [(1-\alpha)^{k-1} \cdot sample_1 + \ldots + (1-\alpha) \cdot sample_{k-1} + sample_k] \\ V_k^{\pi}(s) &\leftarrow \alpha \cdot [(1-\alpha)^{k-1} \cdot sample_1 + \ldots + (1-\alpha) \cdot sample_{k-1} + sample_k] \end{aligned}$$

- Since the learning rate a: $0 \leq a \leq 1$, $(1-a)^k \rightarrow 0$ as k increases.
- So older samples contribute exponentially less to $V^{\pi}(s)$, which is great since these samples are based on old (hence worse) versions of $V^{\pi}(s)$

$$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

The update rule

- So, with a simple update rule $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$ we are able to
  - Learn with each experience/sample → online learning approach
  - Give exponentially less weight to older, potentially less accurate sample estimates
  - Converge to learning the true value states much faster comparing to direct policy evaluation

Outline

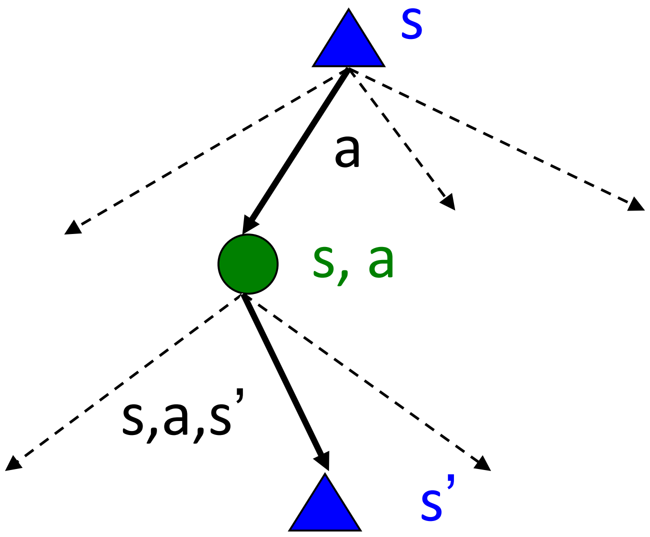
- Model-free learning
- Direct policy evaluation
- TD-learning
- Q-learning
- Exploration-vs-Exploitation
- Things you should know from this lecture & reading material

Problems with TD Value Learning

- TD value learning is a model-free way to do policy evaluation, mimicking Bellman updates with running sample averages
- However, if we want to turn values into a (new) policy, we're sunk (recall previous lectures):

$$\pi(s) = \arg\max_a Q(s,a)$$

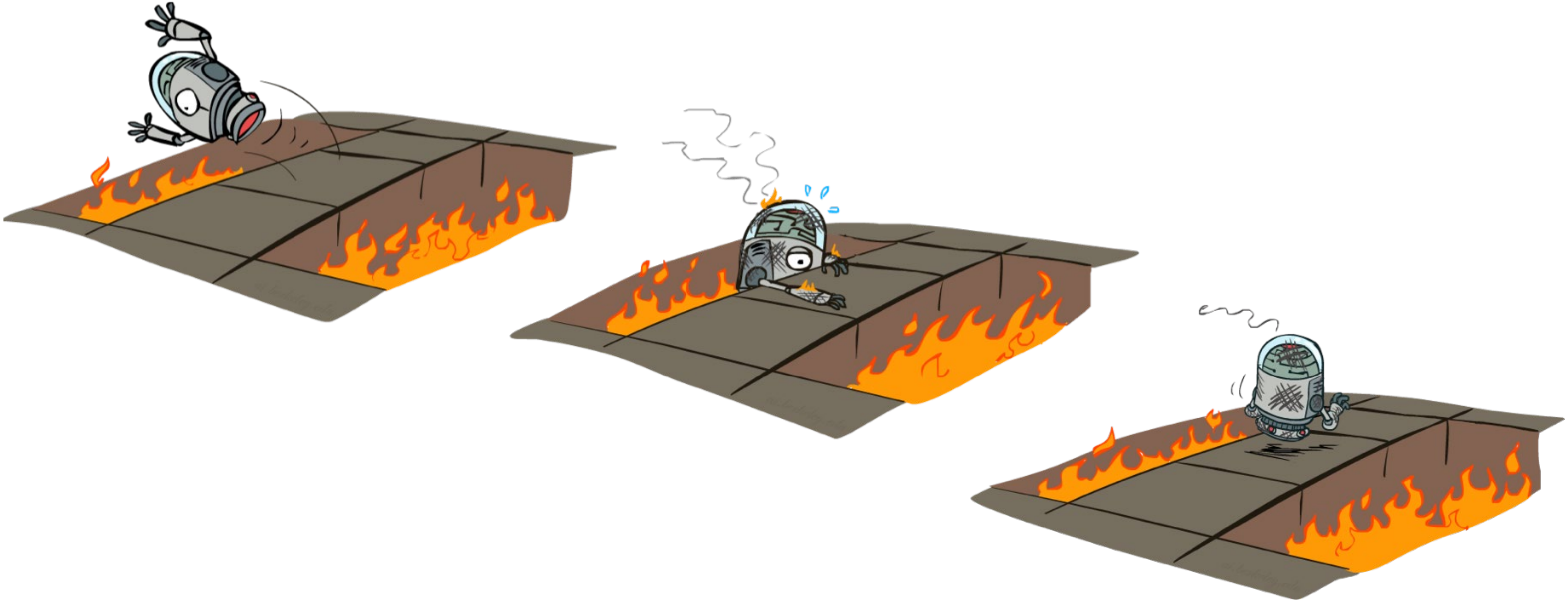
$$Q(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$



- Idea: learn Q-values using samples, not values
- Makes action selection model-free too!

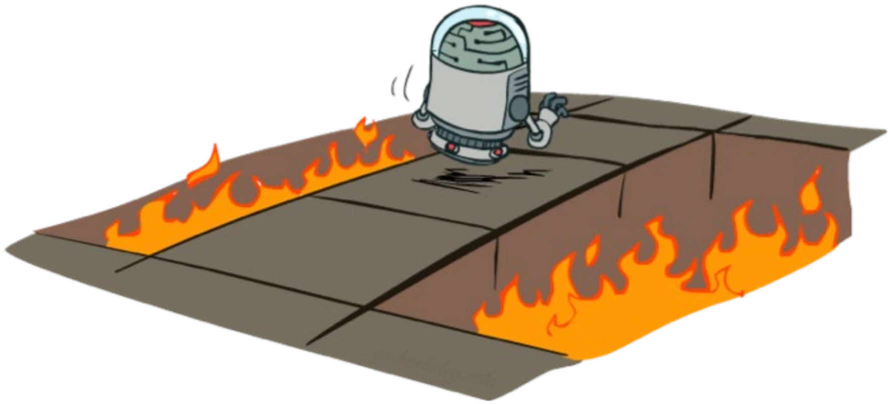
Active Reinforcement Learning

- The agent can use the feedback it receives to iteratively update its policy



Active Reinforcement Learning

- Full reinforcement learning: optimal policies (like value iteration)
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You choose the actions now
  - Goal: learn the optimal policy / values



- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation

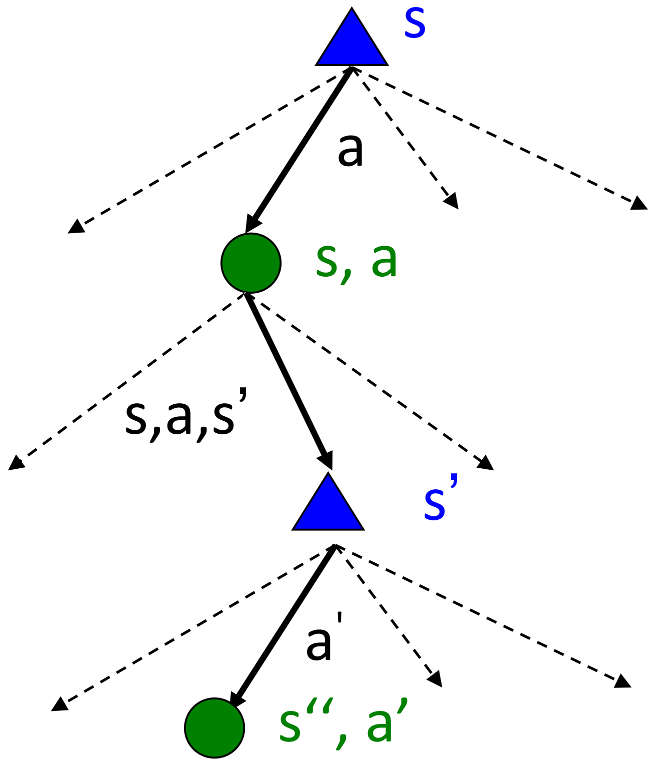
Detour: Optimal V and Q values (see previous lectures for more details)

- $V^*(s)$ = expected utility starting in s and acting optimally from that point onwards

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

  - The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.



Bellman equation

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$

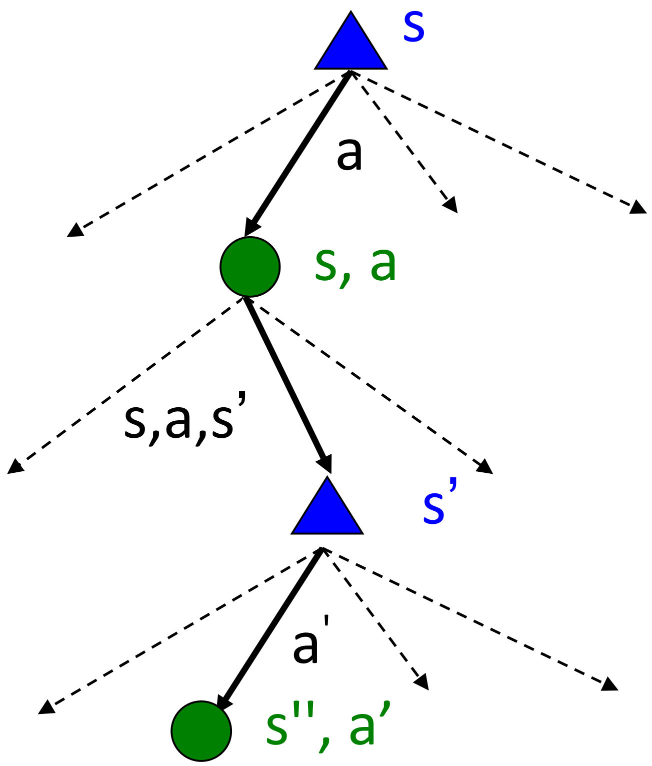
Detour: Q-Value Iteration (see previous lectures for more details)

- Value iteration: find successive (depth-limited) values
  - Start with $V_0(s) = 0$, which we know is right
  - Given V_k, calculate the depth *k+1* values for all states:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

- But Q-values are more useful, so compute them instead
  - Start with $Q_0(s,a) = 0$, which we know is right
  - Given Q_k, calculate the depth k+1 q-values for all q-states:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s'',a')\right]$$



Temporal Difference Q-learning

- Q-Learning: sample-based Q-value iteration

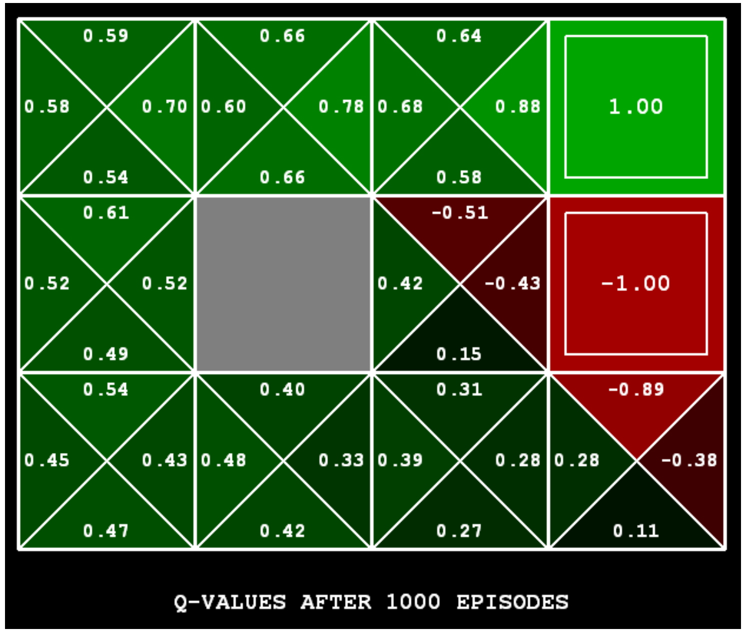
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right]$$

- Learn Q(s,a) values as you go
  - Receive a sample (s,a,s’,r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Update the old estimate using the new sample estimate:
    - Again, *a* is the learning rate

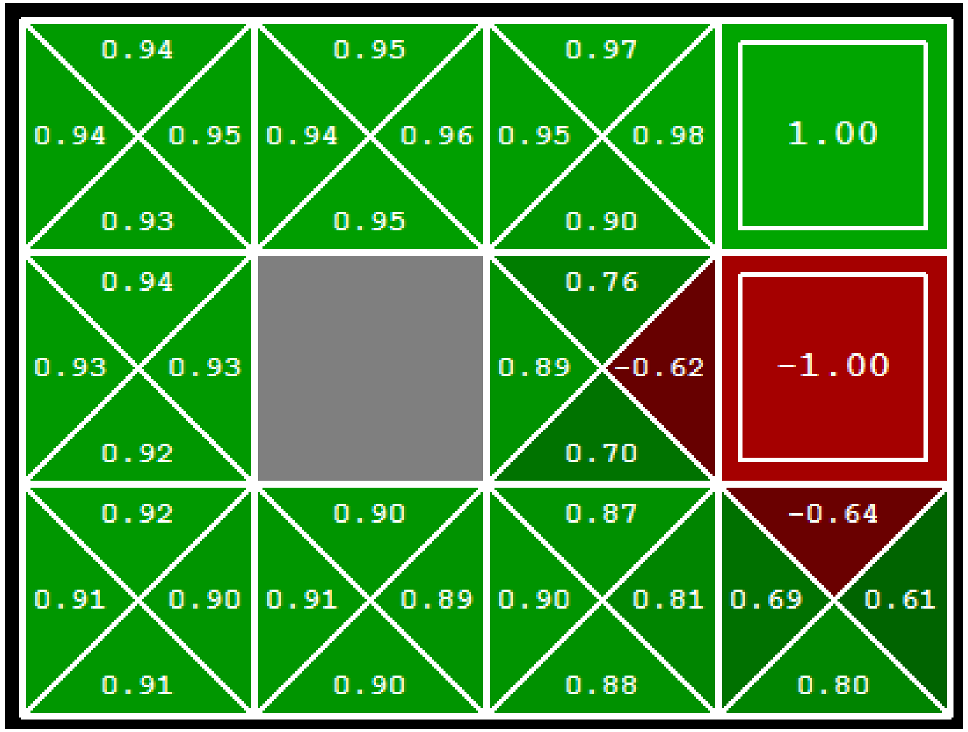
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$$



Detour: Extracting policy from Q values (see previous lecture)

- If we have q-values, completely trivial to decide
  - Select the action that takes us to the q-state with the max q-value
  - So we don't need T,R components

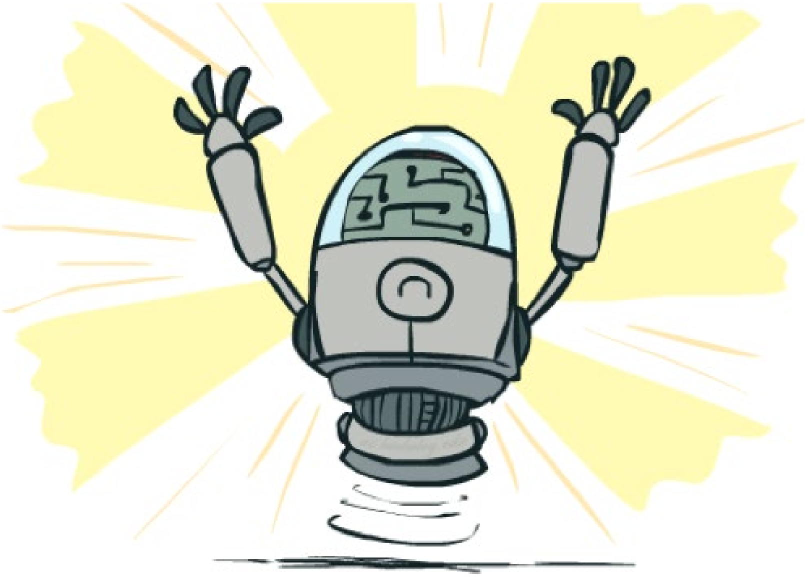
$$\pi^*(s) = \arg\max_a Q^*(s,a)$$



Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!
- This is called off-policy learning
- TD learning is an example of on-policy learning: A policy is followed and information from policy-dependent sampling of the value function is not used immediately to improve the policy

- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate *a* small enough
  - … but not decrease it too quickly

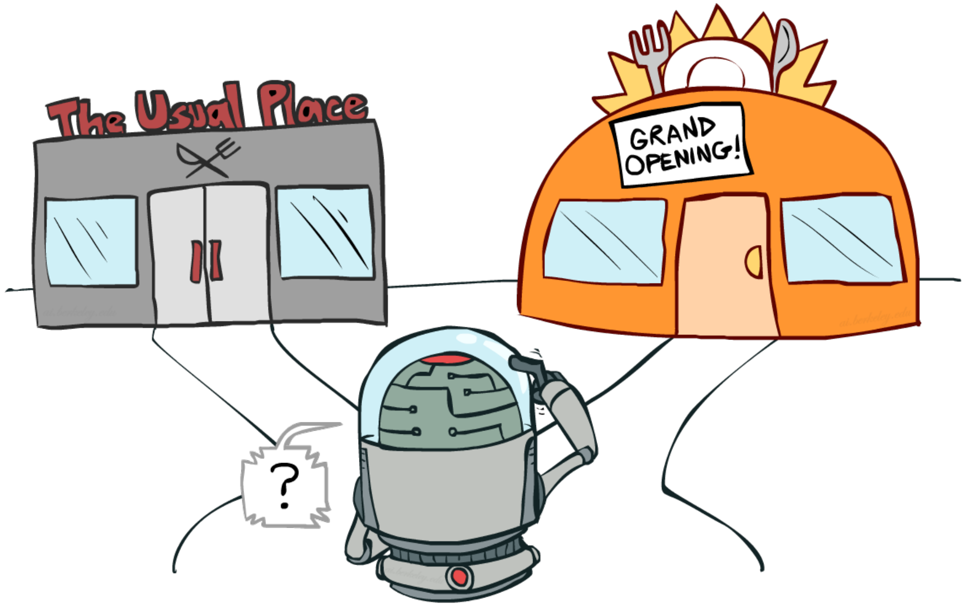


Outline

- Model-free learning
- Direct policy evaluation
- TD-learning
- Q-learning
- Exploration-vs-Exploitation
- Things you should know from this lecture & reading material

Exploration vs Exploitation dilemma

- Exploitation: Make best decision given current information
- Exploration: Gather more information that might lead us to better decisions in the future
- Examples
  - Restaurant selection
    - Exploitation: go to your favorite restaurant
    - Exploration: try a new restaurant
  - Game playing:
    - Exploitation: play the move you believe is best
    - Exploration: play a random move



Exploration schemes

- ε-greedy policies
  - Explore with probability ε and exploit with probability 1-ε, 0≤ε ≤1
  - More concretely:
    - With probability ε, select a random action a
    - With probability 1-ε, select based on current policy, i.e., $a = \arg_a \max Q(s,a)$
- Problems with fixed ε-greedy
  - Setting ε
    - Small ε: too little exploration
    - Large ε: too little exploitation
  - You do eventually explore the space, but keep thrashing around once learning is done
- One solution: lower ε over time
  - Still actions are selected uniformly at random
- Another solution: exploration functions

Exploration functions

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$$

- When to explore?
  - ε-greedy: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring
- Exploration function
  - Modify q-value update to consider the frequency of visiting a particular state

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} f(s',a')]$$

  - f(s,a) is the exploration function typically defined as

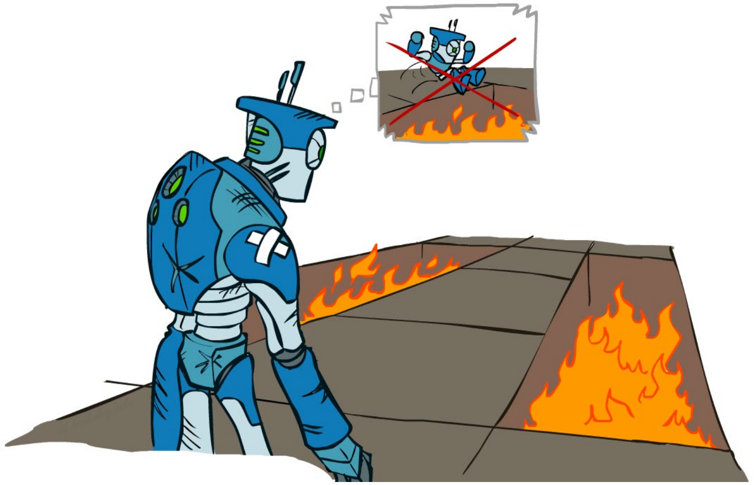
$$f(s,a) = Q(s,a) + \frac{k}{N(s,a)}$$

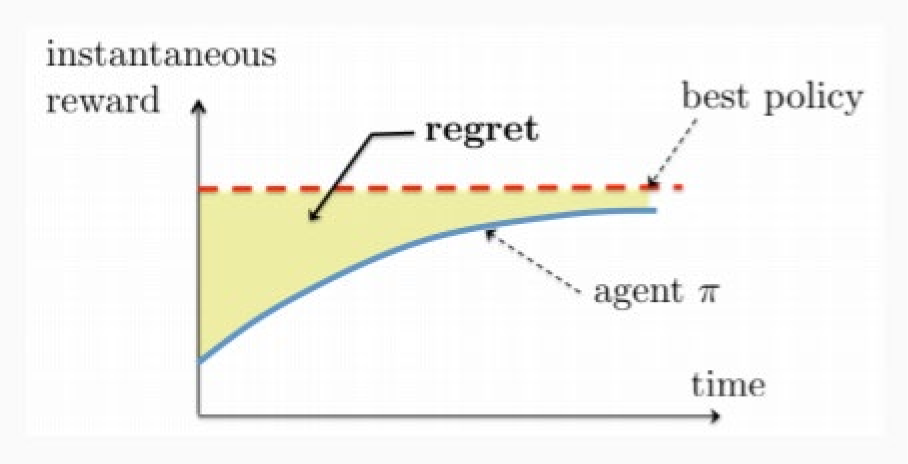
    - N(s,a) is the frequency of visiting a particular (s,a) state
    - k is a predefined value

  - Give some preference to less visited states
  - After enough visits it relies on the q-value

Regret

- Even if you learn the optimal policy, you still make mistakes along the way!
  - Example: both ε-greedy and exploration function learn the optimal policy but make different mistakes along the way
  - How can we compare these strategies?
- Regret is a measure of your total mistake cost:
  - Defined as the difference between the cumulative reward of the optimal policy and that gathered by π.
- Example: random exploration and exploration functions both end up optimal, but random exploration has higher regret





Overview and Reading

- Overview
  - Model-free learning
  - Direct policy evaluation
  - TD-learning
  - Q-learning
  - Exploration-exploitation tradeoff
- Reading
  - Chapter 16& 23, AI book, 4^{th} edition
  - RL Book, Barto and Sutton, 2^{nd} edition
  - Introduction to Reinforcement Learning with David Silver (DeepMind)
  - Stanford CS234: Reinforcement Learning with Emma Brunskill

Hands on experience

- Check project 3, released today

Thank you

Questions/Feedback/Wishes?

Acknowledgements

- The slides are based on
  - CS 188 | Introduction to Artificial Intelligence, Berkeley
  - Artificial Intelligence: A modern approach (Russel and Norvig), 4^{th} edition