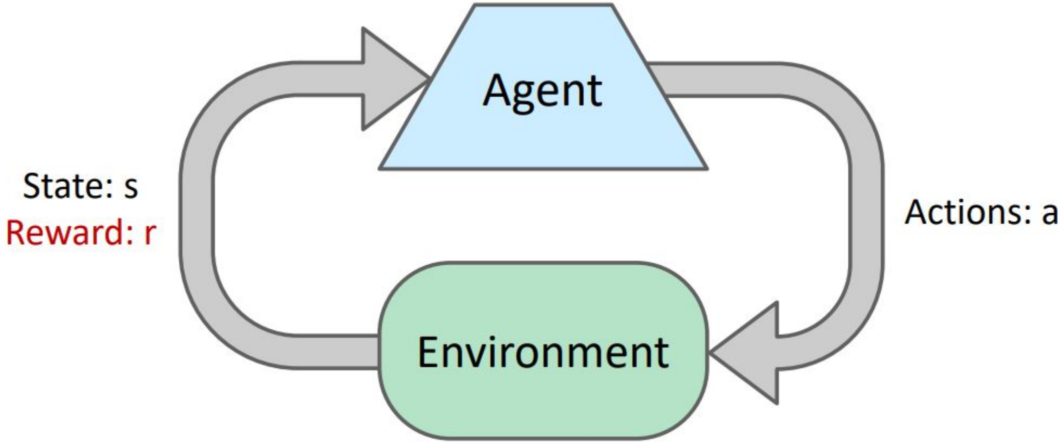
Reinforcement Learning I

CSE 4617: Artificial Intelligence

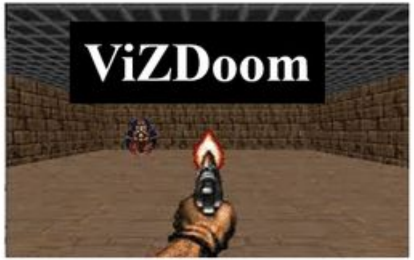
Isham Tashdeed
Lecturer, CSE

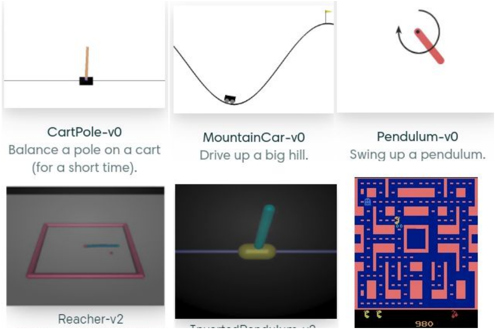
Reinforcement Learning

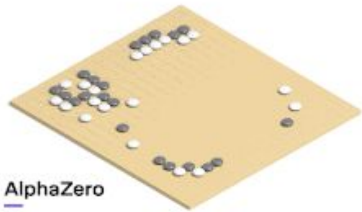
- Receive feedback from rewards
- Agent's utility is defined by the reward function
- Must act in such a way that maximizes expected utility
- All learning is based on observed samples of outcomes → No model of the world available

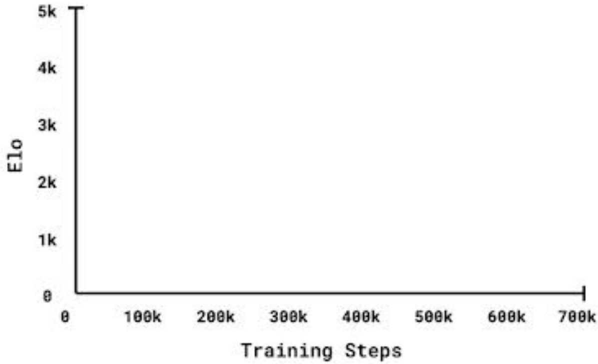


Reinforcement Learning in Use



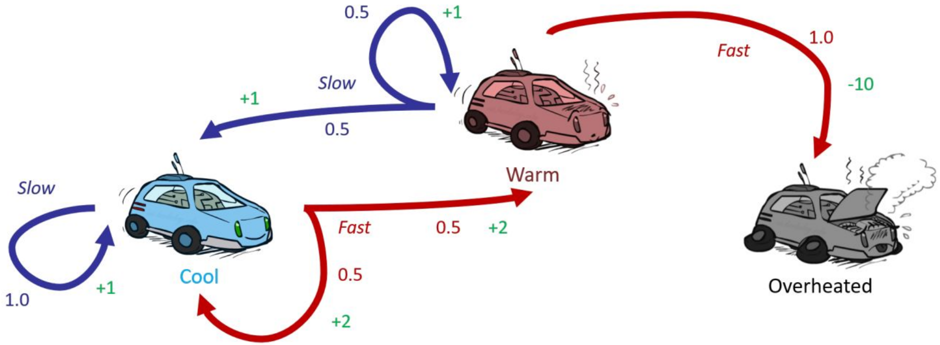






Formalizing RL Problems

- Still assume an MDP
  - A set of states $s \in S$
  - A set of actions per state A
  - A probability model $T(s, a, s')$
  - A reward function $R(s, a, s')$
- Still trying to find the most optimal policy π^*
- We do not know $T(s, a, s')$ and $R(s, a, s')$
- Must actually try out actions and states to learn



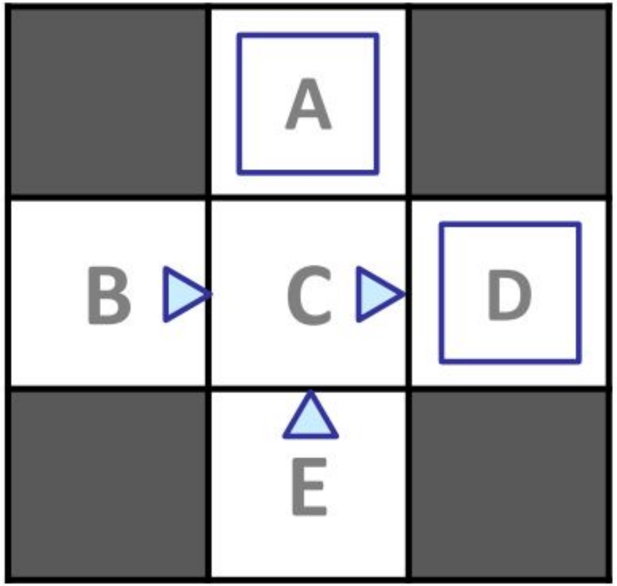
Model-Based Learning

Model-Based Learning

- Learn an approximate model based on experience
- Solve for the policy as if the learned model was the correct one

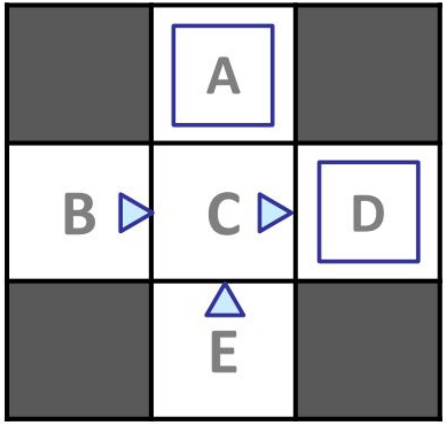
Steps:

- Learn empirical MDP model
  - Count *s'* for each *s*, *a*
  - Normalize for an estimate of $T(s, a, s')$
  - Keep track of rewards for an estimate of $R(s, a, s')$
- Solve the learned MDP



Assume: $\gamma = 1$

Model-Based Learning



Assume: $\gamma = 1$

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

$\hat{T}(s, a, s')$

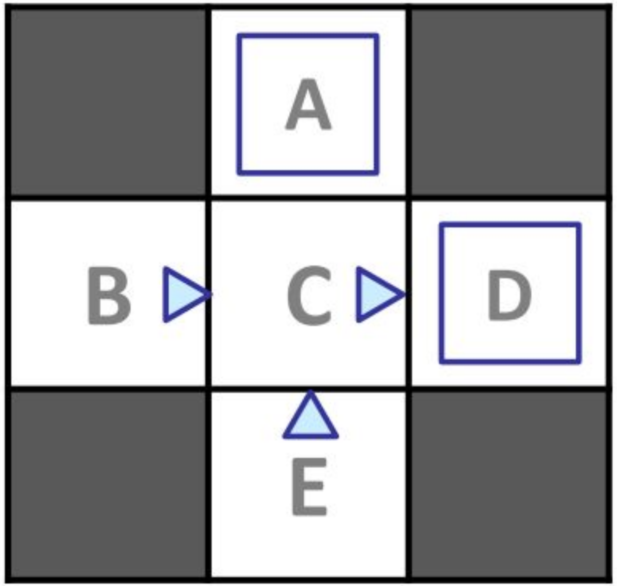
T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
...

$\hat{R}(s, a, s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
...

Model Free Learning

Passive Reinforcement Learning



Assume: $\gamma = 1$

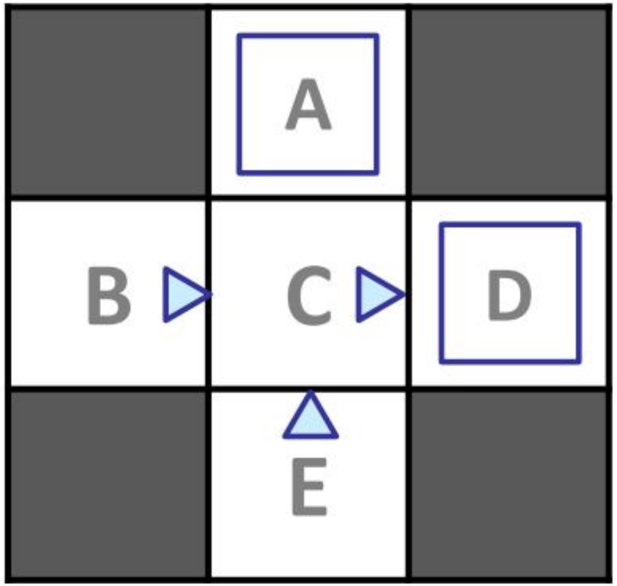
Policy Evaluation

- Input is a fixed policy $\pi(s)$
- You do not know $T(s, a, s')$
- You do not know $R(s, a, s')$
- Goal → Learn the state values

In this case:

- No choice about what actions to take
- Just execute the policy and learn from experience
- This is NOT offline planning! You actually take actions in the world

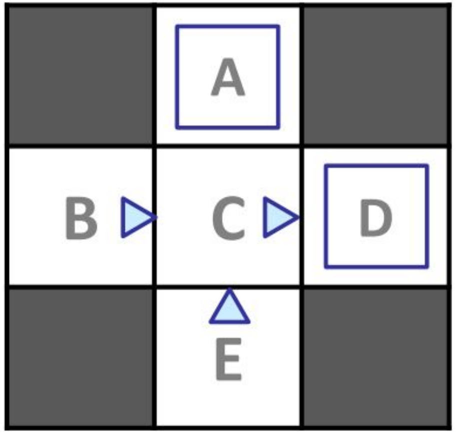
Direct Evaluation



Assume: $\gamma = 1$

- Compute values for each state following π
- Every time you visit a state, write down what the sum of discounted rewards turned out to be
- Average those samples to get an estimate of the value

Direct Evaluation



Assume: $\gamma = 1$

Episode 1

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

Episode 2

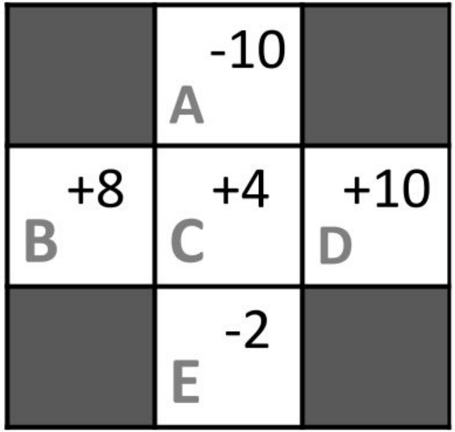
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

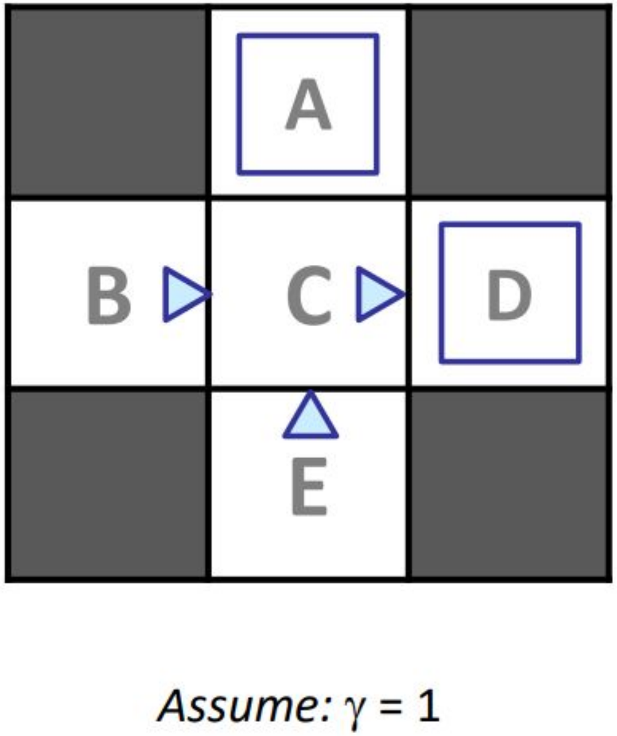
E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

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



Direct Evaluation



Pros:

- Easy to understand
- Does not require prior knowledge on T or R
- It eventually computes the correct average values, using just sample transitions → Need lots of samples

Cons:

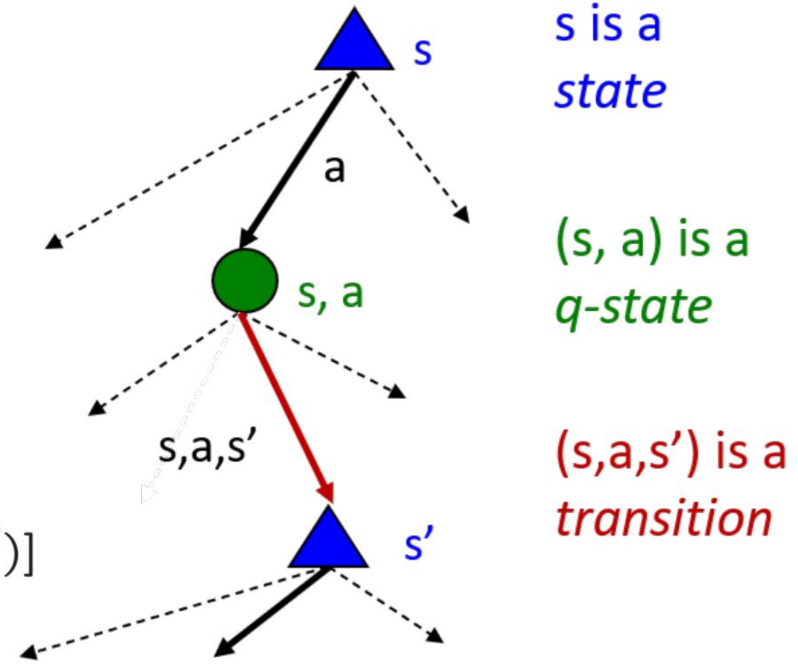
- Long time to learn
- Have to learn each state separately
- Wastes information about state connections

Why Don’t We Use Policy Evaluation

- Each round, replace *V* with a one step look ahead layer over *V*
- This approach fully exploited the connections between the states
- Unfortunately, we need *T* and *R* to do it!

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

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



- How can we do this update to *V* without knowing *T* and *R*?

Sample Based Policy Evaluation

- We want to improve our estimate of V by computing these averages

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

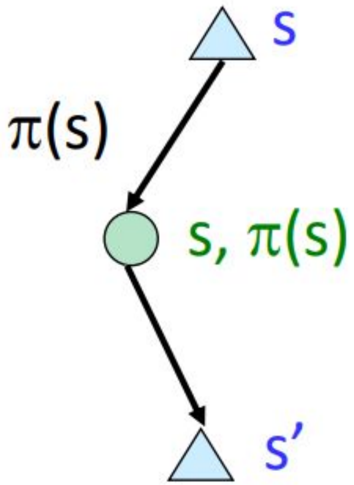
- Take samples of outcomes *s'* (by doing the action!) and take average

$$sample_1 = R(s, \pi(s), s_1') + \gamma V_k^{\pi}(s_1')$$
$$sample_2 = R(s, \pi(s), s_2') + \gamma V_k^{\pi}(s_2')$$
$$\dots$$
$$sample_n = R(s, \pi(s), s_n') + \gamma V_k^{\pi}(s_n')$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

Temporal Difference Learning

Temporal Difference Learning



- Learn from every experience!
- Update $V(s)$ each time we experience a transition (s, a, s', r)
- Policy still fixed, still doing evaluation!

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

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

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

Exponential Moving Average

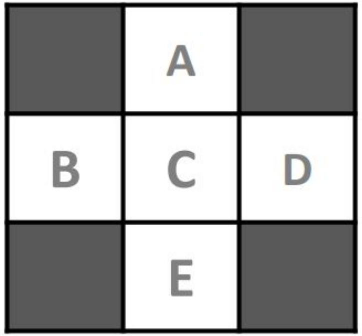
- The running interpolation update: $\bar{x}_n = (1-\alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
- Makes recent samples more important:

$$\bar{x}_n = \frac{x_n + (1-\alpha) \cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

- Forgets about the past (distant past values were wrong anyway)
- Decreasing learning rate (alpha) can give converging averages

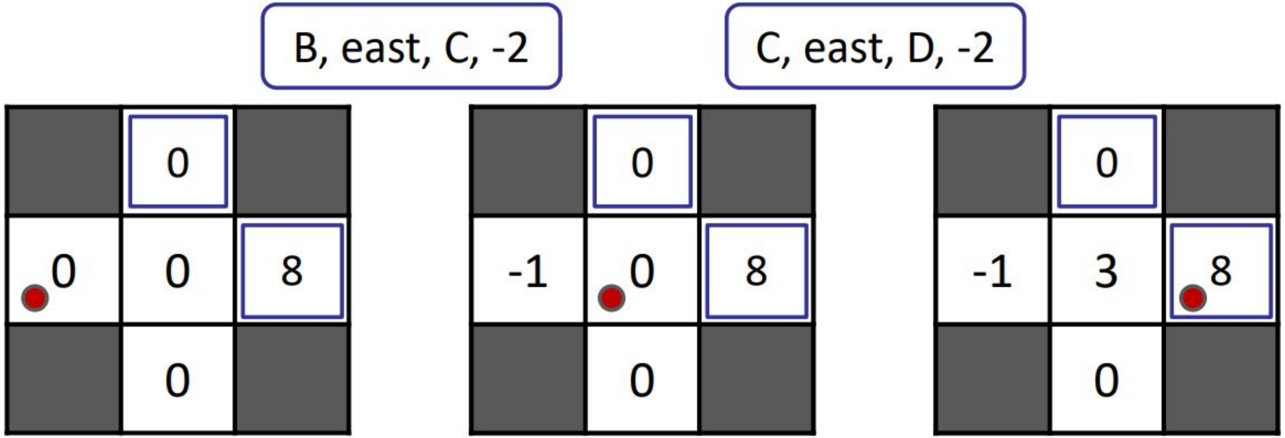
Temporal Difference Learning

States



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

Observed Transitions



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

Problems with TD Value Learning

- TD value learning is a model-free approach to do policy evaluation, trying to mimic the Bellman updates
- But it is only good for policy evaluation
- If we are asked to create a new policy using TD value learning, we would need *T* and *R*

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

The better approach is to learn Q-values, not just values

$$\pi(s) = \underset{a}{\text{argmax}}\, Q(s, a)$$

Q-Value Iteration

Value Iteration → Find successive (depth-limited) values

- Start with $V_0(s)$=0
- Given V_k find the depth k + 1 for all states

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

Q-value Iteration → Same thing, just for q-values

- Start with $Q_0(s, a)$=0
- Given Q_k find the depth k + 1 for all q-states

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

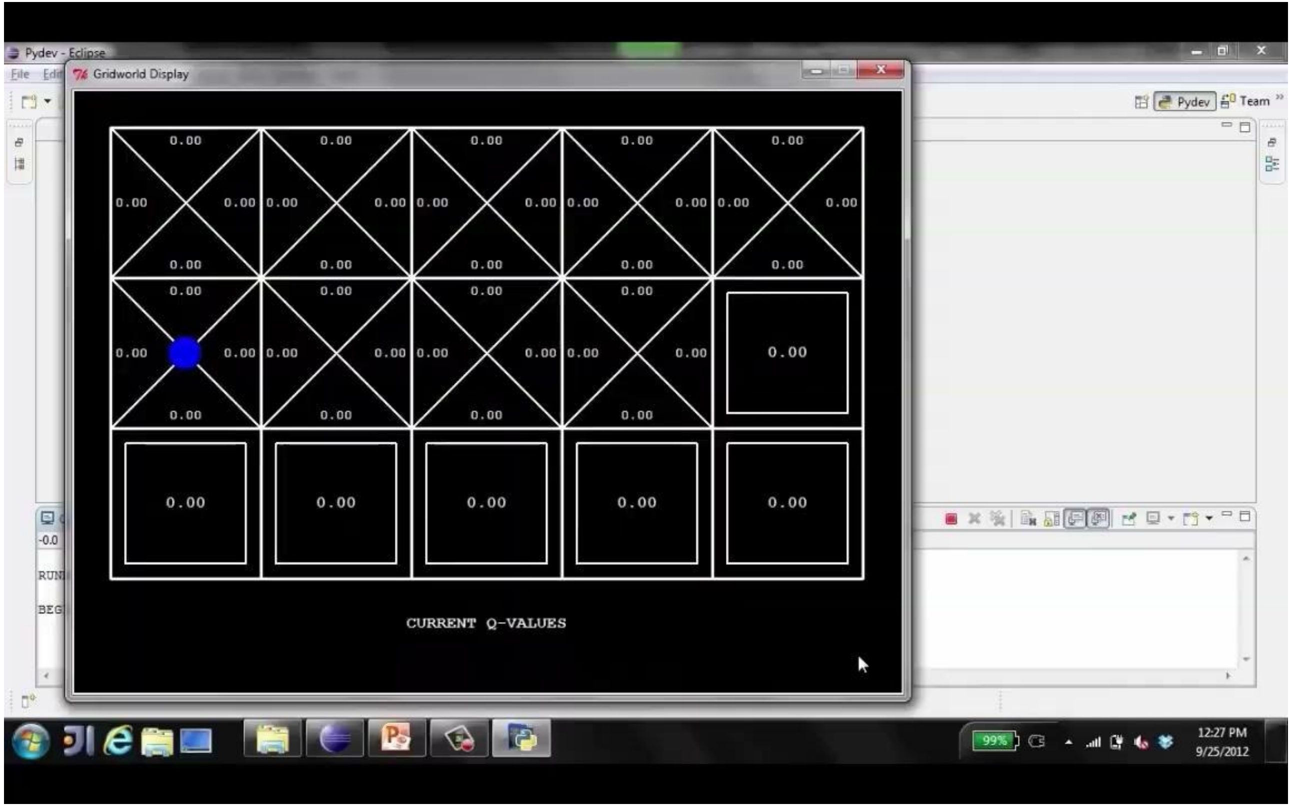
Q-Learning

Learn $Q(s, a)$ as you go

- Receive a sample (s, a, s', r)
- Consider the old estimate $Q(s, a)$
- New sample estimate $\rightarrow$ $sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$
- Incorporate the new estimate into the running average
  - $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha\,(sample)$

Q-learning converges to optimal policy- even if you're acting suboptimally! This is called off-policy learning!

Q-Learning



Pydev - Eclipse
Gridworld Display
0.00
CURRENT Q-VALUES
Pydev
Team
12:27 PM
9/25/2012

Thank you