Reinforcement Learning

CS 486/686: Introduction to Artificial Intelligence

Outline

- What is reinforcement learning
- Quick MDP review
- Passive learning
  - Temporal Difference Learning
- Active learning
  - Q-Learning

What is RL?

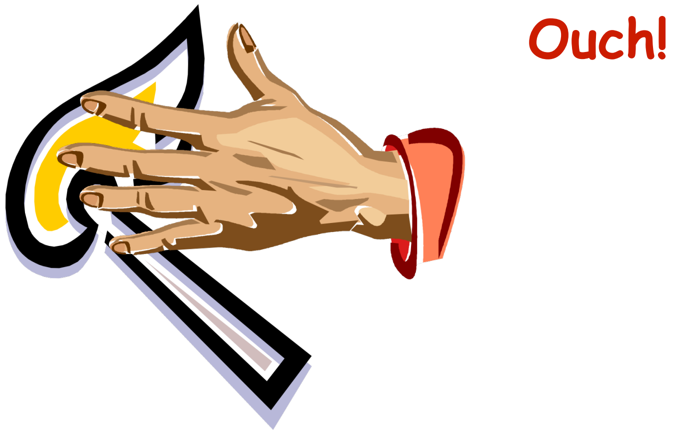
- Reinforcement learning is learning what to do so as to maximize a numerical reward signal
- Learner is not told what actions to take
- Learner discovers value of actions by
  - Trying actions out
  - Seeing what the reward is

What is RL?

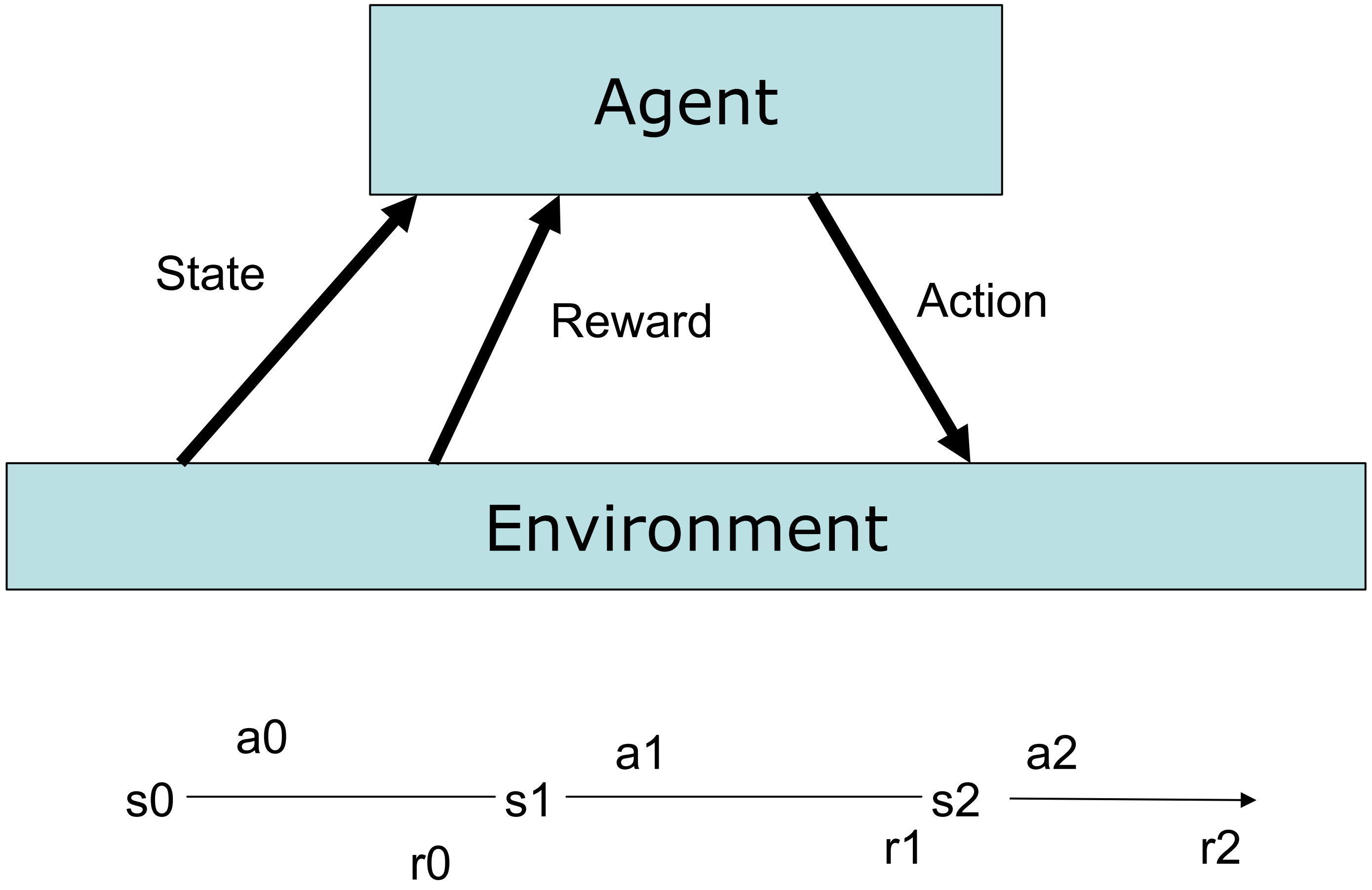
- Another common learning framework is supervised learning (we will see this later in the semester)

Supervised learning

Reinforcement learning



Reinforcement Learning Problem



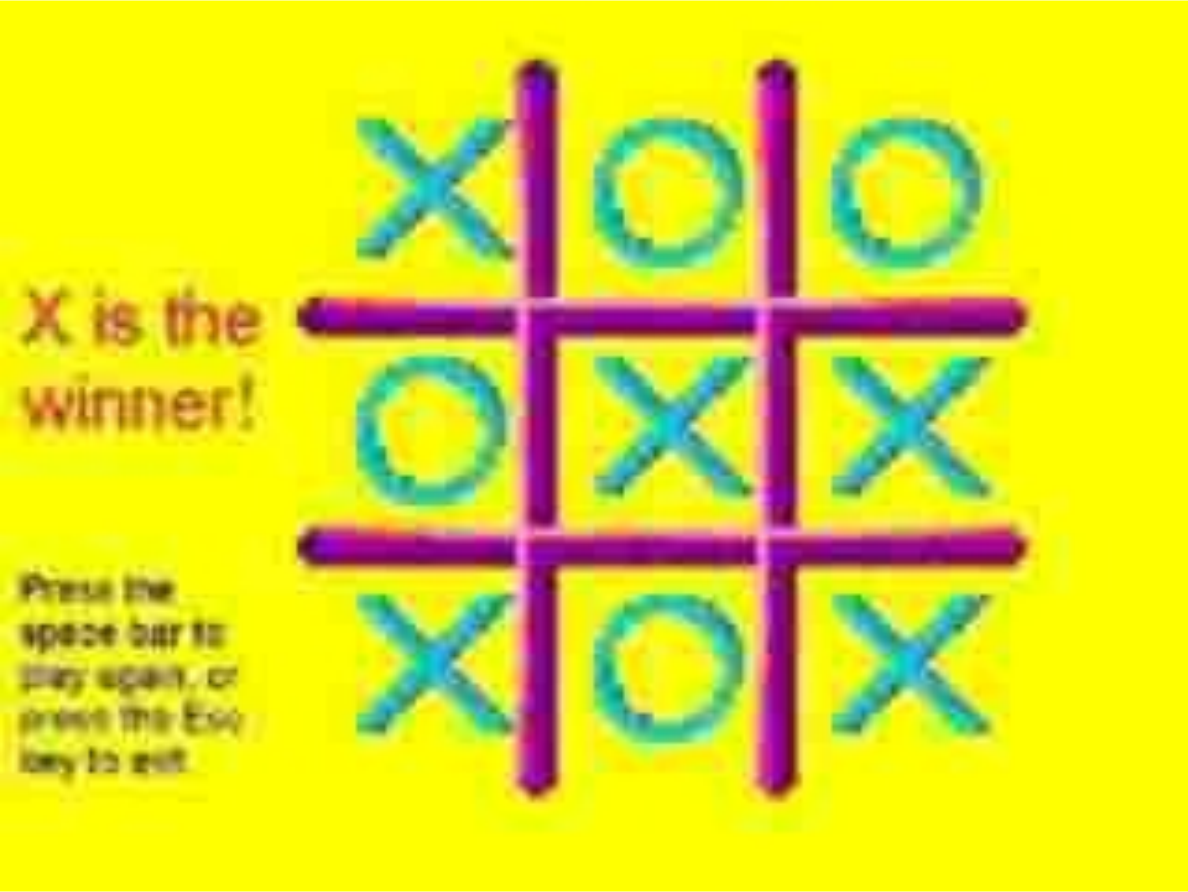
Goal: Learn to choose actions that maximize $r_0+\gamma\ r_1+\gamma^2 r_2+\ldots$, where $0\cdot<\gamma<1$

Example: Slot Machine

- **State**: Configuration of slots
- **Actions**: Stopping time
- **Reward**: $$$
- **Problem**: Find π: S →A that maximizes the reward

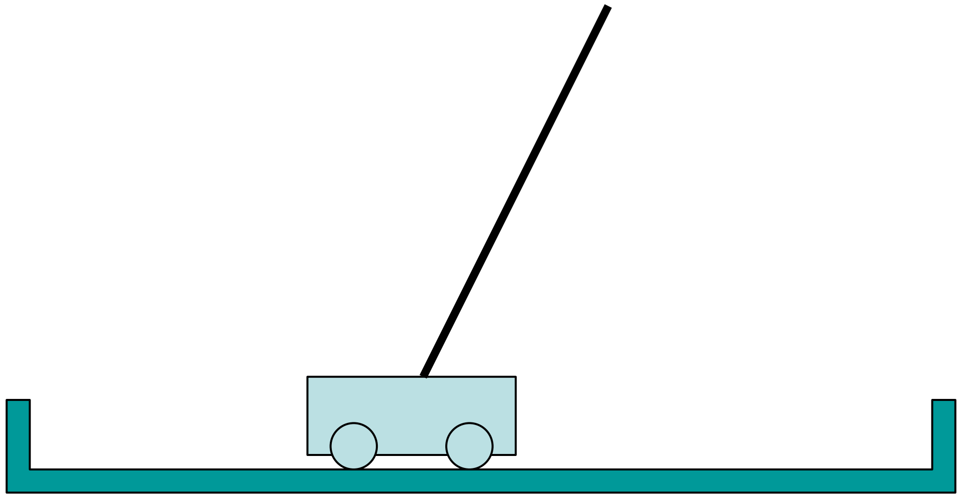
Example: Tic Tac Toe

- **State**: Board configuration
- **Actions**: Next move
- **Reward**: 1 for a win, -1 for a loss, 0 for a draw
- **Problem**: Find π: S →A that maximizes the reward



Example: Inverted Pendulem

- **State**: x(t), x'(t), θ(t), θ'(t)
- **Actions**: Force F
- **Reward**: 1 for any step where the pole is balanced
- **Problem**: Find π: S →A that maximizes the reward



Example: Mobile Robot

- **State**: Location of robot, people
- **Actions**: Motion
- **Reward**: Number of happy faces
- **Problem**: Find π: S →A that maximizes the reward

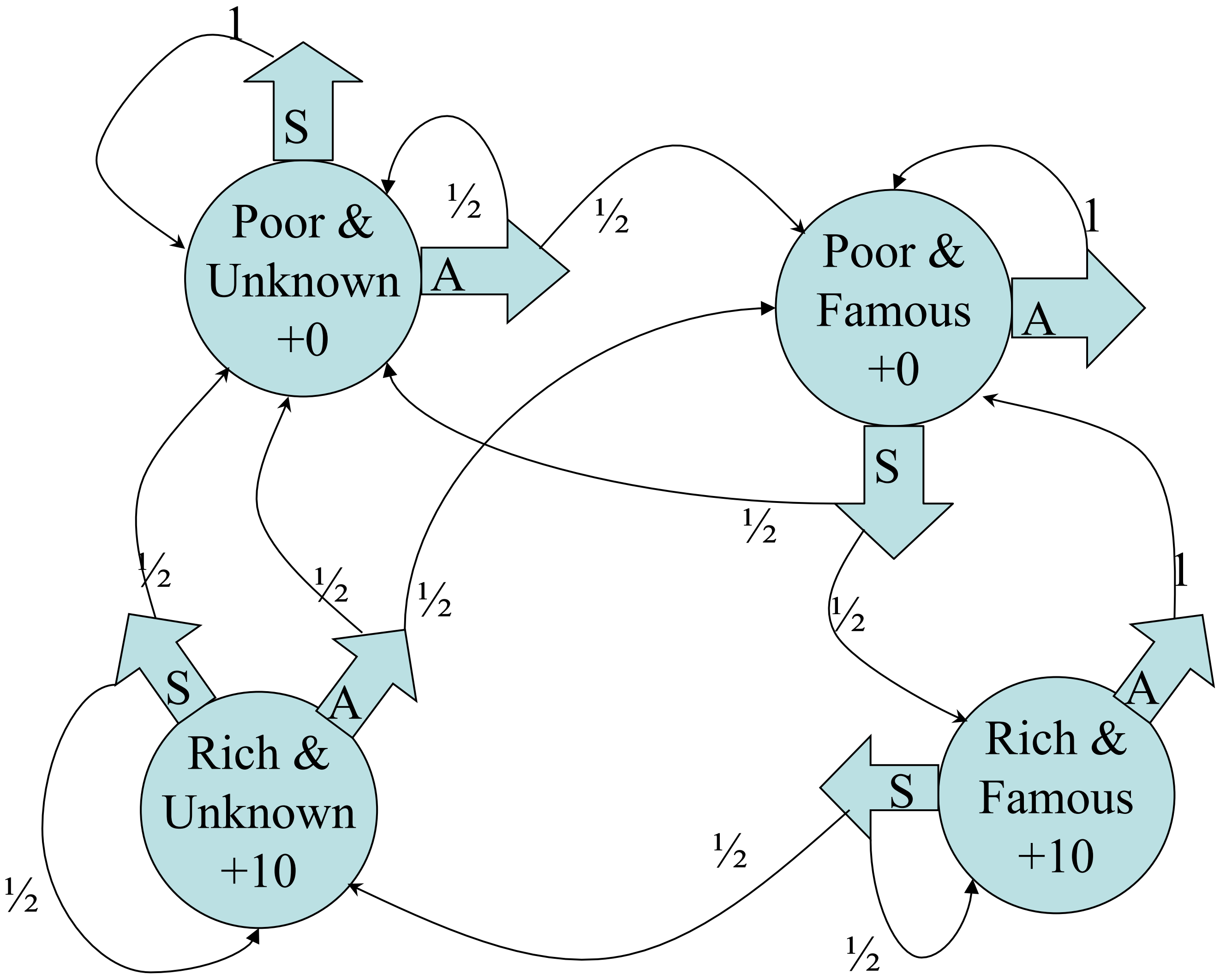
Reinforcement Learning Characteristics

- Delayed reward
  - Credit assignment problem
- Exploration and exploitation
- Possibility that a state is only partially observable
- Life-long learning

Reinforcement Learning Model

- Set of states S
- Set of actions A
- Set of reinforcement signals (rewards)
    - Rewards may be delayed

Markov Decision Process



γ = 0.9

You own a company

In every state you must choose between **S**aving money or **A**dvertising

Markov Decision Process

- Set of states $\{s_1, s_2, \ldots s_n\}$
- Set of actions $\{a_1, \ldots, a_m\}$
- Each state has a reward $\{r_1, r_2, \ldots r_n\}$
- Transition probability function

$$P_{ij}^{k} = (\text{Next}{=}s_j \mid \text{This}{=}s_i \text{ and I take action } a_k)$$

- ON EACH STEP…

0. Assume your state is s_i
1. You get given reward r_i
2. Choose action a_k
3. You will move to state s_j with probability P_{ij}^{k}
4. All future rewards are discounted by γ

MDPs and RL

- With an MDP our goal was to **find the optimal policy given the model**
  - Given rewards and transition probabilities
- In RL our goal is to **find the optimal policy but we start without knowing the model**
  - Not given rewards and transition probabilities

Agent’s Learning Task

- Execute actions in the world
- Observe the results
- Learn policy π:S→A that maximizes $E[r_t+\Upsilon r_{t+1}+\Upsilon^2 r_{t+2}+\ldots]$ from any starting state in S

Types of RL

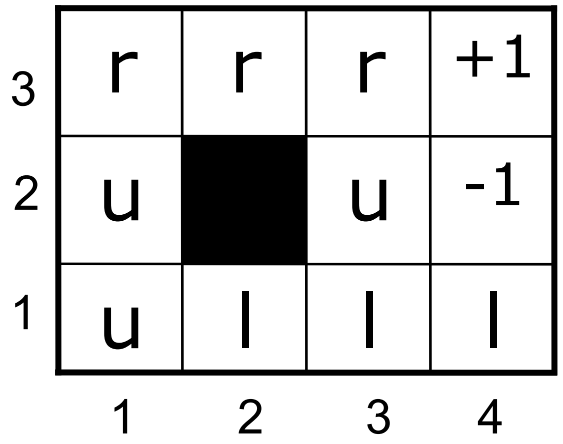
Model-based vs Model-free

- **Model-based**: Learn the model of the environment
- **Model-free**: Never explicitly learn P(s’ls,a)

Passive vs Active

- **Passive**: Given a fixed policy, evaluate it
- **Active**: Agent must learn what to do

Passive Learning



3	r	r	r	+1
2	u	■	u	-1
1	u	l	l	l
	1	2	3	4

$\gamma = 1$

r_i = -0.04 for non-terminal states

We do not know the transition probabilities

$(1,1)\to(1,2)\to(1,3)\to(1,2)\to(1,3)\to(2,3)\to(3,3)\to(4,3)_{+1}$
$(1,1)\to(1,2)\to(1,3)\to(2,3)\to(3,3)\to(3,2)\to(3,3)\to(4,3)_{+1}$
$(1,1)\to(2,1)\to(3,1)\to(3,2)\to(4,2)_{-1}$

What is the value, $V^*(s)$ of being in state s?

Direct Utility Estimation (Sampling)

3	r	r	r	+1
2	u	■	u	-1
1	u	l	l	l
	1	2	3	4

$\gamma = 1$

r_i = -0.04 for non-terminal states

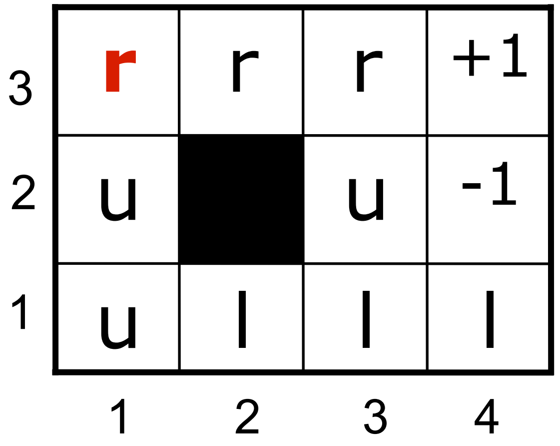
(1,1)→(1,2)→(1,3)→(1,2)→(1,3)→(2,3)→(3,3)→$(4,3)_{+1}$
(1,1)→(1,2)→(1,3)→(2,3)→(3,3)→(3,2)→(3,3)→$(4,3)_{+1}$
(1,1)→(2,1)→(3,1)→(3,2)→$(4,2)_{-1}$

What is the value, $V^*(s)$ of being in state s?

$$V^{\pi}(S) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t)]$$

Adaptive Dynamic Programming (ADP)

$\gamma = 1$



3	**r**	r	r	+1
2	u	■	u	-1
1	u	l	l	l
	1	2	3	4

r_i = -0.04 for non-terminal states

$$V^{\pi}(s_i) = r(s_i) + \gamma \sum_j P^{\pi}_{ij} V^{\pi}(s_j)$$

(1,1)→(1,2)→ **(1,3)** → (1,2)→**(1,3)**→ (2,3)→(3,3)→(4,3)$_{+1}$
(1,1)→(1,2)→**(1,3)**→(2,3)→(3,3)→(3,2)→(3,3)→(4,3)$_{+1}$
(1,1)→(2,1)→(3,1)→(3,2)→(4,2)$_{-1}$

$P_{(1,3)(2,3)}^{r}=2/3$
$P_{(1,3)(1,2)}^{r}=1/3$

Use this information in the Bellman equation

Temporal Difference

Key Idea: Use observed transitions to adjust values of observed states so that they satisfy Bellman equations

$$V^{\pi}(s) = V^{\pi}(s) + \underset{\swarrow\ \text{Learning rate}}{\alpha}\underbrace{(r(s) + \gamma V^{\pi}(s') - V^{\pi}(s))}_{\text{Temporal difference}}$$

Theorem: If α is appropriately decreased with the number of times a state is visited, then $V^{\pi}(s)$ converges to the correct value.

- α must satisfy $\Sigma_n \alpha(n) \rightarrow \infty$ and $\Sigma_n \alpha^2(n) < 1$

Temporal Difference

- No explicit model of T or R
- Estimate V and expectation through samples
- Update from each experience
  - Update V(s) after each state transition
  - Likely outcomes s’ will contribute updates more often

Temporal Difference

- Temporal difference learning of values
  - Policy is still fixed (doing evaluations)
  - Move values toward sample of V(s) (running average)

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

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

TD-Lambda

Idea: Update from the whole training sequence, not just a single state transition

$$V^{\pi}(s_i) \to V^{\pi}(s_i) + \alpha \sum_{m=i}^{\infty} \lambda^{m-i} \left[r(s_m) + \gamma V^{\pi}(s_{m+1}) - V^{\pi}(s_m) \right]$$

Special cases:

- Lambda = 1 (basically ADP)
- Lambda=0 (TD)

Active Learning

Recall that real goal is to find a good policy

- If the transition and reward model is known then
  - $V^*(s)=\max_a[r(s)+\gamma\Sigma_{s'}P(s'|s,a)V^*(s')]$
- If the transition and reward model is unknown
  - Improve policy as agent executes it

Q-Learning

Key idea: Learn a function Q:SxA->**R**

- Value of a state-action pair
- Optimal Policy: $\pi^*(s)=\text{argmax}_a\, Q(s,a)$
- $V^*(s)=\max_a Q(s,a)$
- Bellman's equation:
  $Q(s,a)=r(s)+\gamma\Sigma_{s'}P(s'|s,a)\max_{a'}Q(s',a')$

On-Policy/Off-Policy

An active RL agent can have two different types of policies

- Behaviour policy: used to generate actions and gather data
- Learning policy: target policy to learn

On policy learning: Behaviour = Learning

Off policy learning: Behaviour != Learning

On Policy Learning: SARSA

Agent learns policy being used, including exploration actions (policy used is usually non-deterministic so as to ensure exploration). E.g. epsilon-greedy

```
For each (s,a) initialize Q(s,a)
Observe current state S
Choose action a from s using policy
Loop
    - Take action a, observe r and s'
    - Choose a' from s' according to policy
    - Update Q(s,a): $Q(s,a)=Q(s,a)+\alpha(r+\gamma Q(s',a')-Q(s,a))$
    - s=s', a=a'
```

Off Policy: Q-Learning

Target policy is learned regardless of actions chosen from exploring (agent follows a policy but learns the value of a different policy)

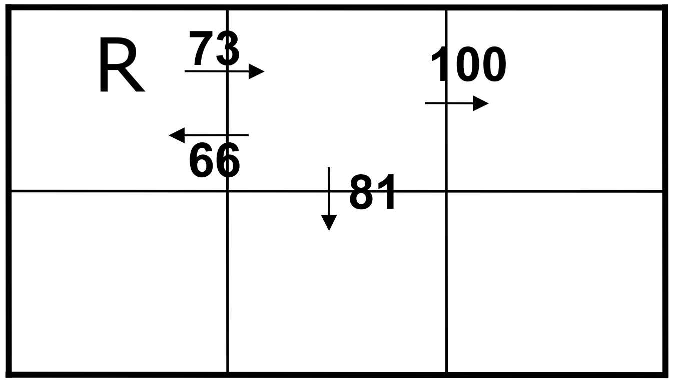
For each (s,a) initialize Q(s,a)

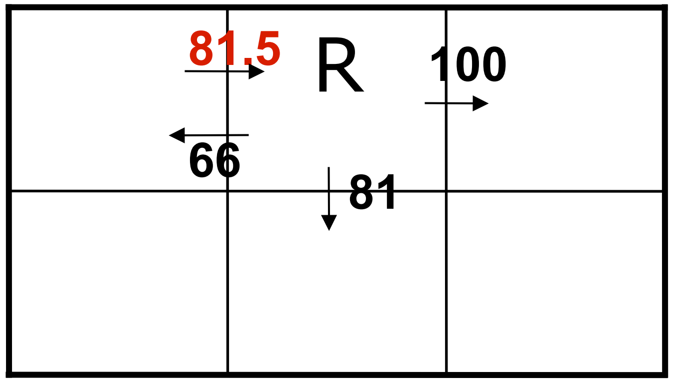
Observe current state

Loop

- Select action a and execute it
- Observe r and s'
- Update Q(s,a): $\mathbf{Q(s,a)=Q(s,a)+\alpha(r+\gamma\ max_{a'}Q(s',a')-Q(s,a))}$
- s=s'

Example: Q-Learning





r=0 for non-terminal states
γ=0.9
α = 0.5

Exploration vs Exploitation

Exploiting: Taking greedy actions (those with highest value)

Exploring: Randomly choosing actions
Need to balance the two

Common Exploration Methods

- Use an optimistic estimate of utility
- Chose best action with probability p and a random action otherwise
- Boltzmann exploration

$$P(a) = \frac{e^{Q(s,a)/T}}{\sum_{a} e^{Q(s,a)/T}}$$

Exploration and Q-Learning

Q-Learning converges to the optimal Q-values if

- Every state is visited infinitely often (due to exploration)
- The action selection becomes greedy as time approaches infinity
- The learning rate is decreased appropriately

Summary

- Active vs Passive Learning
- Model-Based vs Model-Free
- TD
- Q-learning
    - Exploration-Exploitation tradeoff