Reinforcement Learning

CS 486/686

Introduction to AI

University of Waterloo

Outline

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

- Readings: R&N Ch 21.1-21.4

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

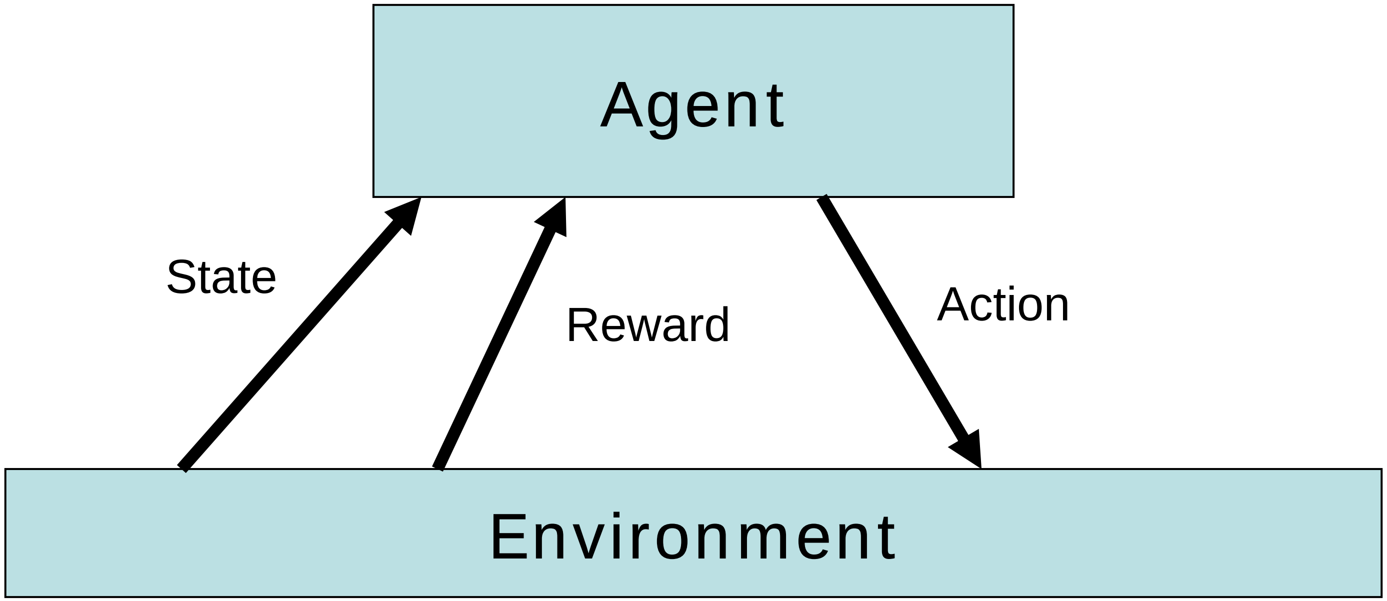
- Reinforcement learning differs from supervised learning

Supervised learning

Reinforcement learning

Ouch!

Reinforcement Learning Problem



$$s0 \xrightarrow[r0]{a0} s1 \xrightarrow[r1]{a1} s2 \xrightarrow[r2]{a2} \cdots$$

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

Example 1: Slot Machine

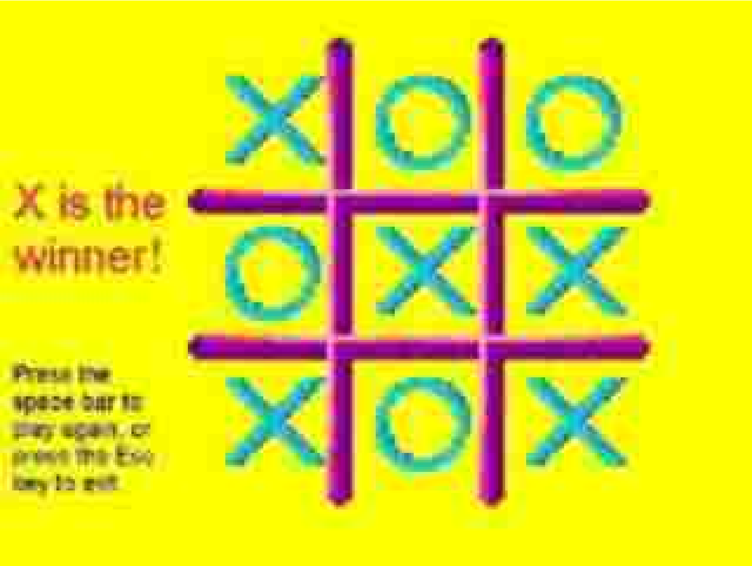
- **State**: Configuration of slots
- **Action**: Stopping time
- **Reward**: $$$

- **Problem**: Find $\pi: S \rightarrow A$ that maximizes R

Example 2: Tic Tac Toe

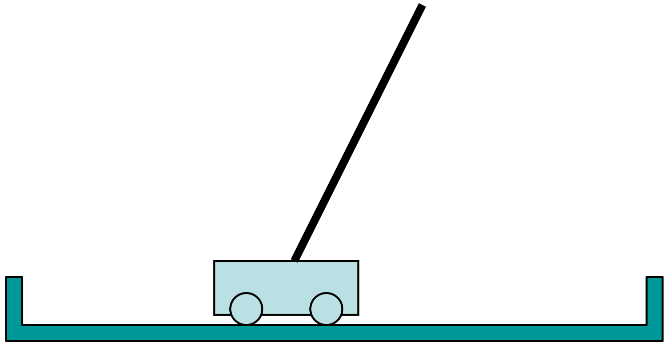
- **State**: board
- **Action**: next move
- **Reward**: 1 for win, -1 for loss, 0 for draw

- **Problem**: Find π:S$\rightarrow$A that maximizes R



Example 3: Inverted Pendulum

- **State**: $x(t), x'(t), \theta(t), \theta'(t)$
- **Action**: Force F
- **Reward**: 1 for any step where pole balanced



- **Problem**: Find $\pi: S \rightarrow A$ that maximizes R

Example 4: Mobile Robot

- **State**: location of robot, people
- **Action**: motion
- **Reward**: number of happy faces

- **Problem**: Find $\pi : S \rightarrow A$ that maximizes R

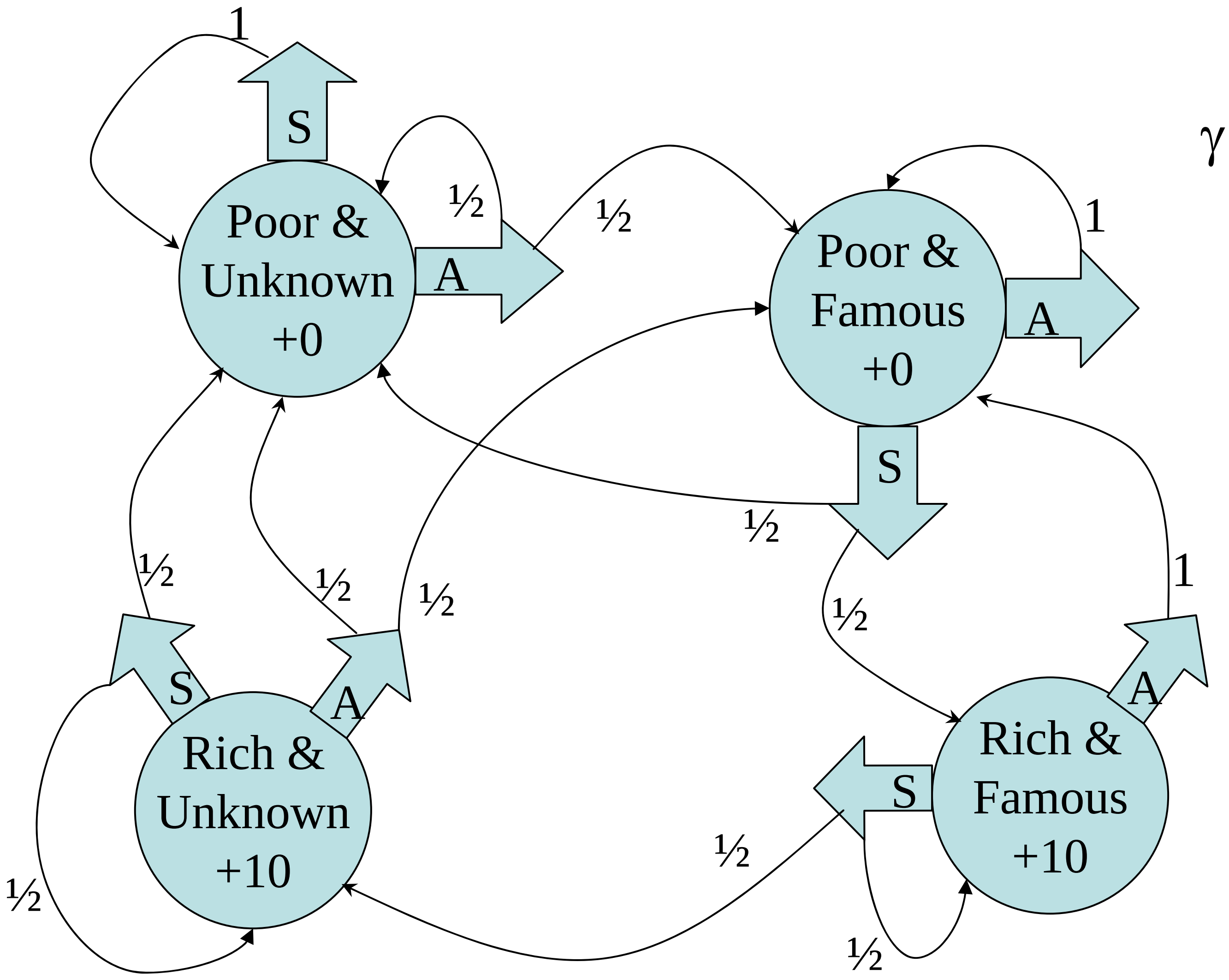
RL Characteristics

- 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
  - Actions may be non-deterministic
- Set of reinforcement signals (rewards)
  - Rewards may be delayed

A Markov Decision Process



$\gamma = 0.9$

You own a company

In every state you must choose between Saving money or Advertising

Markov Decision Processes (MDPs)

- Has a set of states $\{s_1, s_2, \ldots s_n\}$
- Has a set of actions $\{a_1, \ldots, a_m\}$
- Each state has a reward $\{r_1, r_2, \ldots r_n\}$
- Has a 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 environment
- Observe the results
- Learn policy $\pi : S \mapsto A$ that maximizes

  $$E[r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \ldots]$$

  From any starting state in S

- Note:
  - Target function is π
  - We have no training examples of the form <s,a>
  - Our training examples are of the form <<s,a>,r>

Types of RL

- Model based vs Model free
  - **Model based**: Learn the model and use it to determine optimal policy
  - **Model free**: Derive optimal policy without learning the model

- Passive vs Active learning
  - **Passive learning**: Agent observes world and tries to determine the value of being in different states
  - **Active learning**: Agent watches and takes actions

Passive learning

- An agent has a policy π
- Executes a set of trials using π
  - Starts in s_0, has a series of state transitions until it reaches a terminal state
- Tries to determine the expected utility of being in each state

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

- Direct utility estimation is a form of supervised learning
  - Input: state
  - Output: reward to go

- Ignores an important piece of information
  - Utility values obey Bellman equations

- Misses opportunities for learning

Adaptive dynamic programming (ADP)

- Recall Bellman equations
  - $V^{\pi}(s_i) = r_i(s_i) + \gamma\Sigma_j\ P_{ij}^{\pi(s_i)} V^{\pi}(s_j)$
  - Connection between states can speed up learning
    - Do not need to consider any situation where the above constraint is violated

- Adaptive dynamic programming (ADP)
  - Learns transition probabilities, rewards from observations
  - Updates values of states

ADP Example

$\gamma = 1$

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

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

(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)→$(4,2)_{-1}$

$$\left.\begin{array}{l} P_{(1,3)(2,3)}^{r}=2/3 \\ P_{(1,3)(1,2)}^{r}=1/3 \end{array}\right\}$$ Use this information in the Bellman equation

(Passive) TD

- Temporal difference
  - Model free
- Key idea
  - Use observed transitions to adjust values of observed states so that they satisfy Bellman equations
- At each time step
  - Observe s, a, s', r
  - Update V^π after each move
  - $V^\pi(s) \rightarrow V^\pi(s) + \alpha\ (r(s) + \gamma\ V^\pi(s') - V^\pi(s))$

TD(0)

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

Learning rate

- **Thm**: If α is appropriately decreased with number of times a state is visited, then $V^\pi(s)$ converges to correct value

- α must satisfy
  - $\sum_n \alpha(n) \rightarrow \infty$
  - $\sum_n \alpha^2(n) < \infty$
- $\alpha(n) = 1/n$ satisfies conditions

Algorithm TD(λ)

(not in Russell & Norvig book)

Idea: update from the whole training sequence, not just on state transition.

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

Special cases:

$\lambda=1$: basically ADP (but use learning rate instead of explicit counts)

$\lambda=0$: TD

Intermediate choice of λ (between 0 and 1) is best

Empirically, $\lambda=0.7$ works well

Active Learning

- We are really interested in finding a good policy π

- Transition and reward model known
  - $V^*(s) = \max_a [r(s) + \gamma \sum_{s'} P(s'|s,a) V^*(s')]$

- Transition and reward model unknown:
  - Improve policy as agent executes policy

Q-Learning

- Key idea: Learn a function $Q: S \times A \to \Re$
  - Value of state action pair
  - Policy $\pi(s) = \text{argmax}_a Q(s,a)$ is optimal policy
  - $V^*(s) = \max_a Q(s,a)$

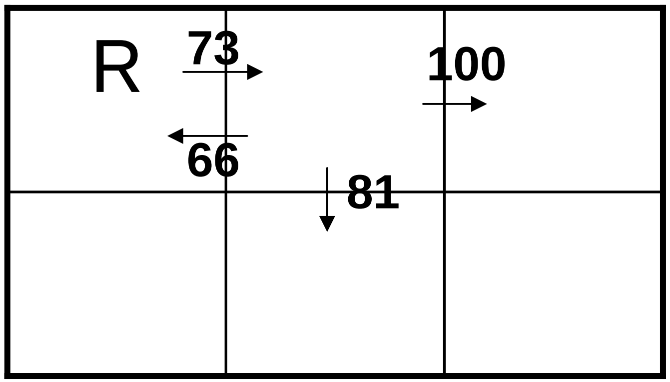
- Bellman's equation:

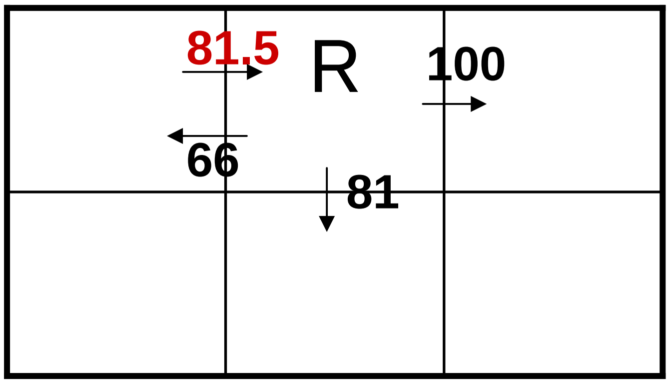
$$\boxed{Q(s_i, a) = r(s_i) + \sum_j P_{ij}^a \max_{a'} Q(s_j, a')}$$

Q-learning

- For each state *s* and action *a* initialize Q(s,a)
  - 0 or random
- Observe current state
- **Loop**
  - Select action a and execute it
  - Receive immediate award r
  - Observe new state s'
  - Update Q(a,s)
    - $Q(s,a)=Q(s,a)+\alpha(r+\gamma \max_{a'} Q(s',a')-Q(s,a))$
  - s=s'

Q-learning example





r=0 for non-terminal states

γ=0.9

α = 0.5

$$\begin{aligned} Q(s_1, a_{\text{right}}) &= Q(s_1, a_{\text{right}}) + \alpha(r + \gamma \max_{a'} Q(s_2, a') - Q(s_1, a_{\text{right}})) \\ &= 73 + 0.5(0 + 0.9\max[66, 81, 100] - 73) \\ &= 73 + 0.5(17) \\ &= 81.5 \end{aligned}$$

Q-learning

- For each state *s* and action *a* initialize Q(s,a)
  - 0 or random
- Observe current state
- **Loop**
  - **Select action a and execute it**
  - Receive immediate award r
  - Observe new state s'
  - Update Q(a,s)
    - $Q(s,a)=Q(s,a)+\alpha(r+\gamma \max_{a'} Q(s',a')-Q(s,a))$
  - s=s'

Exploration vs Exploitation

- If an agent always chooses the action with the highest value then it is exploiting
  - The learned model is not the real model
  - Leads to suboptimal results
- By taking random actions (pure exploration) an agent may learn the model
  - But what is the use of learning the correct model if you never use it?
- Need a balance between exploitation and exploration

Common exploration methods

1. In value iteration in an ADP agent:
   Optimistic estimate of utility $J^{+}(i)$

$$V^*(i) \leftarrow r(i) + \max_a f[\sum_j P_{ij}^a V^*(j), N(a,i)]$$

Exploration func $f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$

Optimistic estimate of best reward

Fixed parameter

2. Choose best action w.p. p and a random action otherwise.

3. 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 optimal Q-values if

1. Every state is visited infinitely often (due to exploration)

2. The action selection becomes greedy as time approaches infinity

3. The learning rate α is decreased fast enough but not too fast

A Triumph for Reinforcement Learning: TD-Gammon

- Backgammon player with a neural network representation of the value function:

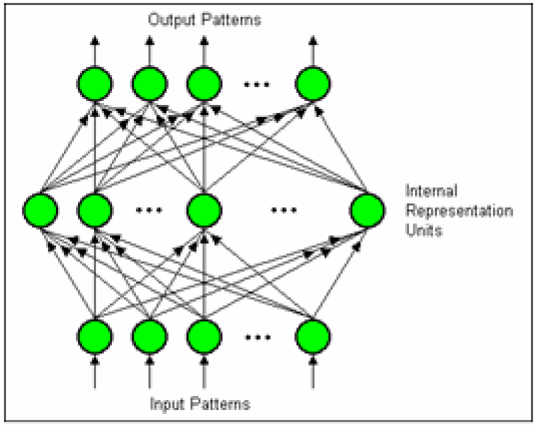


Figure 1. An illustration of the multilayer perception architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].

- Performs TD(λ) changing it's policy as it goes towards the currently predicted optimal one.

Summary

- Active vs Passive learning
- Model based vs Model free
- ADP
- TD
- Q-learning
- Exploration vs exploitation tradeoff