

Reinforcement Learning

June 21, 2005

CS 486/686

University of Waterloo

Outline

- Russell & Norvig Sect 21.1-21.3
- What is reinforcement learning
- Temporal-Difference learning
- Q-learning

Machine Learning

- Supervised Learning
 - Teacher tells learner what to remember
- Reinforcement Learning
 - Environment provides hints to learner
- Unsupervised Learning
 - Learner discovers on its own

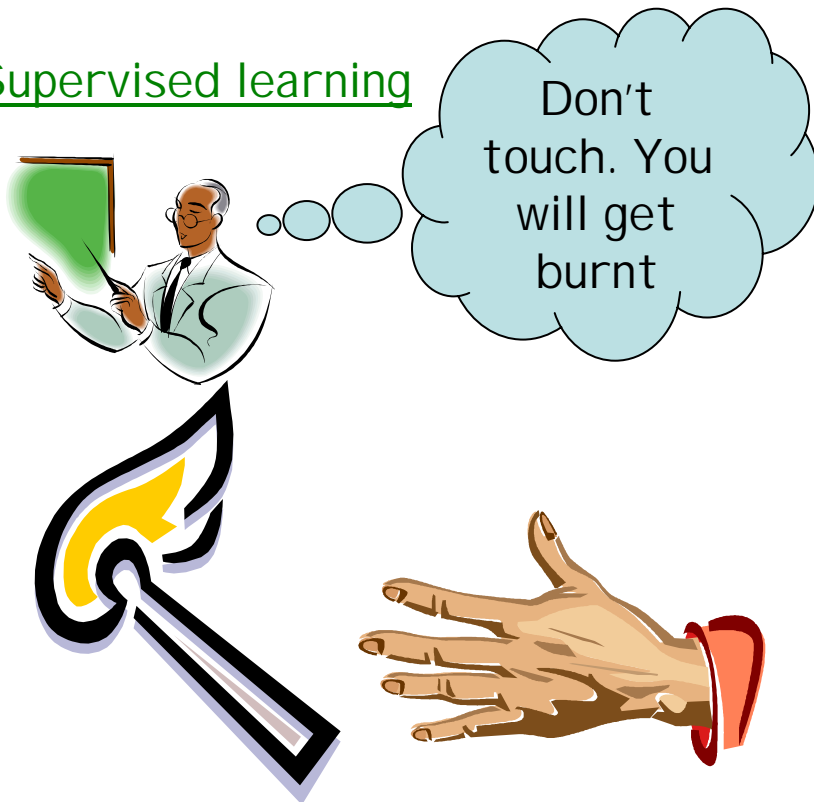
What is RL?

- Reinforcement learning is learning what to do so as to maximize a numerical reward signal
 - Learner is not told what actions to take, but must discover them by trying them out and seeing what the reward is

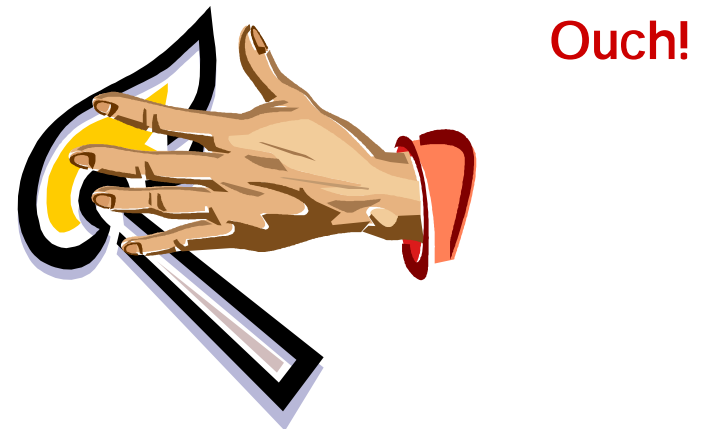
What is RL

- Reinforcement learning differs from supervised learning

Supervised learning



Reinforcement learning



Animal Psychology

- Negative reinforcements:
 - Pain and hunger
- Positive reinforcements:
 - Pleasure and food
- Reinforcements used to train animals
- Let's do the same with computers!

RL Examples

- Game playing (backgammon, solitaire)
- Operations research (pricing, vehicle routing)
- Elevator scheduling
- Helicopter control

- <http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/SuccessesOfRL>

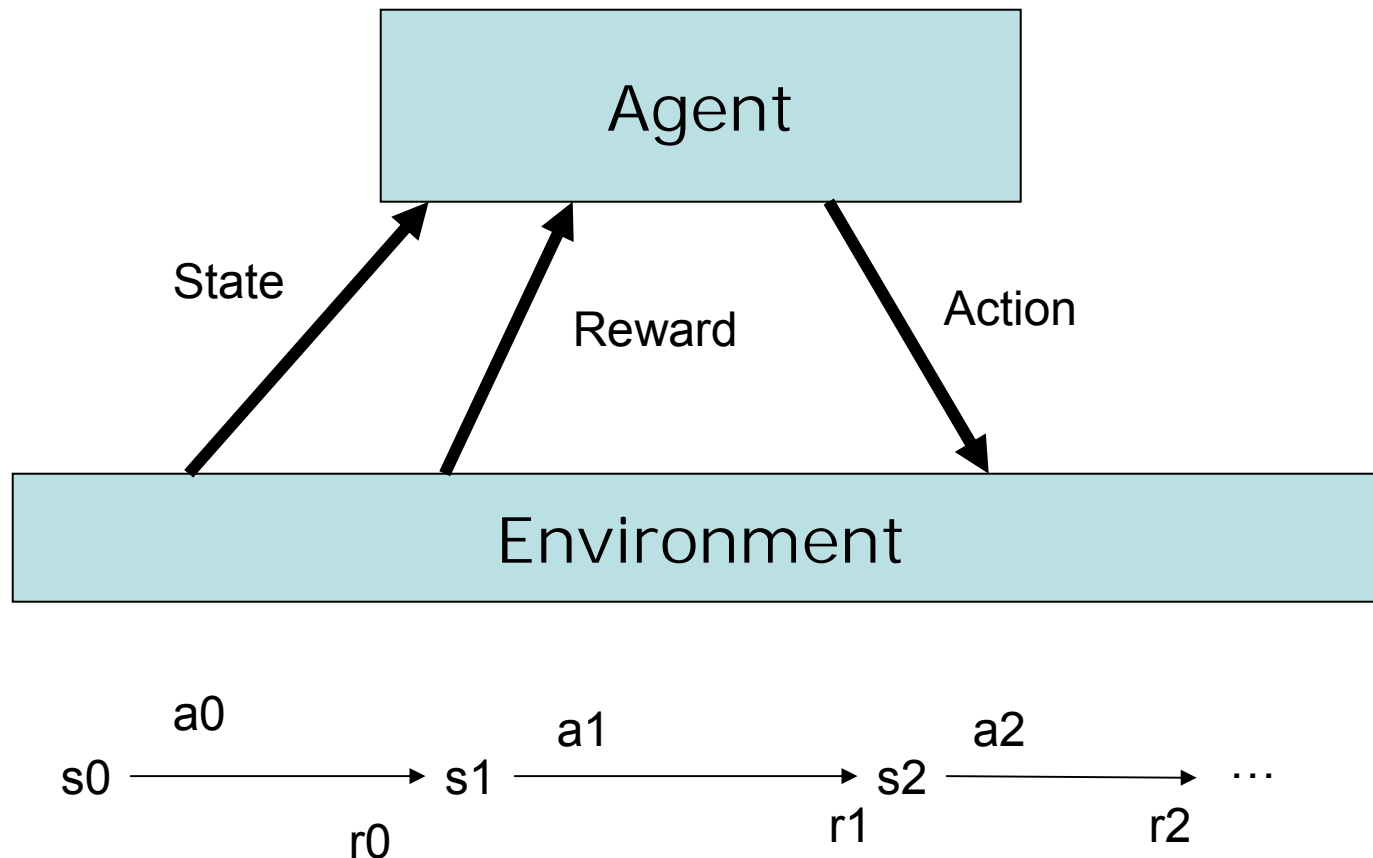
Reinforcement Learning

- Definition:
 - Markov decision process with unknown transition and reward models
- Set of states S
- Set of actions A
 - Actions may be stochastic
- Set of reinforcement signals (rewards)
 - Rewards may be delayed

Policy optimization

- Markov Decision Process:
 - Find optimal policy given transition and reward model
 - Execute policy found
- Reinforcement learning:
 - Learn an optimal policy while interacting with the environment

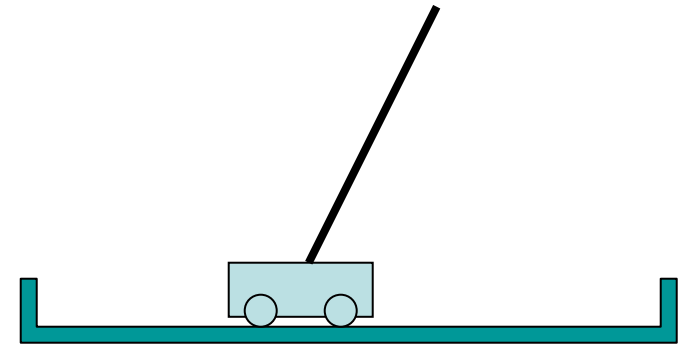
Reinforcement Learning Problem



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

Example: Inverted Pendulum

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



Problem: Find $\delta: S \rightarrow A$ that maximizes rewards

RL Characteristics

- **Reinforcements:** rewards
- **Temporal credit assignment:** when a reward is received, which action should be credited?
- **Exploration/exploitation tradeoff:** as agent learns, should it exploit its current knowledge to maximize rewards or explore to refine its knowledge?
- **Lifelong learning:** reinforcement learning

Types of RL

- **Passive vs Active learning**
 - **Passive learning**: the agent executes a fixed policy and tries to evaluate it
 - **Active learning**: the agent updates its policy as it learns
- **Model based vs model free**
 - **Model-based**: learn transition and reward model and use it to determine optimal policy
 - **Model free**: derive optimal policy without learning the model

Passive Learning

- Transition and reward model known:
 - Evaluate δ :
 - $V^\delta(s) = R(s) + \gamma \sum_{s'} \Pr(s'|s, \delta(s)) V^\delta(s')$
- Transition and reward model unknown:
 - Estimate policy value as agent executes policy: $V^\delta(s) = E_\delta[\sum_t \gamma^t R(s_t)]$
 - Model based vs model free

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

Do not know the transition probabilities

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

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

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

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

Passive ADP

- Adaptive dynamic programming (ADP)
 - Model-based
 - Learn transition probabilities and rewards from observations
 - Then update the values of the states

$\gamma = 1$

ADP Example

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

$$V^\delta(s) = R(s) + \gamma \sum_{s'} \Pr(s'|s, \delta(s)) V^\delta(s')$$

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

$$P((2,3) | (1,3), r) = 2/3$$

$$P((1,2) | (1,3), r) = 1/3$$

} Use this information in

We need to learn all the transition probabilities!

Passive TD

- Temporal difference (TD)
 - Model free
- At each time step
 - Observe: s, a, s', r
 - Update $V^\delta(s)$ after each move
 - $V^\delta(s) = V^\delta(s) + \alpha (R(s) + \gamma V^\delta(s') - V^\delta(s))$

Learning rate



Temporal difference



TD Convergence

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

- α must satisfy:
 - $\sum_t \alpha_t \rightarrow \infty$
 - $\sum_t (\alpha_t)^2 < \infty$
- Often $\alpha(s) = 1/n(s)$
 - $n(s) = \#$ of times s is visited

Active Learning

- Ultimately, we are interested in improving δ
- Transition and reward model known:
 - $V^*(s) = \max_a R(s) + \gamma \sum_{s'} \Pr(s'|s,a) V^*(s')$
- Transition and reward model unknown:
 - Improve policy as agent executes policy
 - Model based vs model free

Q-learning (aka active temporal difference)

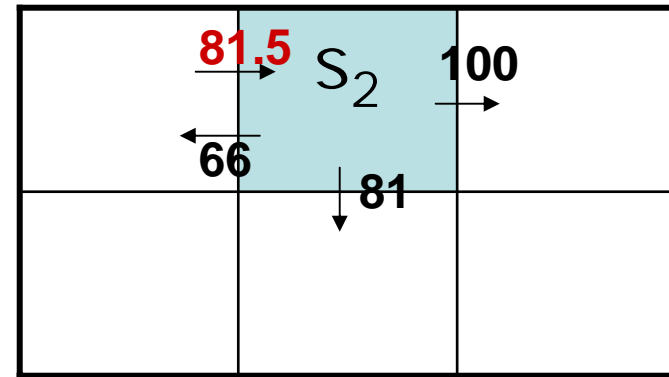
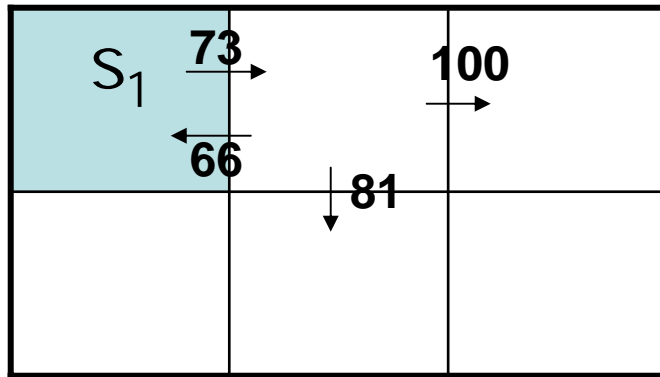
- Q-function: $Q: S \times A \rightarrow \mathcal{R}$
 - Value of state-action pair
 - Policy $\delta(s) = \operatorname{argmax}_a Q(s,a)$ is the optimal policy
- Bellman's equation:

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} \operatorname{Pr}(s'|s,a) \max_{a'} Q^*(s',a')$$

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 reward r
 - Observe new state s'
 - Update $Q(a,s)$
 - $Q(s,a) = Q(s,a) + \alpha(r(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$
 - $s=s'$

Q-learning example



$r=0$ for non-terminal states

$\gamma=0.9$

$\alpha=0.5$

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

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 reward r
 - Observe new state s'
 - Update $Q(a,s)$
 - $Q(s,a) = Q(s,a) + \alpha(r(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$
 - $s=s'$

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 a complete model if parts of it are never used?
- Need a balance between exploitation and exploration

Common exploration methods

- ϵ -greedy:
 - With probability ϵ execute random action
 - Otherwise execute best action a^*
 $a^* = \operatorname{argmax}_a Q(s,a)$
- 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
 - Every state is visited infinitely often (due to exploration)
 - The action selection becomes greedy as time approaches infinity
 - The learning rate α is decreased fast enough but not too fast

A Triumph for Reinforcement Learning: TD-Gammon

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

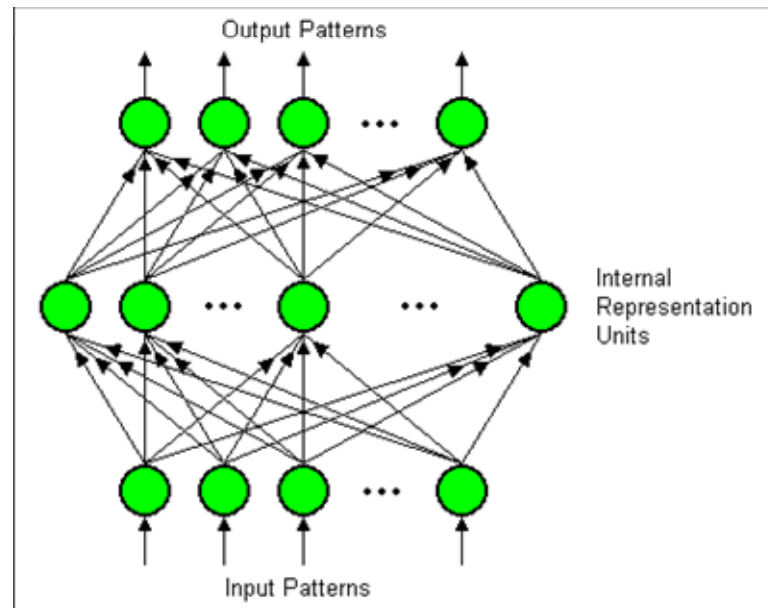


Figure 1. An illustration of the multilayer perceptron architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].

Next Class

- Machine learning
- Decision trees
- Russell and Norvig: chapter 18