# CS 536 – Montana State University

# Reinforcement Learning
# Chapter 13 of Tom Mitchell's Machine Learning Book

Neal Richter – April 24th 2006

Slides adapted from Mitchell's lecture notes and
Dr. Geehyuk Lee's Machine Learning class at ICU

# Outline

- Control learning
- Control policies that choose optimal actions
- *Q*-learning
- Convergence

Other Reinforcement Learning type methods

- Swarm & Ant World Intelligence
- Adaptive Control

# Control Learning

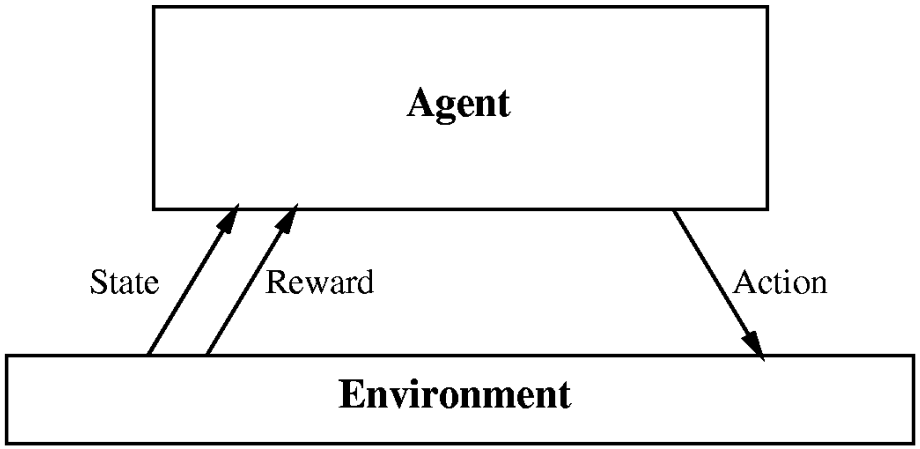
Consider learning to choose actions, e.g.,

- Robot learning to dock on battery charger
- Learning to choose actions to optimize factory output
- Learning to play Backgammon

Note several problem characteristics:

- Delayed reward
- Opportunity for active exploration
- Possibility that state only partially observable
- Possible need to learn multiple tasks with same sensors / effectors

# Reinforcement Learning Problem




$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \text{ , where } 0 \leq \gamma < 1$$

# Markov Decision Processes

Assume

- finite set of states $S$
- set of actions $A$
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward $r_t$
- and state changes to $s_{t+1}$
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
  - i.e., $r_t$ and $s_{t+1}$ depend only on *current* state and action
  - functions $\delta$ and $r$ may be nondeterministic
  - functions $\delta$ and $r$ not necessarily known to agent

## Agent's Learning Task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

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

  from any starting state in $S$
- here $0 \leq \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \rightarrow A$
- but we have no training examples of form $\langle s, a \rangle$
- training examples are of form $\langle \langle s, a \rangle, r \rangle$

## Value Function

To begin, consider deterministic worlds...

For each possible policy $\pi$ the agent might adopt, we can define an evaluation function over states

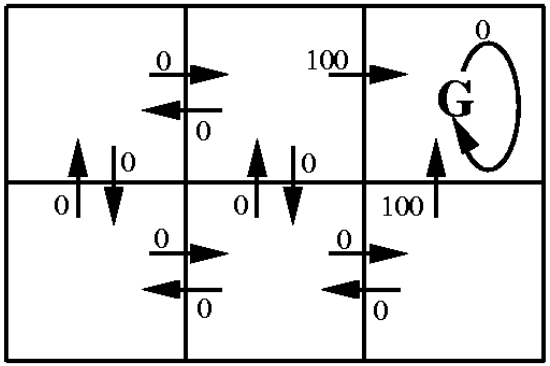
$$\begin{aligned} V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

where $r_t, r_{t+1}, \ldots$ are generated by following policy $\pi$ starting at state $s$

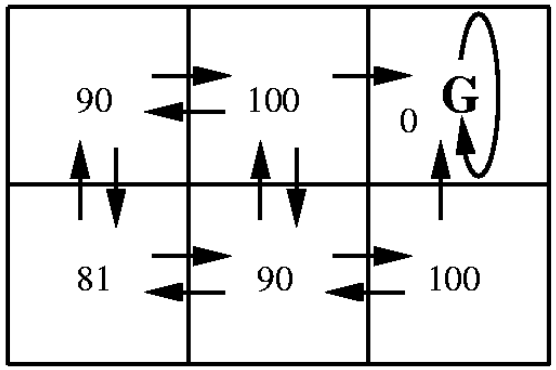
Restated, the task is to learn the optimal policy $\pi^*$

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

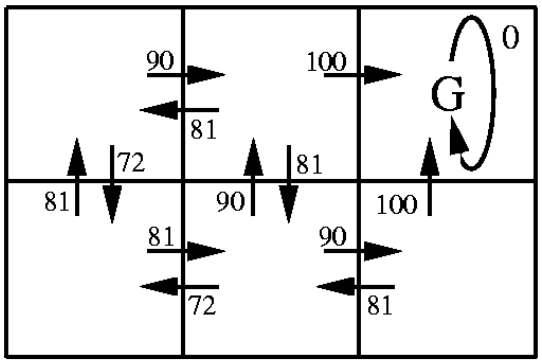
# A simple deterministic world



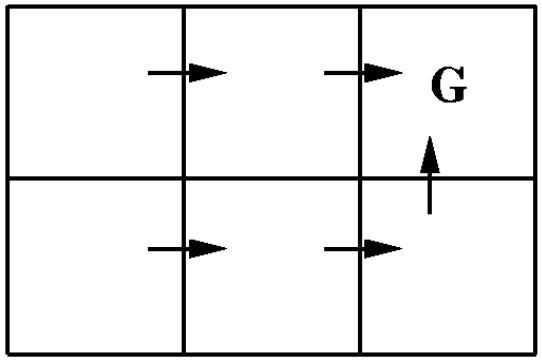

r(s, a): immediate reward




$V^*(s)$: value function




Q(s, a): Q-function




One optimal policy

## What to Learn

We might try to have agent learn the evaluation function $V^{\pi^*}$ (which we write as $V^*$)

It could then do a lookahead search to choose best action from any state $s$ because

$$\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \Re$
- But when it doesn't, it can't choose actions this way

## *Q*-Function

Define new function very similar to $V^*$

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

$$\pi^*(s) = \operatorname{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \operatorname{argmax}_a Q(s,a)$$

$Q$ is the evaluation function the agent will learn

If we learn Q **DIRECTLY**, we have no need to know the *r* and *delta* functions.

## Training Rule to Learn APPROXIMATE *Q*

Note $Q$ and $V^*$ closely related:

$$V^*(s) = \max_{a'} Q(s, a')$$

Which allows us to write $Q$ recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Nice! Let $\hat{Q}$ denote learner's current approximation to $Q$. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

where $s'$ is the state resulting from applying action $a$ in state $s$

# *Q* Learning for Deterministic Worlds

For each $s, a$ initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state $s$

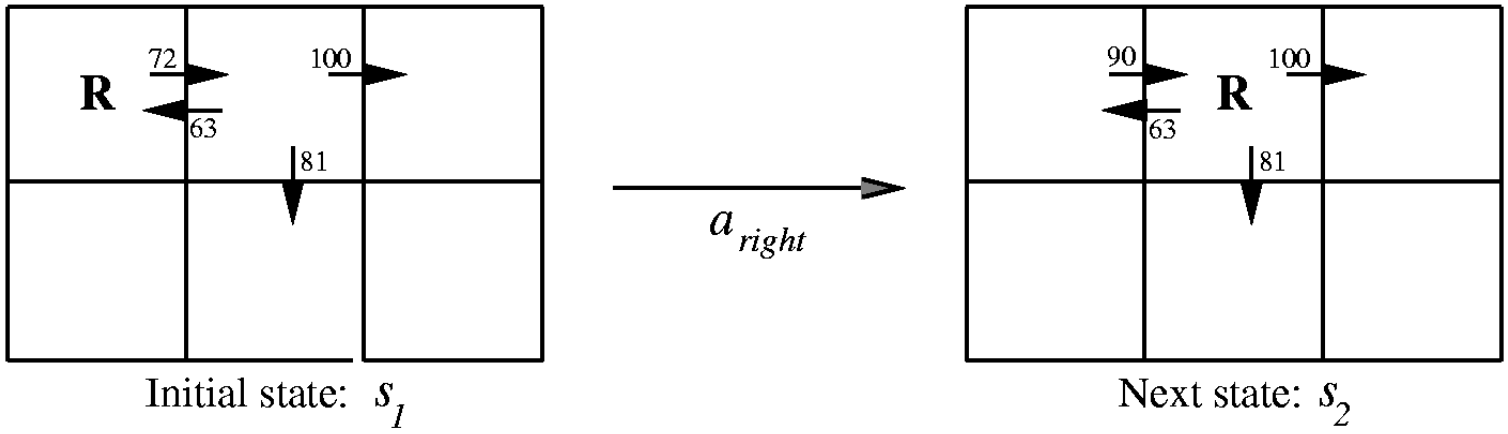
Do forever:

- Select an action $a$ and execute it
- Receive immediate reward $r$
- Observe the new state $s'$
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

# Updating *Q*-hat




$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$
$$\leftarrow 0 + 0.9 \ \max\{63, 81, 100\} \leftarrow 90$$

Notice if rewards non-negative, then

$$(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$
$$(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

# *Q*-hat converges to *Q*

Q-hat converges to Q under the following conditions:

- System is Deterministic Markov Decision Process
- Reward values are bounded
- Agent must use/choose all action-state pairs
    - All paths in graph must be taken a "high" number of times

- See book for gory details of proofs

# Explore or Exploit?

In Q-learning algorithm, there was no mention about how to select an action. A few alternatives are

- Choose an action a that maximize $\hat{Q}(s, a)$
- Give each action equal opportunity
- Select an action at probability $P(a_i|s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$

Q1. Pros and cons of each strategy?
Q2. What is the precondition for the theorem that ensures the convergence of $\hat{Q}(s, a)$?

## Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine $V, Q$ by taking expected values

$$
\begin{aligned}
V^{\pi}(s) &\equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] \\
&\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}] \\
Q(s,a) &\equiv E[r(s,a) + \gamma V^*(\delta(s,a))] \\
&\equiv E[r(s,a)] + \gamma E[V^*(\delta(s,a))] \\
&\equiv E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) V^*(s') \\
&\equiv E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')
\end{aligned}
$$

## Nondeterministic Case

- $Q$ learning is now generalized to nondeterministic worlds.
- But, do we have the guarantee that the training will converge?
- No, because $r$ can change on every iteration.
- Alter training rule to

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a'} \hat{Q}_{n-1}(s',a')]$$

where

$$\alpha_n = \frac{1}{1 + visits_n(s,a)}$$

- It can be proven that $\hat{Q}$ converges to $Q$ [Watkins and Dayan, 1992]

Key Idea: Revisions to Q-hat are made more gradually here. Chose of alternate alpha-n functions are available.

# Swarm Intelligence & Ant Colony Models

Swarm intelligence (SI) is an artificial intelligence technique based aroun d the study of collective behaviour in decentralised, self-organised, syste ms. SI systems are typically made up of a population of simple agents in teracting locally with one another and with their environment to produce e mergence behavior.

Ant Colony Optimization:
The ACO algorithm is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.

Ants (initially) wander randomly, and upon finding food return to their colo ny while laying down pheromone trails. If other ants find such a path, the y are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. Over time the phe romone trails evaporate, and a stable near-optimal or optimal path can e merge.

## Adaptive Control

Adaptive Control is a very mature discipline that deals with (among other things) estimating transition probabilities of unknown Markov Models.

This paper relates Q Learning to a common method in Adaptive Optimal Control problems.

Reinforcement Learning is Direct Adaptive Optimal Control (1992)
Richard S. Sutton, Andrew G. Barto, Ronald J. Williams

There also exist stochastic variants of Adaptive Optimal Control methods, and these are likely the same as some non-deterministic RL/Q methods.