# COMP9414: Artificial Intelligence

# Lecture 9b: Reinforcement Learning

Wayne Wobcke

e-mail:w.wobcke@unsw.edu.au



## This Lecture

- Reinforcement Learning vs Supervised Learning
- Models of Optimality
- Exploration vs Exploitation
- Temporal Difference Learning
- Q-Learning



## Types of Learning

- Supervised Learning
  - Agent is presented with examples of inputs and their target outputs, and must learn a function from inputs to outputs that agrees with the training examples and generalizes to new examples
- Reinforcement Learning
  - Agent is not presented with target outputs for each input, but is periodically given a reward, and must learn to maximize (expected) rewards over time
- Unsupervised Learning
  - Agent is only presented with a series of inputs, and must find useful patterns in these inputs



## Supervised Learning

- Given a training set and a test set, each consisting of a set of items for each item in the training set, a set of features and a target output
- Learner must learn a model that can predict the target output for any given item (characterized by its set of features)
- Learner is given the input features and target output for each item in the training set
  - Items may be presented all at once (batch) or in sequence (online)
  - Items may be presented at random or in time order (stream)
  - Learner **cannot** use the test set **at all** in defining the model
- Model is evaluated by its performance on predicting the output for each item in the test set

# Learning Actions

Supervised learning can be used to learn actions from a training set of situation-action pairs (called Behavioural Cloning)

However, there are many applications for which it is difficult, inappropriate, or even impossible to provide a "training set"

- Optimal control
  - Mobile robots, pole balancing, flying a helicopter
- Resource allocation
  - Job shop scheduling, mobile phone channel allocation
- Mix of allocation and control
  - Elevator control, Backgammon





# Reinforcement Learning Framework

- Agent interacts with its environment
- There is a set $S$ of *states* and a set $A$ of *actions*
- At each time step $t$, the agent is in some state $s_t$ and must choose an action $a_t$, whereupon it goes into state $s_{t+1} = \delta(s_t, a_t)$ and receives reward $r(s_t, a_t)$
- In general, $r()$ and $\delta()$ can be multi-valued, with a random element
- The aim is to find an optimal *policy* $\pi : S \rightarrow A$ which maximizes the cumulative reward





# Models of Optimality

Is a fast nickel worth a slow dime?

Finite horizon reward $\sum_{i=0}^{h} r_{t+i}$

Average reward $\lim_{h\to\infty} \frac{1}{h} \sum_{i=0}^{h-1} r_{t+i}$

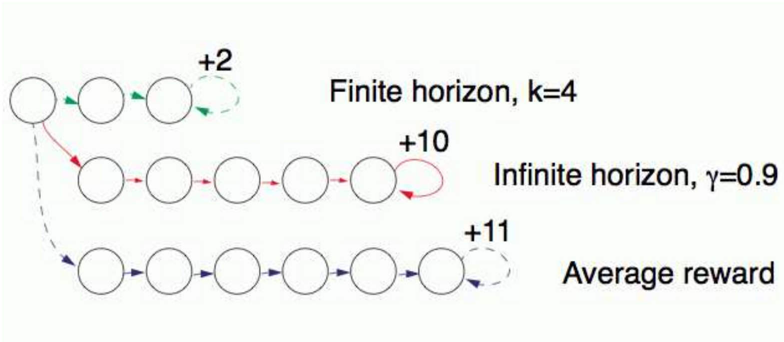
Infinite discounted reward $\sum_{i=0}^{\infty} \gamma^i r_{t+i}, \quad 0 \leq \gamma < 1$

- Finite horizon reward is simple computationally
- Infinite discounted reward is easier for proving theorems
- Average reward is hard to deal with, because can't sensibly choose between small reward soon and large reward very far in the future





# Comparing Models of Optimality



# Environment Types

Environments can be

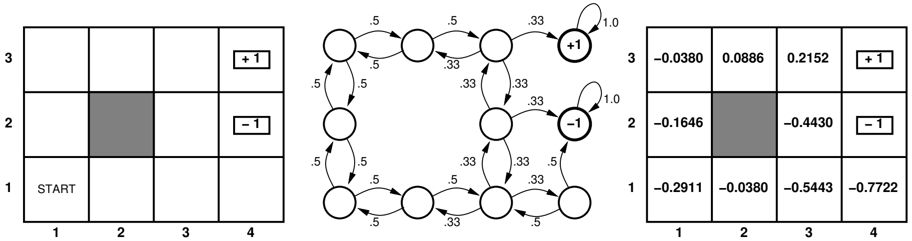
- Passive and stochastic
- Active and deterministic (Chess)
- Active and stochastic (Backgammon)



# Value Function

For each state $s \in S$, let $V^*(s)$ be the maximum discounted reward obtainable from $s$



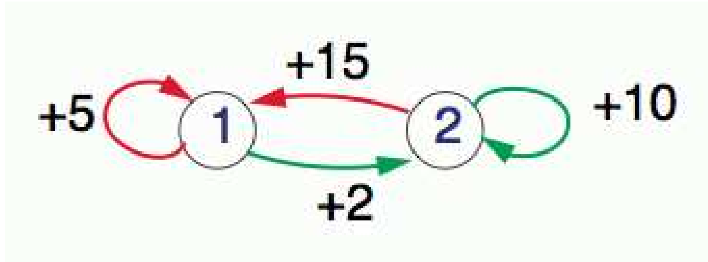

(a) (b) (c)

The optimal value function determines the optimal policy



# Example: Delayed Rewards





# Calculation

**Theorem:** In a deterministic environment, for an optimal policy, the value function $V^*$ satisfies the Bellman equations: $V^*(s) = r(s,a) + \gamma V^*(\delta(s,a))$ where $a = \pi^*(s)$ is the optimal action at state $s$.

Let $\delta^*(s)$ be the transition function for $\pi^*(s)$ and suppose $\gamma = 0.9$

1. Suppose $\delta^*(s_1) = s_1$. Then $V^*(s_1) = 5 + 0.9V^*(s_1)$ so $V^*(s_1) = 50$
   Suppose $\delta^*(s_2) = s_2$. Then $V^*(s_2) = 10 + 0.9V^*(s_2)$ so $V^*(s_2) = 100$
2. Suppose $\delta^*(s_1) = s_2$. Then $V^*(s_1) = 2 + 0.9V^*(s_2)$ so $V^*(s_1) = 92$
   Suppose $\delta^*(s_2) = s_2$. Then $V^*(s_2) = 10 + 0.9V^*(s_2)$ so $V^*(s_2) = 100$
3. Suppose $\delta^*(s_1) = s_2$. Then $V^*(s_1) = 2 + 0.9V^*(s_2)$ so $V^*(s_1) = 81.6$
   Suppose $\delta^*(s_2) = s_1$. Then $V^*(s_2) = 15 + 0.9V^*(s_1)$ so $V^*(s_2) = 88.4$

So 2 is the optimal policy

# Exploration/Exploitation Tradeoff

Most of the time, the agent should choose the “best” action

However, in order to ensure the optimal strategy can be learned, the agent must occasionally choose a different action, e.g.

- Choose a random action 5% of the time, or
- Use a Boltzmann distribution to choose the next action

$$P(a) = \frac{e^{\hat{V}(a)/T}}{\sum_{b \in A} e^{\hat{V}(b)/T}}$$



# K-Armed Bandit Problem


The special case of an active stochastic environment with only one state is called a K-Armed Bandit Problem, because it is like being in a room with several (friendly) slot machines, for a limited time, and trying to collect as much money as possible

Each action (slot machine) provides a different average reward



# Temporal Difference Learning

TD(0) [also called AHC, or Widrow-Hoff Rule]

$$\hat{V}(s) \leftarrow \hat{V}(s) + \eta\,[r(s,a) + \gamma \hat{V}(\delta(s,a)) - \hat{V}(s)]$$

($\eta$ = learning rate)

The (discounted) value of the next state, plus the immediate reward, is used as the target value for the current state

A more sophisticated version, called TD($\lambda$), uses a weighted average of future states



# Q-Learning

For each $s \in S$, let $V^*(s)$ be the maximum discounted reward obtainable from $s$, and let $Q(s,a)$ be the discounted reward available by first doing action $a$ and then acting optimally

Then the optimal policy is

$$\pi^*(s) = \arg\max_a Q(s,a)$$

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

Then $$V^*(s) = \max_a Q(s,a)$$

so $$Q(s,a) = r(s,a) + \gamma \max_b Q(\delta(s,a),b)$$

This allows iterative approximation of $Q$ by

$$\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \max_b \hat{Q}(\delta(s,a),b)$$

# Theoretical Results

**Theorem:** Q-learning will eventually converge to the optimal policy, for any deterministic Markov Decision Process, assuming an appropriately randomized strategy.

(Watkins & Dayan 1992)

**Theorem:** TD-learning will also converge, with probability 1.

(Sutton 1988, Dayan 1992, Dayan & Sejnowski 1994)






# Limitations of Theoretical Results

- Delayed reinforcement
  - Reward resulting from an action may not be received until several time steps later, which also slows down the learning
- Search space must be finite
  - Convergence is slow if the search space is large
  - Relies on visiting every state infinitely often
- For "real world" problems, can't rely on a lookup table
  - Need to have some kind of generalization (e.g. TD-Gammon)






# Summary

- Reinforcement Learning is an active area of research
- Mathematical results (more than in other areas of AI)
- Need to have an appropriate representation
- Future algorithms which choose their own representations?
- Many practical applications