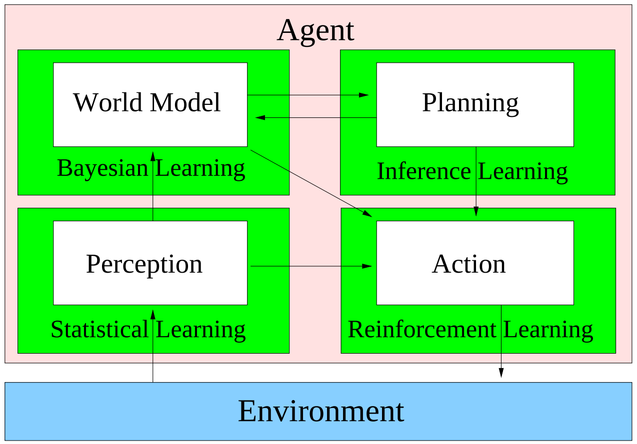
COMP3411/COMP9414: Artificial Intelligence

9b. Reinforcement Learning

Outline

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

Learning Agents

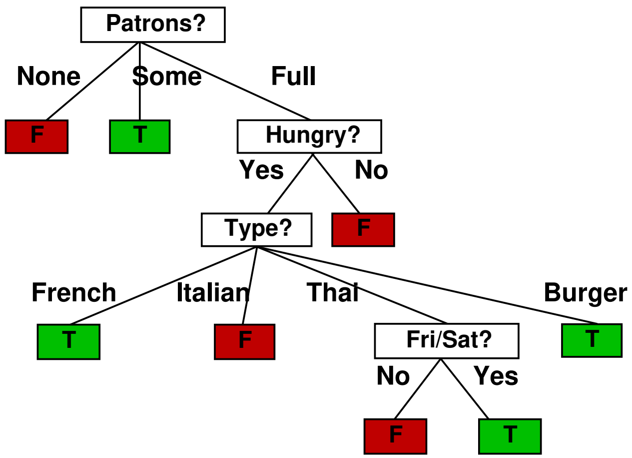


Supervised Learning

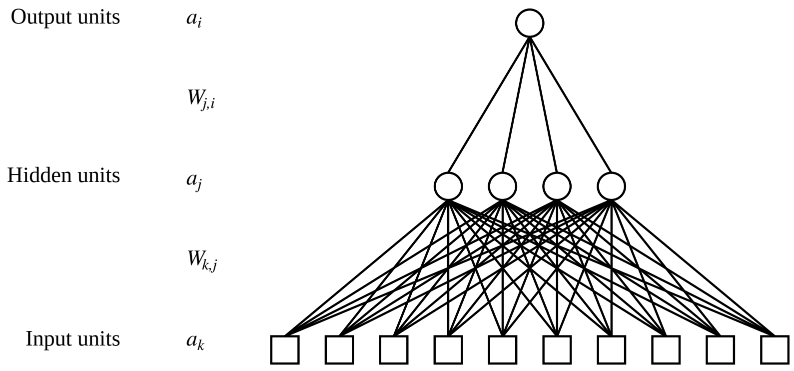
Recall: Supervised Learning

- We have a training set and a test set, each consisting of a set of examples. For each example, a number of input attributes and a target attribute are specified.
- The aim is to predict the target attribute, based on the input attributes.
- Various learning paradigms are available:
  - Decision Trees
  - Neural Networks
  - .. others ..

Decision Tree



Neural Network



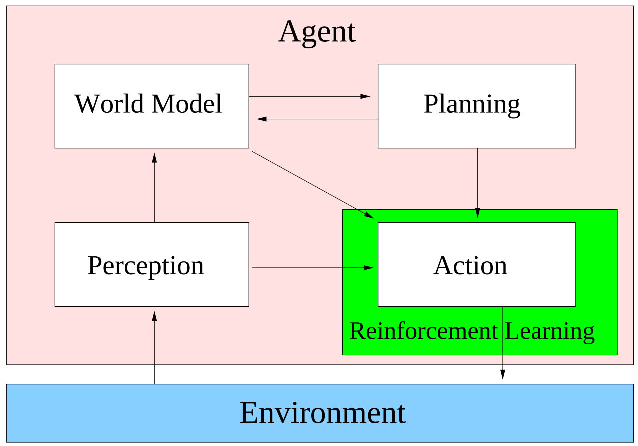
Learning of Actions

Supervised Learning can also be used to learn Actions, if we construct a training set of situation-action pairs (called Behavioral 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 Agent



Reinforcement Learning Framework

- An 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. It 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 will maximize the cumulative reward.

Models of optimality

Is a fast nickel worth a slow dime?

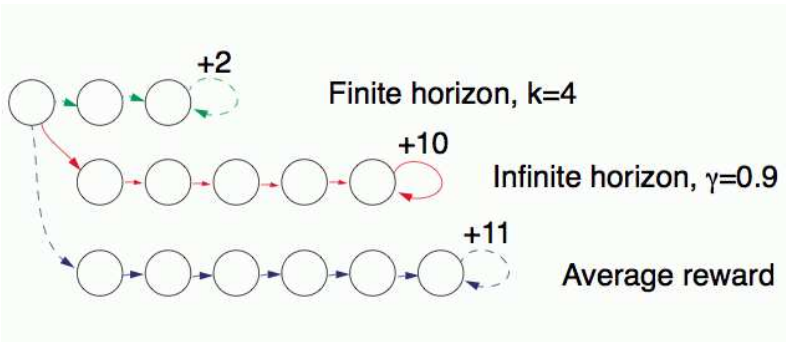
Finite horizon reward $\sum_{i=0}^{h} r_{t+i}$

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

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

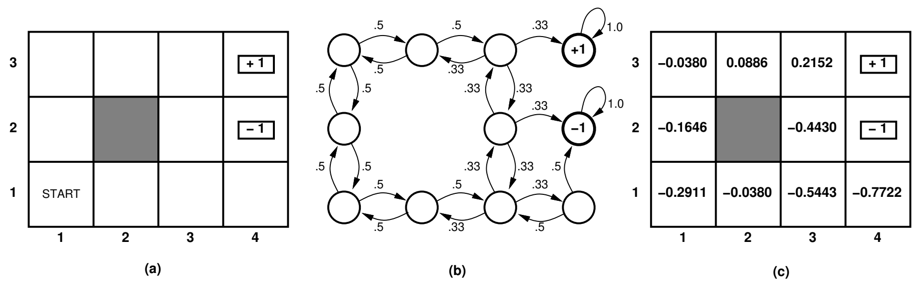
- Finite horizon reward is simple computationally
- Infinite discounted reward is easier for proving theorems
- Average reward is hard to deal with, because can't sensibly choose between small reward soon and large reward very far in the future.

Comparing Models of Optimality



Value Function

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



Learning this Value Function can help to determine the optimal strategy.

Environment Types

Environments can be:

- passive and stochastic (as in previous slide)
- active and deterministic (chess)
- active and stochastic (backgammon)

K-Armed Bandit Problem

The special case of an active, stochastic environment with only one state is called the 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.

Exploration / Exploitation Tradeoff

Most of the time we should choose what we think is the best action.

However, in order to ensure convergence to the optimal strategy, we must occasionally choose something different from our preferred action, e.g.

- choose a random action 5% of the time, or
- use a Bolzmann distribution to choose the next action:

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

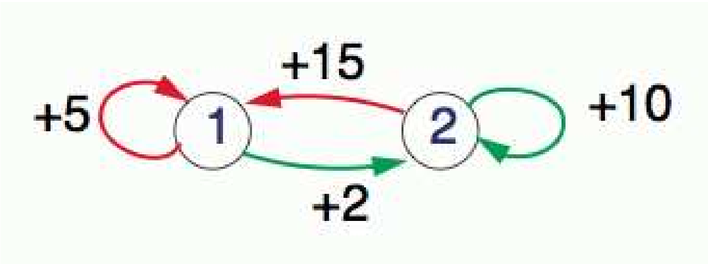
Exploration / Exploitation Tradeoff

I was born to try...

But you've got to make choices
Be wrong or right
Sometimes you've got to sacrifice the things you like.

- Delta Goodrem

Delayed Reinforcement



Temporal Difference Learning

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

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

(η = 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(λ), 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

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

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

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

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

which allows us to iteratively approximate Q by

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

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, we can't rely on a lookup table
  - need to have some kind of generalisation (e.g. TD-Gammon)