REINFORCEMENT LEARNING

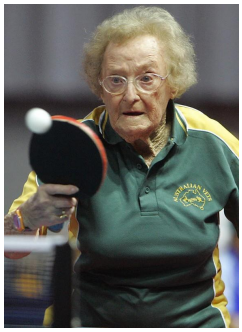
Today

- Reading
  - AIMA 17.1-17.3 (Markov Decision Processes)
  - AIMA 21.1-21.3 (Reinforcement Learning)
- Goals
  - Markov decision processes
  - (Reinforcement Learning)

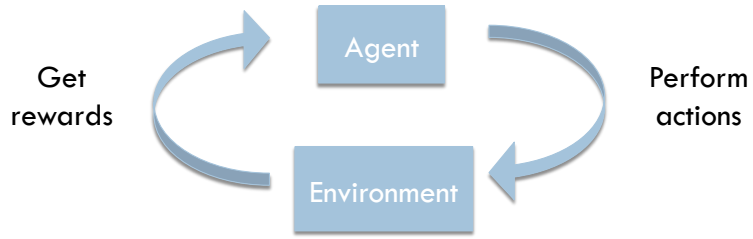
Machine Learning

- Supervised learning:
  - Learning with labels
  - Decision trees, SVMs, neural networks, perceptrons, linear regression, logistic regression, naïve Bayes, etc.
- Unsupervised Learning:
  - Learning without labels
  - K-means, agglomerative, divisive, model-based clustering (i.e. using Expectation Maximization)
- Reinforcement learning:
  - Learning with rewards
  - Robots, autonomous vehicles

Reinforcement Learning



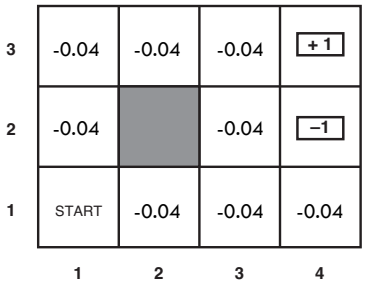
Reinforcement Learning

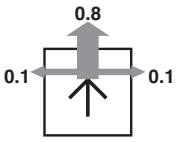


Learning problem: Given a set of observed (actions, rewards) pairs, learn how to act so as to maximize expected rewards

Learning to walk

Markov Decision Processes





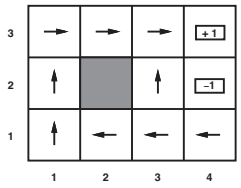
- In a deterministic environment, a solution would be [UP, UP, RIGHT, RIGHT, RIGHT]
- In our stochastic environment, the probability of reaching goal state +1 given this sequence of actions is only 0.33

Markov Decision Process

- "A sequential decision problem for a fully observable, stochastic environment with a Markovian transition function and additive rewards is called a Markov decision process"
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A stochastic transition function $T(s, a, s') = p(s' \mid s, a)$
  - A reward function $R(s)$

Solution to an MDP

- A policy is a function π: S ⟶ A that specifies what action the agent should take in any given state.
- *Executing a policy can give rise to many action sequences!*
- How can we determine the quality of a policy?



Utility

- Utility is an internal measure of an agent's success.
  - Agent's own internal performance measure
  - Surrogate for success or happiness
- The utility is a function of the rewards:

$$U([s_0, s_1, s_2, \ldots]) = \gamma^0 R(s_0) + \gamma^1 R(s_1) + \gamma^2 R(s_2) + \ldots$$
$$= \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

discount factor

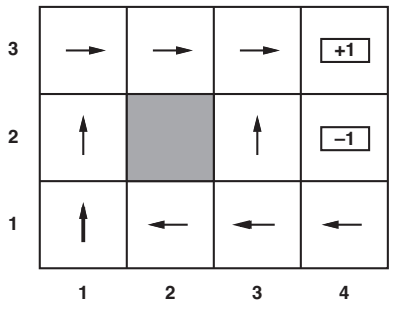
Optimal policy

- The optimal policy π* has the highest *expected* utility

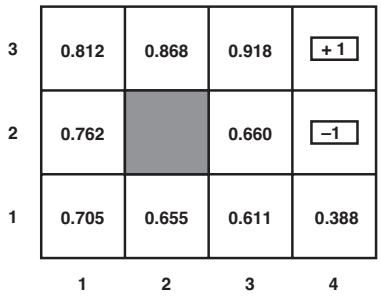
$$\pi^* = \text{argmax}_\pi \, U^\pi(s)$$

- The solution to an MDP is some optimal policy π*
  - Solution to a deterministic search problem: a sequence of actions
  - Solution to a non-deterministic sequential decision process: a function

Gridworld Example

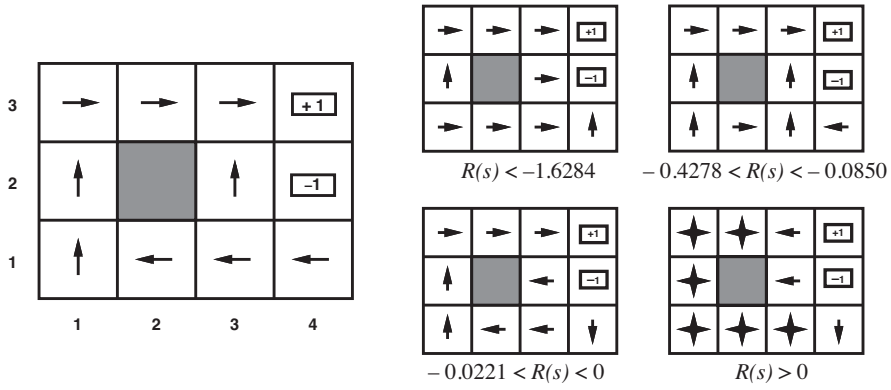


Optimal policy



Utilities

Markov Decision Processes



Solving MDPs

- Value Iteration Algorithm
  - Given an MDP, recursively formulate the utility of starting in a state s

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Bellman Equation

  - Suggests an iterative algorithm:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U_i(s')$$

  - Once we have U(s) for all states s, we can construct the optimal policy

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Solving MDPs

- Policy Iteration Algorithm
  - Policy evaluation: Given π_i compute U_i

$$U_i(s) = U^{\pi_i}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

  - Policy improvement: Given U_i compute π_{i+1}

$$\pi_{i+1}(s) = \text{argmax}_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

  - Repeat

Reinforcement Learning

- What happens if we don't have the whole MDP?
  - We know the states and actions
  - We don't know the transition function or reward function
- We're only allowed to sample from the MDP
  - Can observe experiences (s, a, r, s')
  - Need to perform actions to generate new experiences
- This is Reinforcement Learning (RL)
  - Sometimes called Approximate Dynamic Programming (ADP)

Learning Utility Functions

- Passive Reinforcement Learning (fixed policy)
  - Direct utility estimation
- Active Reinforcement Learning (unknown policy)
  - Q-learning Algorithm
  - SARSA
  - TD learning