Reinforcement Learning

What is reinforcement learning?

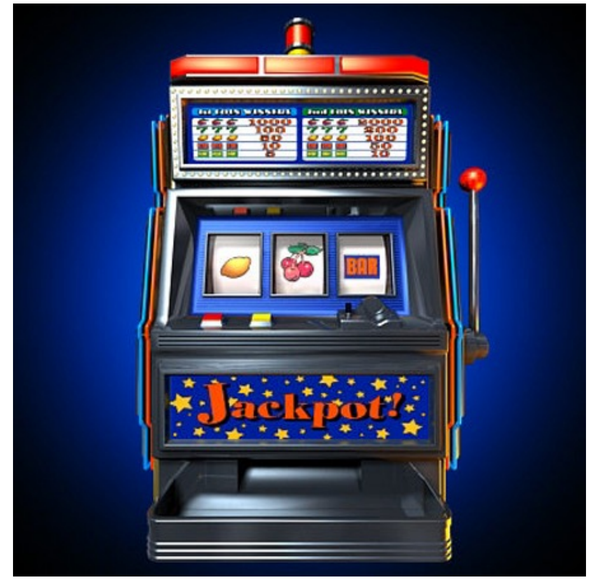
- Three machine learning paradigms:
  - Supervised learning
  - Unsupervised learning (overlaps w/ data mining)
  - Reinforcement learning
- In reinforcement learning, the agent receives incremental pieces of feedback, called rewards, that it uses to judge whether it is acting correctly or not.

Examples of real-life RL

- Learning to play chess.
- Animals (or toddlers) learning to walk.
- Driving to school or work in the morning.

- **Key idea**: Most RL tasks are *episodic*, meaning they repeat many times.
  - So unlike in other AI problems where you have one shot to get it right, in RL, it's OK to take time to try different things to see what's best.

n-armed bandit problem



- You have n slot machines.
- When you play a slot machine,
  it provides you a reward (negative or positive) according to some fixed probability distribution.
- Each machine may have a different probability distribution, and you don't know the distributions ahead of time.
- You want to maximize the amount of reward (money) you get.
- In what order and how many times do you play the machines?

RL problems

- Every RL problem is structured similarly.
- We have an ***environment***, which consists of a set of ***states***, and ***actions*** that can be taken in various states.
  - Environment is often stochastic (there is an element of chance).
  - Environment can be fully or partially observable (here, we will focus on fully observable).
- Our RL agent wishes to learn a ***policy***, π, a function that maps states to actions.
  - $\pi(s)$ tells you what action to take in a state s.

What is the goal in RL?

- In other AI problems, the "goal" is to get to a certain state. Not in RL!
- A RL environment gives feedback every time the agent takes an action. This is called a ***reward***.
  - Rewards are usually numbers.
  - Goal: Agent wants to maximize the amount of reward it gets over time.
  - Critical point: Rewards are given by the environment, not the agent.

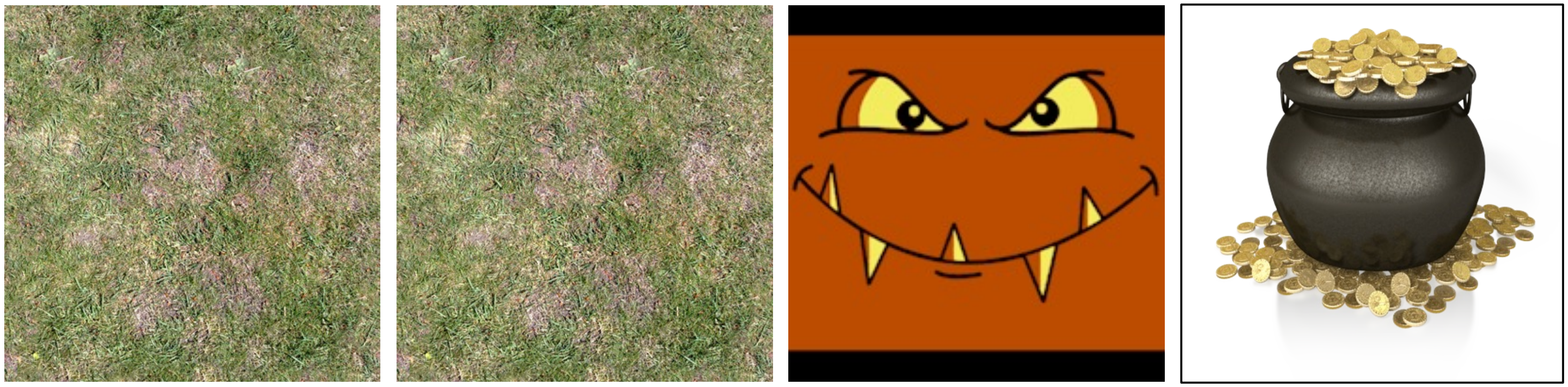
Mathematics of rewards

- Assume our rewards are r_0, r_1, r_2, …
- What expression represents our total rewards?
- How do we maximize this? Is this a good idea?
- Use discounting: at each time step, the reward is discounted by a factor of γ (called the discount rate).
- Future rewards from time t =

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Markov Decision Processes

- An MDP has a set of states, S, and a set of actions, A(s), for every state s in S.
- An MDP encodes the probability of transitioning from state s to state s' on action a: **P(s' | s, a)**
- RL also requires a reward function, usually denoted by **R(s, a, s')** = reward for being in state s, taking action a, and arriving in state s'.
- An MDP is a Markov chain that allows for outside actions to influence the transitions.



- Grass gives a reward of 0.
- Monster gives a reward of -5.
- Pot of gold gives a reward of +10 (and ends game).
- Two actions are always available:
  - Action A: 50% chance of moving right 1 square,
50% chance of staying where you are.
  - Action B: 50% chance of moving right 2 squares,
50% chance of moving left 1 square.
  - Any movement that would take you off the board moves you as far in that direction as possible or keeps you where you are.

Policies and value functions

- Almost all RL algorithms are based around computing, estimating, or learning ***policies*** and ***value functions***.
- A policy (usually π) is a function from states to actions that tells you what action you should do in each state.
- A value function represents the ***expected future reward*** from either a state, or a state-action pair.
  - $V^{\pi}(s)$: If we are in state s, and follow policy π, what is the total future reward we will see, on average?
  - $Q^{\pi}(s, a)$: If we are in state s, and take action a, then follow policy π, what is the total future reward we will see, on average?

Optimal policies

- Given an MDP, there is always a "best" policy, called π^*.
- The point of RL is to discover this policy by employing various algorithms.
  - Some algorithms can use sub-optimal policies to discover π^*.
- We denote the value functions corresponding to the optimal policy by $V^*(s)$ and $Q^*(s, a)$.