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.

Value functions

- Almost all RL algorithms are based around learning ***value functions***.
- A value function estimates 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

- There is always a "best" policy, called π^*.
- The point of RL is to discover this policy by employing various algorithms.
- We denote the value functions corresponding to the optimal policy by $V^*(s)$ and $Q^*(s, a)$.

Bellman equations

- The V(s) and Q(s, a) functions, always satisfy certain recursive relationships for any MDP.
- These relationships, in the form of equations, are called Bellman equations.

Recursive relationship of V and Q:

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

The average future rewards from a state s is equal to the average future rewards of whatever the best action is from that state.

$$Q^*(s,a) = \sum_{s'} P(s' \mid s, a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

The average future rewards obtained by taking an action from a state is the weighted average of the average future rewards from the new state.

Bellman equations

$$V^*(s) = \max_a \sum_{s'} P(s' \mid s, a)\big[R(s, a, s') + \gamma V^*(s')\big]$$

$$Q^*(s, a) = \sum_{s'} P(s' \mid s, a)\big[R(s, a, s') + \gamma \max_{a'} Q^*(s', a')\big]$$

- Most RL algorithms use these equations in various ways to estimate V* or Q*. An optimal policy can be derived from either V* or Q*.

RL algorithms

- A main categorization of RL algorithms is whether or not they require a full model of the environment.
- In other words, do we know P(s' | s, a) and R(s, a, s') for all combinations of s, a, s'?
  - If we have this information (uncommon in the real world), we can compute V* or Q* directly.
  - If we don't have this information, we can estimate V* or Q* from experience or simulations.

Value iteration

- ***Value iteration*** is an algorithm that computes an optimal policy, given a full model of the environment.
- Algorithm is derived directly from the Bellman equation (usually for V*, but can use Q* as well).
- Value iteration maintains a table of V values, one for each state. Each value V[s] eventually converges to the true value V*(s).

Value iteration

Initialize V arbitrarily, e.g., V[s] = 0 for all states s.
Repeat
  for each state s:
    $V_{\text{new}}[s] \leftarrow \max_a \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V[s']\right]$
  $V \leftarrow V_{\text{new}}$ (copy new table over old)
until the maximum difference in new and old values is smaller than some threshold
Output a policy π where $\pi(s) = \text{argmax}_a \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V^*(s')\right]$

- 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.