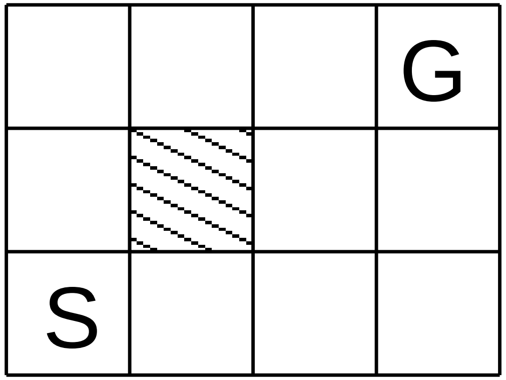
# REINFORCEMENT LEARNING

## Overview

- Last lecture looked at inductive learning
  - How to learn rules given examples of decisions.
- Supervised learning = examples of correct behavior.
- Often we don't have such examples.
- Just know when we succeed or fail.
- This is the domain of *reinforcement learning* (RL).

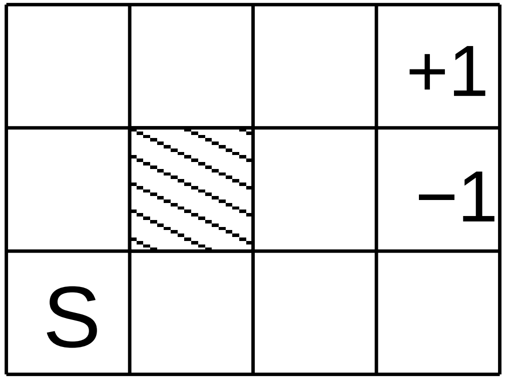
# Making complex decisions

- Before we look at RL, we need to look at sequential decision making.
- (We have seen this before).



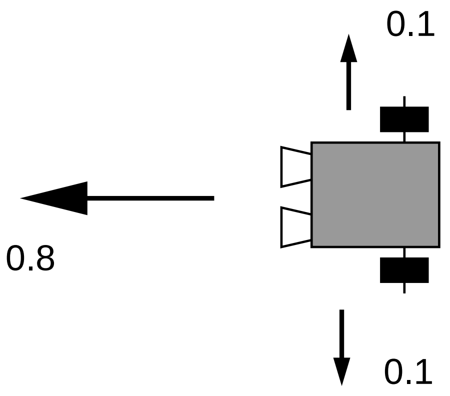
- To get from the start point (S) to the goal (G), an agent needs to repeatedly make a decision about what to do.

- Here we exchange the notion of a goal for the notion of a reward in specific states:



- And the action model gets more complex. Now actions are non-deterministic.

- If the agent chooses to move in some direction, there is a probability of 0.8 it will move that way.

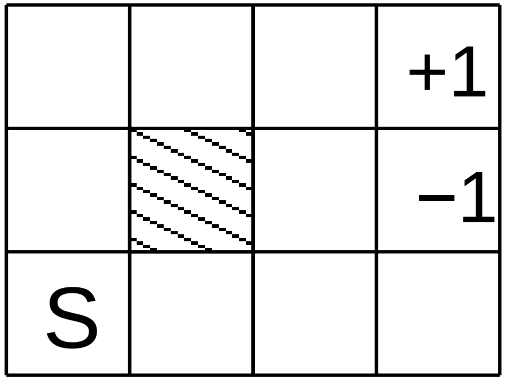


- Probability of 0.2 it will move in the perpendicular direction.
- If the agent hits a wall, it doesn't move.

- This is an approximation to how a robot moves.

- Why?

- If the agent goes $\{Up, Up, Right, Right, Right\}$



- It will get to the goal with probability $0.8^5 = 0.32768$ doing what it expects/hopes to do.

- It can also reach the goal going around the obstacle the other way, with probability = $0.1^4 \times 0.8$.
- Total probability of reaching the goal is $0.32776$.

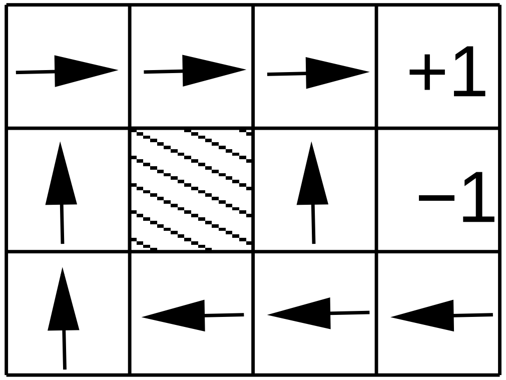
- To complete the description, we have to give a *reward* to each state.
- To give the agent an incentive to reach the goal quickly, we give each non-terminal state a reward of $-0.04$.
- So if the goal is reached after 10 steps, the agent's overall reward is $0.6$.

- This kind of problem is a *Markov Decision Process* (MDP).
- We have:
  - a set of states $S$.
  - an initial state $s_0$.
  - a set $\textsc{Actions}(s)$ of actions in each state.
  - A transition model $\Pr(s'|s, a)$; and
  - A reward function $R(s)$.
- What does a solution look like?

- A plan — a sequence of actions — is not much help.
  - Isn’t guaranteed to find the goal.
- Better is a *policy* $\pi$, which tells us which action $\pi(s)$ to do *in every state*.
- Then the non-determinism doesn’t matter.
  - However badly we do as a result of an action, we will know what to do.

- Because of the non-deteminism, a policy will give us different sequences of actions different times it is run.
- To tell how good a policy is, we can compute the *expected value*.
- We all remember how to do that from the other week, right?
- The *optimal policy* $\pi^*$ is the one that gives the highest expected utility.
  - On average it will give the best reward.
- Given $\pi^*$ an agent doesn't have to think — it just does the right action for the state it is in.

- The optimum policy is then:



- But this is specific to the value of the reward $R(s)$ for non-terminal states.

- How do we find the best policy?
- Turns out that there is a neat way to do this, by first computing the *utility* of each state.
- We compute this using the *Bellman equation*

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

- Does this remind you of anything?
- ($\gamma$ is a discount factor).

- Not this Bellman

(Mervyn Peake's illustrations to "The Hunting of the Snark").

- In an MDP wth $n$ states, we will have $n$ Bellman equations.
- Hard to solvethese simultaneously because of the max operation
  - Makes them non-linear
- Instead use an iterative approach
  - *value iteration*.
- Start with arbitrary values for utilities (say $0$) and then update with:

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

- Repeat until the value stabilises.

# Partial observability

- For all their complexity, MDPs are not an accurate model of the world.
  - Assume accessibility/observability
- To deal with partial observability we have the *Partially observable* Markov decision process (POMDP).
- We don't know which state we are in, but we know what probability we have to being in every state.
- That is all we will say on the subject.

# Reinforcement learning

- Ok, now we have the notion of an MDP, imagine we don't know what the model is.
- We don't know $R(s)$
- We don't know $\Pr(s'|s, a)$
- But it is simple to learn them — the agent just moves around the environment.
- Since it knows what state $s'$ it gets to when it executes $a$ in $s$, it can count how often particular transitions occur to estimate:

  $$\Pr(s'|s, a)$$

  as the proportion of times executing $a$ in $s$ takes the agent to $s'$.
- Similarly the agent can see what reward it gets in $s$ to give it $R(s)$.

- If the agent wanders randomly for long enough, it will learn the probability and reward values.
- (How would it know what “long enough” was?)
- With these values it can apply the Bellman equation(s) and start doing the right thing.

- The agent can also be smarter, and use the values as it learns them.
- At each step it can solve the Bellman equation(s) to compute the best action given what it knows.
- This means it can learn quicker, but also it may lead to sub-optimal performance.

# Summary

- This lecture looked at the basics of reinforcement learning
  - Learning when you only have periodic rewards to guide you.
- The underlying theory of reinforcement learning is that of MDPs.