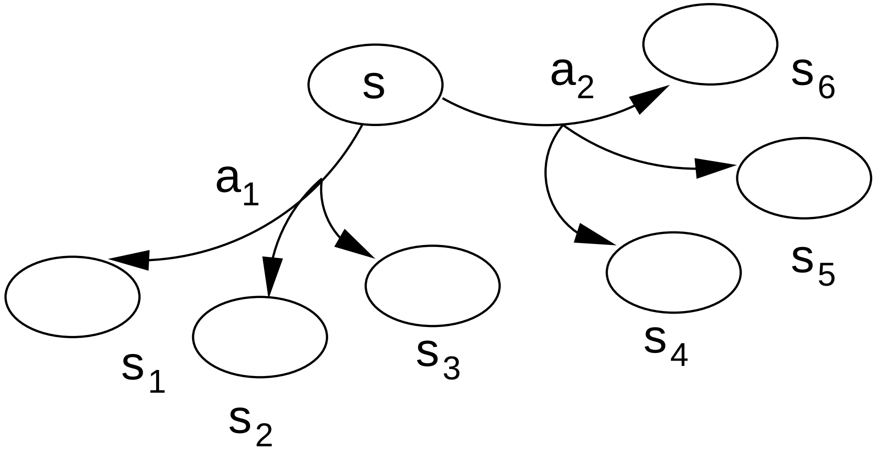
MAKING COMPLEX DECISIONS

Introduction

- A couple of weeks back we looked at how an agent might make a decision under uncertainty.
- Problem structure:
  - Set of non-deterministic actions.
  - Resulting states have utilities associated with them.
- We use probability + utility theory (= decision theory) to establish the best choice of action.

Simple decisions

- Consider an agent with a set of possible actions A available to it.
- Each $a \in A$ has a set of possible outcomes s_a:



- For the outcomes s_a of each action each a, the agent can calculate:

$$E(u(s_a)) = \sum_{s' \in s_a} u(s').P(s_a = s')$$

- A *rational* agent needs to choose a^* such that:

$$a^* = \arg\max_{a \in A} \sum_{s' \in s_a} u(s').P(s_a = s')$$

- That is it picks the action that has the greatest expected utility.
- Here “rational” means “rational in the sense of maximising expected utility”.

- There are other criteria for decision-making than maximising expected utility.
- One approach is to look at the option which has the least-bad worst outcome.
- This *maximin* criterion can be formalised in the same framework as MEU, making the rational (in this sense) action:

$$a^* = \arg\max_{a \in A}\{\min_{s' \in s_a} u(s')\}$$

- Its effect is to ignore the probability of outcomes and concentrate on optimising the worst case outcome.

- The opposite attitude, that of optimisitic risk-seeker, is captured by the *maximax* criterion:

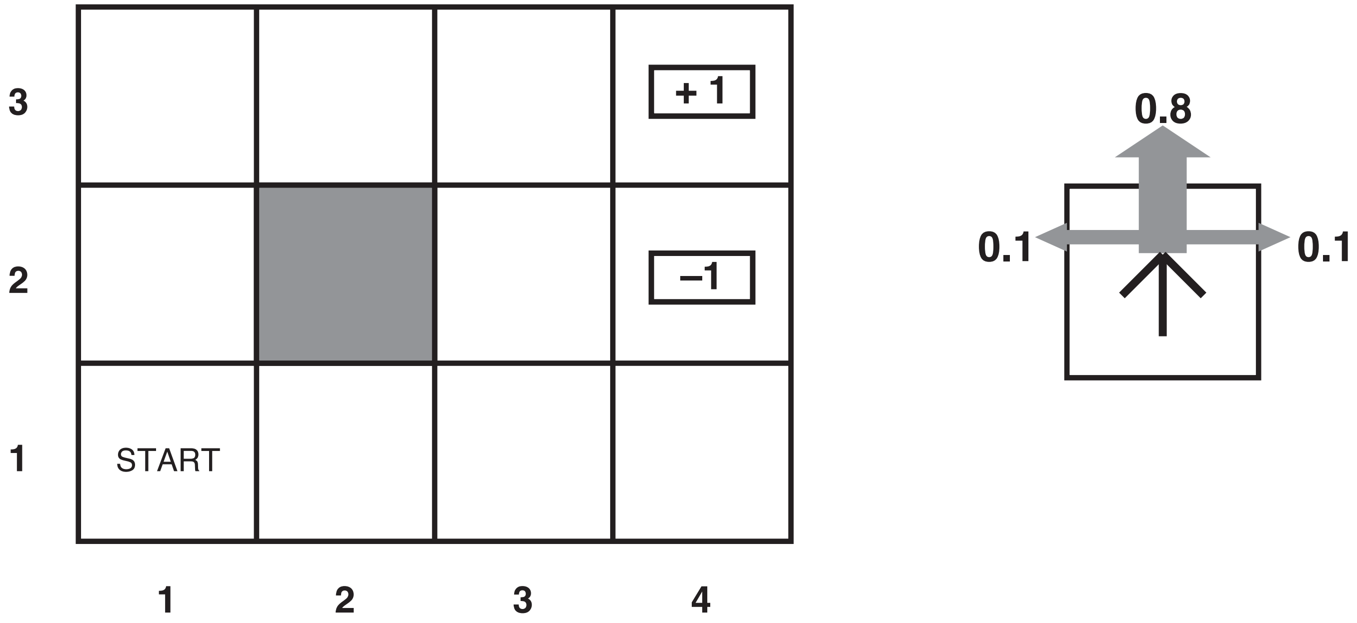
$$a^* = \arg \max_{a \in A} \{ \max_{s' \in s_a} u(s') \}$$

- This will ignore possible bad outcomes and just focus on the best outcome of each action.

Sequential decision problems

- These approaches give us a battery of techniques to apply to individual decisions by agents.
- However, they aren't really sufficient.
- Agents aren't usually in the business of taking single decisions
  - Life is a series of decisions.

  The best overall result is not necessarily obtained by a greedy approach to a series of decisions.
- Need to think about *sequential decision problems* where the agent's utility depends on a sequence of decisions.



(a) (b)

- The agent has to pick a sequence of actions.

$$A(s) = \{Up, Down, Left, Right\}$$

  for all states s.

- The world is fully observable. End states have values $+1$ or -1.

- If the world were deterministic, the choice of actions would be easy here.

$$Up, Up, Right, Right, Right$$

- But actions are stochastic.
- 80% of the time the agent moves as intended, but 20% of the time the agent moves perpendicular to the intended direction.

  Half the time to the left, half the time to the right. The agent doesn't move if it hits a wall.
- So *Up*, *Up*, *Right*, *Right*, *Right* succeeds with probability:

$$0.8^5 = 0.32768$$

- (Also a small chance of going around the obstacle the other way.)

- We can write a *transition* model to describe these actions.
- Since the actions are stochastic, the model looks like:

  $$P(s'|s,a)$$

  where a is the action that takes the agent from s to s'.
- Transitions are assumed to be Markovian.
- So, we could write a large set of probability tables that would describe all the possible actions executed in all the possible states.

  This would completely specify the actions.

- The full description of the problem also has to include the utility function.
- This is defined over sequences of states — *runs* in the terminology of the first lecture.
- We will assume that in each state s the agent receives a reward $R(s)$.
- This may be positive or negative.
- The reward for non-terminal states is -0.04.
- We will assume that the utility of a run is the sum of the utilities of states, so the -0.04 is an incentive to take fewer steps to get to the terminal state.

  (You can also think of it as the cost of an action).

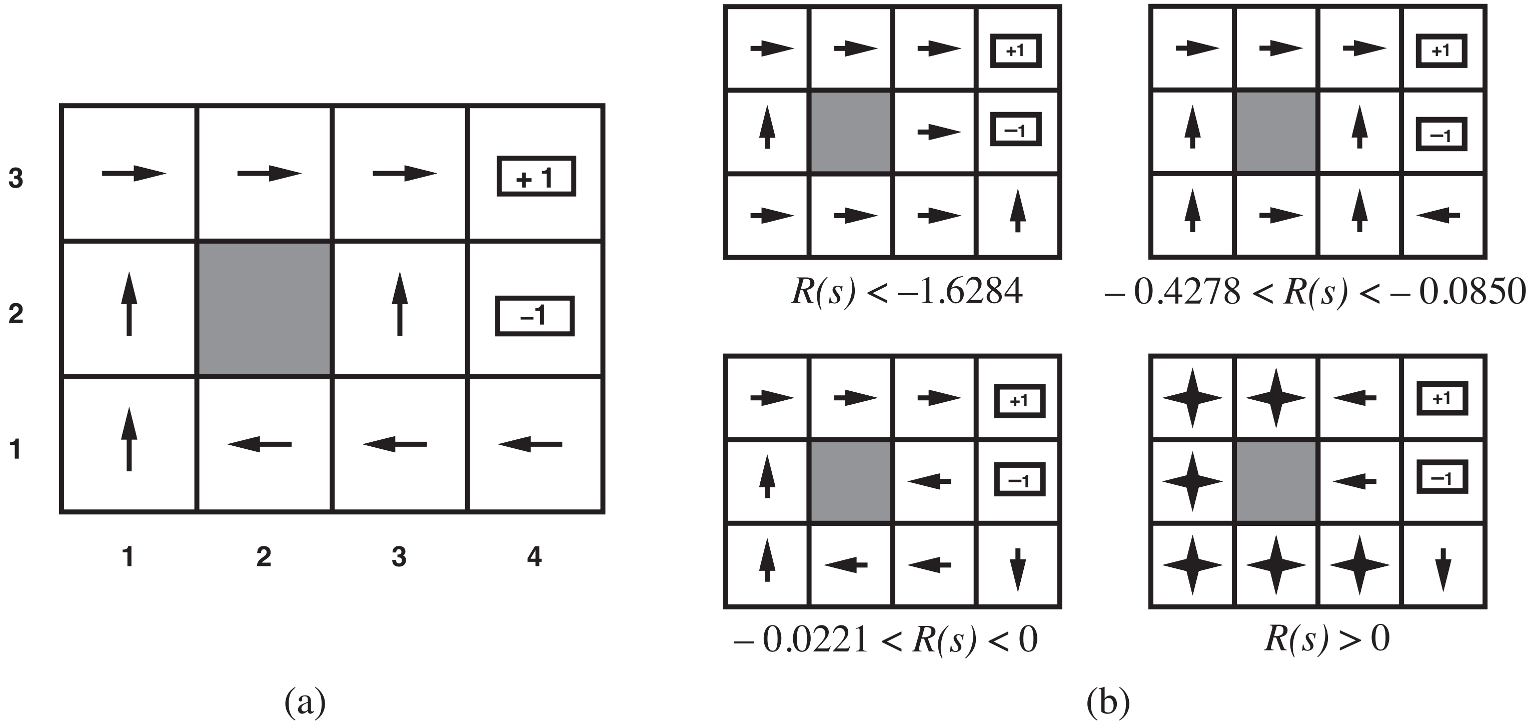
Markov decision process

- The overall problem the agent faces here is a *Markov decision process* (MDP)
- That is any fully observable non-deterministic environment with a Markovian transition model and additive rewards.
- Mathematically we have
  - a set of states $s \in S$ with an initial state s_0.
  - A set of actions $A(s)$ in each state.
  - A transition model $P(s'|s, a)$; and
  - A reward function $R(s)$.

- What does a solution to an MDP look like?

- A solution is a *policy*, which we write as π.
- This is a choice of action for *every* state.
  - that way if we get off track, we still know what to do.
- In any state s, $\pi(s)$ identifies what action to take.

- Naturally we'd prefer not just any policy but the *optimum* policy.
  - But how to find it?
- Need to compare policies by the reward they generate.
- Since actions are stochastic, policies won't give the same reward every time.
  - So compare the expected value.
- The optimum policy π^* is the policy with the highest expected value.
- At every stage the agent should do $\pi^*(s)$.



(a) Optimal policy for the original problem.

(b) Optimal policies for different values of $R(s)$.

- $R(s) \leq -1.6284$, life is painful so the agent heads for the exit, even if is a bad state.
- $-0.4278 \leq R(s) \leq -0.0850$, life is unpleasant so the agent heads for the $+1$ state and is prepared to risk falling into the -1 state.
- $-0.0221 < R(s) < 0$, life isn't so bad, and the optimal policy doesn't take any risks.
- $R(s) > 0$, the agent doesn't want to leave.

How utilities are calculated

- So far we have assumed that utilities are summed along a run.
  - Not the only way.
- In general we need to compute $U_r([s_0, s_1, \ldots, s_n])$.
- Can consider *finite* and *infinite* horizons.
  - Is it “game over” at some point?
- Turns out that infinite horizons are mostly easier to deal with.
  - That is what we will use.
- Also have to consider whether utilities are *stationary* or *non-stationary*.
  - Does the same state always have the same value?

- Normally if we prefer one state to another
  - Passing the mid-term to failing it

  when we have the exam, today or next week, is irrelevant.
- So utilities are *stationary*.

- With stationary utilities, there are two ways to establish $U_r([s_0, s_1, \ldots, s_n])$ from $R(s)$.
- *Additive* rewards:

$$U_r([s_0, s_1, \ldots, s_n]) = R(s_0) + R(s_1) + \ldots + R(s_n)$$

as above.

- *Discounted* rewards:

$$U_r([s_0, s_1, \ldots, s_n]) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^n R(s_n)$$

where the *discount factor* γ is a number between 0 and 1.

- The discount factor models the preference of the agent for current over future rewards.

- There is an issue with infinite sequences with additive, undiscounted rewards.
  - What will the utility of a policy be?

- There is an issue with infinite sequences with additive, undiscounted rewards.
  - What will the utility of a policy be?
- ∞ or $-\infty$.
- This is problematic if we want to compare policies.

- Some solutions as follows.
- *Proper policies* always end up in a terminal state eventually. Thus they have a finite expected utility.
- We can compute the *average reward* per time step.
- With discounted rewards the utility of an infinite sequence is finite:

$$\begin{aligned} U_r([s_0, s_1, \ldots, s_n]) &= \sum_{t=0}^{\infty} \gamma^t R(s_t) \\ &\leq \sum_{t=0}^{\infty} \gamma^t R_{max} \\ &\leq \frac{R_{max}}{(1-\gamma)} \end{aligned}$$

where $0 \leq \gamma < 1$ and rewards are bounded by $\pm R_{max}$

Optimal policies

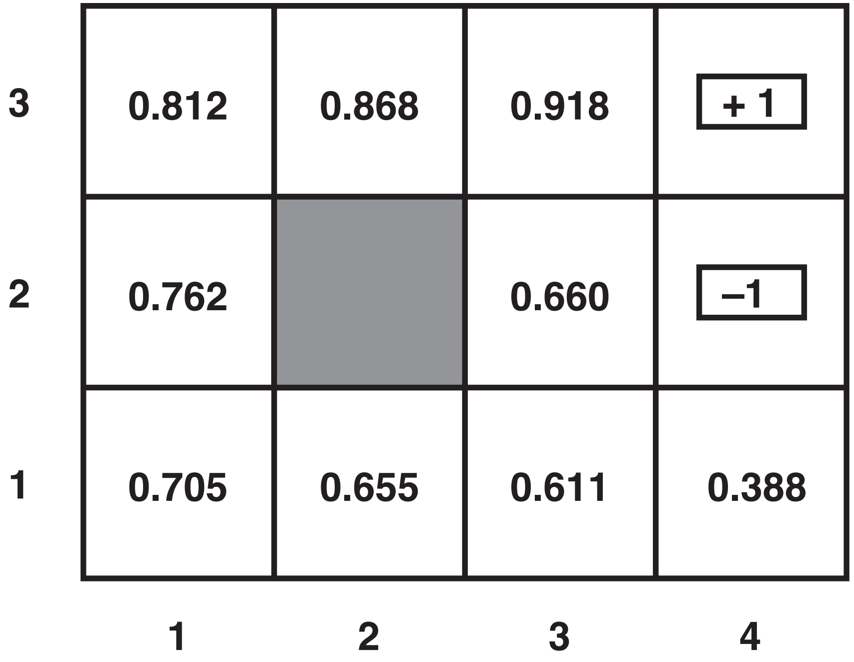
- With discounted rewards we compare policies by computing their expected values.
- The expected utility of executing π starting in s is given by:

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

  where S_t is the state the agent gets to at time t.
- S_t is a random variable and we compute the probability of all its values by looking at all the runs which end up there after t steps.
- The optimal policy is then:

$$\pi^* = \arg\max_{\pi} U^{\pi}(s)$$

  and it turns out that this is independent of the state the agent starts in.



3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

- Here we have the values of states if the agent executes an optimal policy

$$U^{\pi^*}(s)$$

- If we have these values, the agent has a simple decision process
- It just picks the action a that maximises the expected utility of the next state:

$$\pi^*(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U^{\pi^*}(s')$$

- The big question is how to compute $U^{\pi^*}(s)$.

3	0.812	0.868	0.918	+ 1
2	0.762	▨	0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

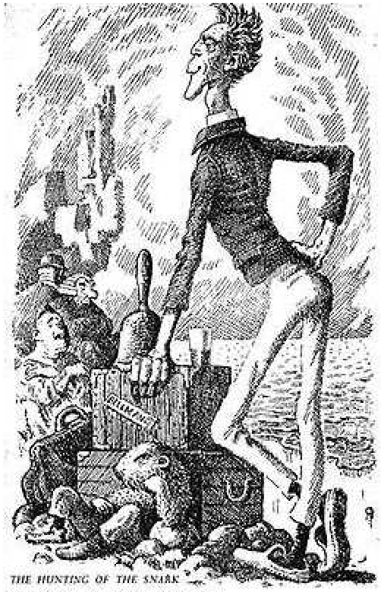
3	→	→	→	+1
2	↑	▨	↑	–1
1	↑	←	←	←
	1	2	3	4

Value iteration

- The key to computing the utility of a state is the *Bellman equation*

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

which says the utility of a state is the reward for being in that state plus the expected discounted reward of being in the next state, assuming the agent picks the optimal action.



THE HUNTING OF THE SNARK

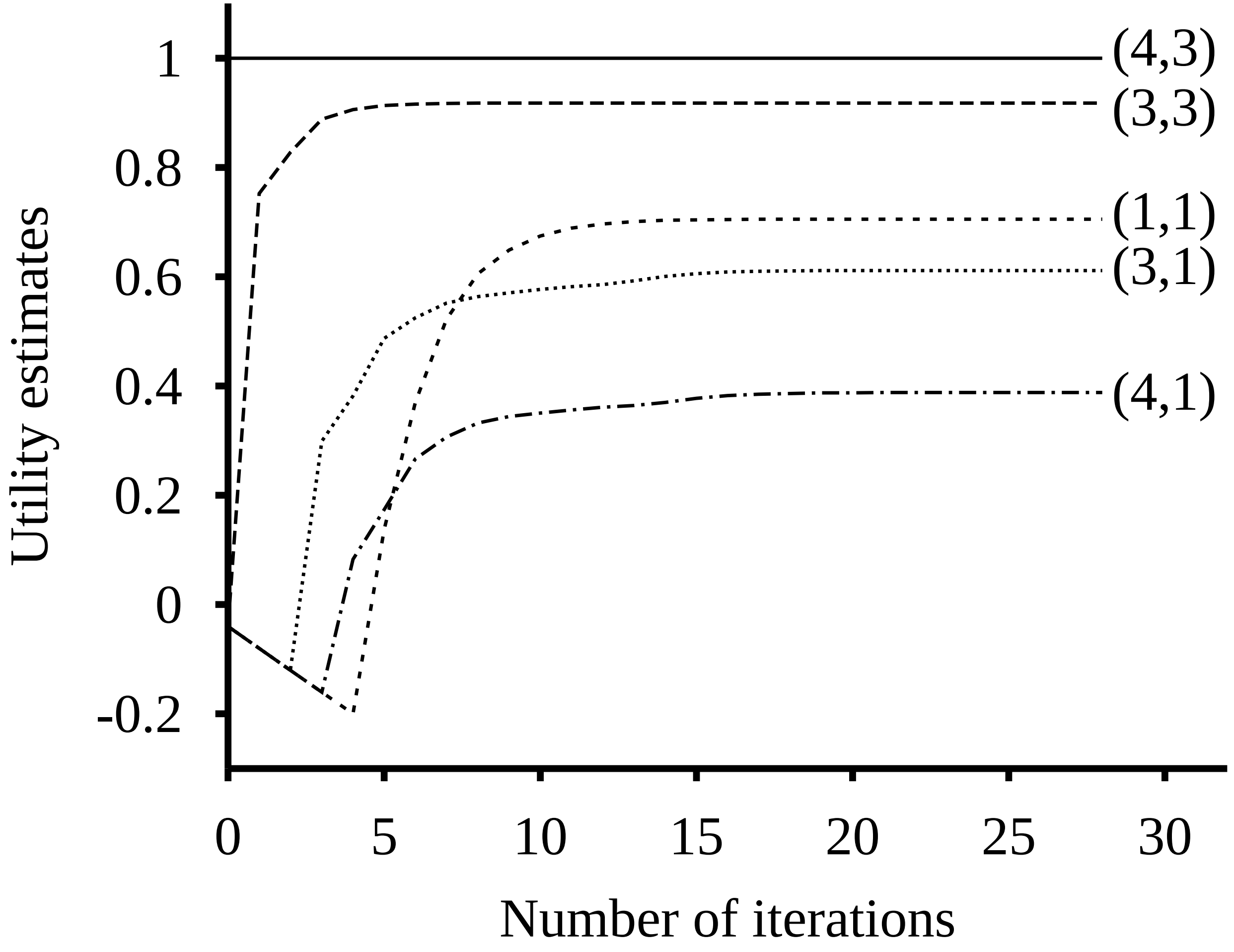
© The Mervyn Peake Estate

- There will be a set of Bellman equations, one for each state.
- We need to solve this set of (non-linear) equations.
  - Hard

- Luckily an iterative approach works.
- Start with arbitrary values for states and apply the Bellman update:

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

  simultaneously to all the states.
- Continue until the values of states do not change.
- After an infinite number of applications, the values will converge on the optimal values.



- How the values of states change as updates occur.

Policy iteration

- Rather than compute optimal utility values, *policy iteration* looks through the space of possible policies.
- Starting from some initial policy π_0 we do:
  - *Policy evaluation*
    Given a policy π_i, calculate $U_i = U^{\pi_i}$.
  - *Policy improvement*
    Calculate a new policy π_{i+1} by applying:

$$\pi_{i+1}(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'|s, a) U_i(s')$$

- The iteration will terminate when there is no improvement in utility from one iteration to the next.
- At this point the utility U_i is a fixed point of the Bellman update and so π_i must be optimal.

- How do we calculate the utility of each step given the policy π_i?
- Turns out not to be so hard.
- Given a policy, the choice of action in a given state is fixed (that is what a policy tells us) so:

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s))U_i(s')$$

- Again there are lots of simultaneous equations, but now they are linear (no $\max$) and so standard linear algebra solutions will work.

Bellman redux

- The Bellman equation(s)/update are widely used.

- D. Romer, It's Fourth Down and What Does the Bellman Equation Say? A Dynamic Programming Analysis of Football Strategy, NBER Working Paper No. 9024, June 2002

This paper uses play-by-play accounts of virtually all regular season National Football League games for 1998-2000 to analyze teams' choices on fourth down between trying for a first down and kicking. Dynamic programming is used to estimate the values of possessing the ball at different points on the field. These estimates are combined with data on the results of kicks and conventional plays to estimate the average payoffs to kicking and going for it under different circumstances. Examination of teams' actual decisions shows systematic, overwhelmingly statistically significant, and quantitatively large departures from the decisions the dynamic-programming analysis implies are preferable.

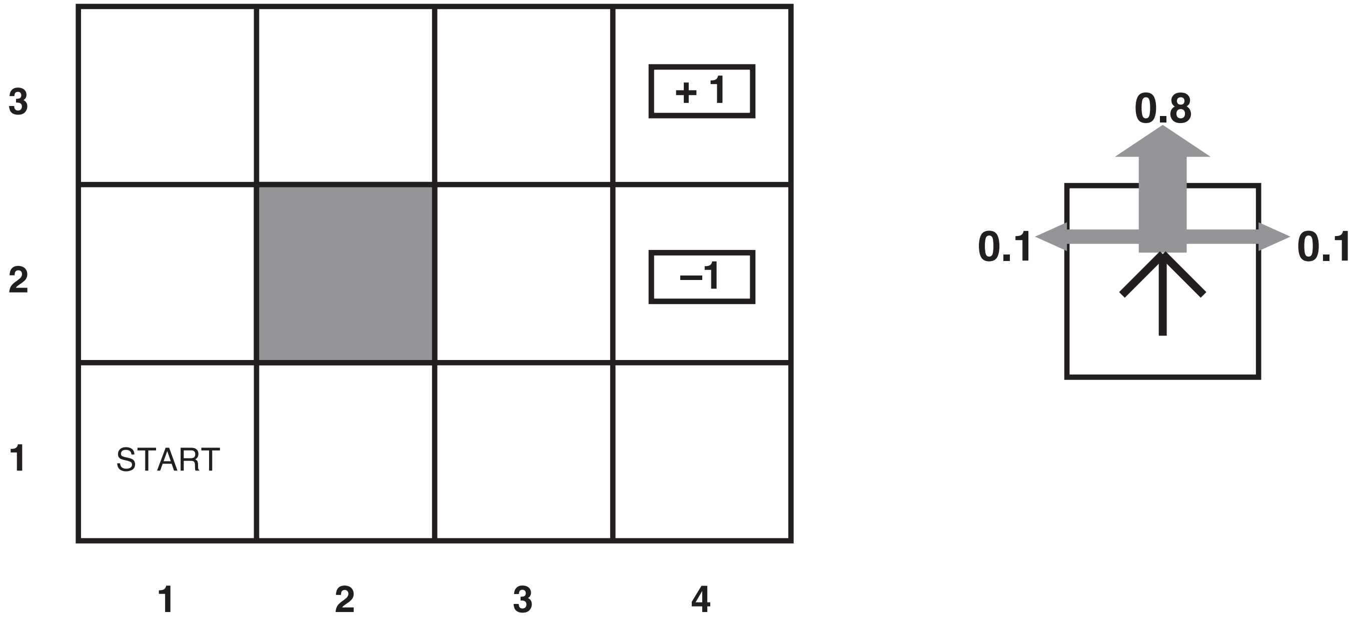
Partially observable MDPs

- MDPs made the assumption that the environment was fully observable.
  - Agent always knows what state it is in.
- The optimal policy only depends on the current state.
- Not the case in the real world.
  - We only have a belief about the current state.
- POMDPs extend the model to deal with partial observability.

- Basic addition to the MDP model is the *sensor* model:

  $$P(e|s)$$

  probability of perceiving e in state s.
- As a result of noise in the sensor model, the agent only has a belief about which state it is in.
- Probability distribution over the possible states.



(a) (b)

$$P(s_{1,1}) = 0.05, P(s_{1,2}) = 0.01, \ldots$$

- The agent can compute its current belief as the conditional probability distribution over the states given the sequence of actions and percepts so far.
  - Sound familiar?

- The agent can compute its curent belief as the conditional probability distribution over the states given the sequence of actions and percepts so far.
  - Sound familiar?
- This is basically the filtering task from the last class.

  (albeit with an action).

- If $(b(s)$ was the distribution before an action and an observation, then afterwards the distribution is:

$$b'(s') = \alpha P(e|s') \sum_s P(s'|s,a) b(s)$$

- Everything in a POMDP hinges on the belief state b.
  - Including the optimal action.
- Indeed, the optimal policy is a mapping $\pi^*(b)$ from beliefs to actions.

  "If you think you are next to the wall, turn left"
- The agent executes the optimal action given its beliefs, receives a percept e and then recomputes the belief state.

- The big issue in solving POMDPs is that beliefs are continuous.
- When we solved MDPs, we could search through the set of possible actions in each state to find the best.
- To solve a POMDP, we need to look through the possible actions for each belief state.

  But belief is continuous, so there are a lot of belief states.
- Exact solutions to POMDPs are intractable for even small problems (like the example we have been using).
- Need (once again) to use approximate techniques.

Summary

- Today we looked at practical decision making for agents.
  - Practical in the sense that agents will need this kind of decision making to do the things they need to do.
- We looked in detail at solutions for techniques that work in fully observable worlds
  - MDPs
- We also briefly mentioned the difficulties of extending this work to partially observable worlds.