Lecture 4: Sequential decision making

Simon Parsons

Department of Informatics
King's College London

Version 1.3

Today

- Introduction
- Probabilistic Reasoning I
- Probabilistic Reasoning II
- Sequential Decision Making
- Temporal Probabilistic Reasoning
- Game Theory
- Argumentation I
- Argumentation II
- (A peek at) Machine Learning
- AI & Ethics

What to do?

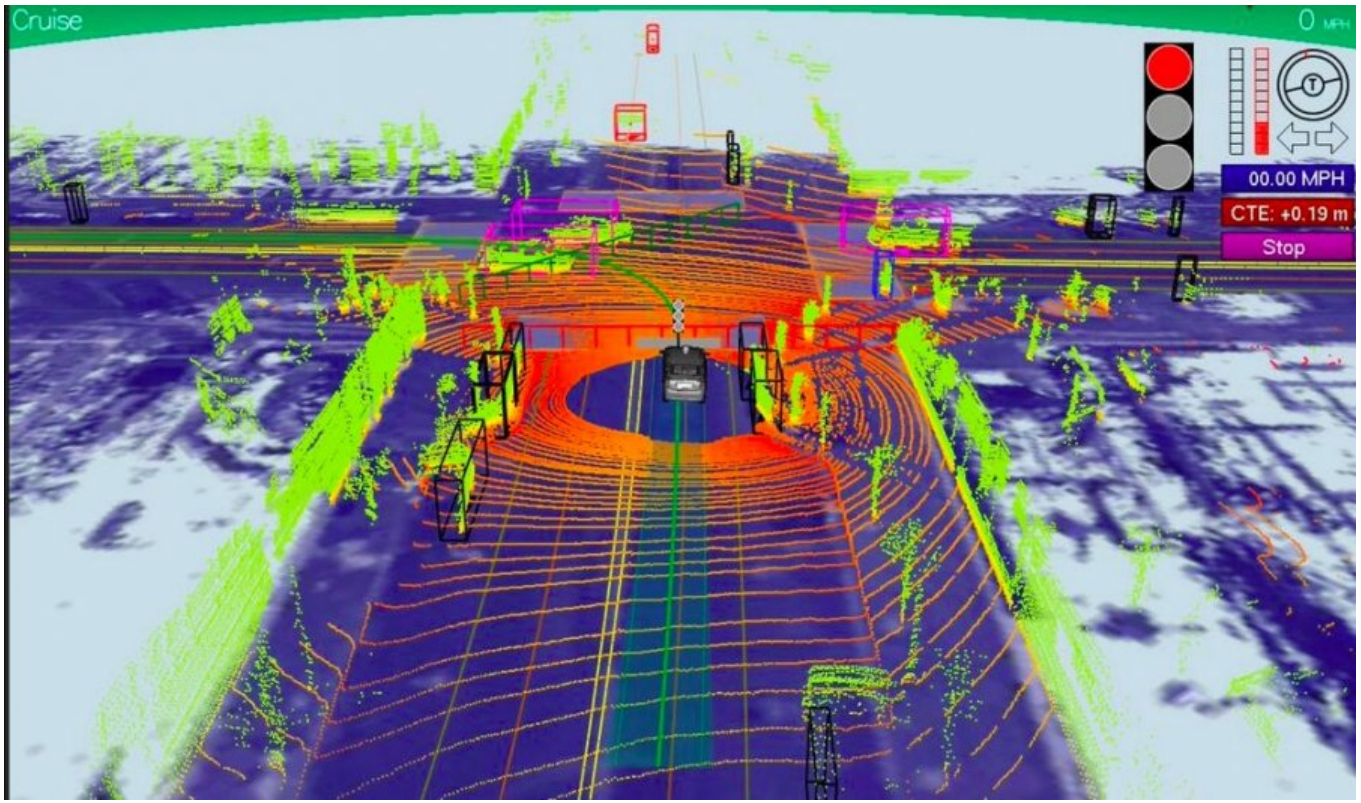
(40 Acres and a Mule Filmworks/Universal Pictures)

KING'S College LONDON

What to do?

(mystorybook.com/books/42485)

What to do?



(Sebastian Thrun & Chris Urmson/Google)

Sequential decision making?

(mystorybook.com/books/42485)

- One decision leads to another.
- Each decision depends on the ones before, and affects the ones after.

How to decide what to do

- Start simple.
- Single decision.
- Consider being offered a bet in which you pay £2 if an odd number is rolled on a die, and win £3 if an even number appears.
- Is this a good bet?

How to decide what to do

- Consider being offered a bet in which you pay £2 if an odd number is rolled on a die, and win £3 if an even number appears.
- Is this a good bet?
- To analyse this, we need the expected value of the bet.

How to decide what to do

- We do this in terms of a random variable, which we will call X.
- X can take two values:

  3 if the die rolls odd

  -2 if the die rolls even
- And we can also calculate the probability of these two values

  $\Pr(X = 3) = 0.5$

  $\Pr(X = -2) = 0.5$

How to decide what to do

- The expected value is then the weighted sum of the values, where the weights are the probabilities.
- Formally the expected value of X is defined by:

$$E(X) = \sum_k k \Pr(X = k)$$

where the summation is over all values of k for which $\Pr(X = k) \neq 0$.

- Here the expected value is:

$$E(X) = 0.5 \times 3 + 0.5 \times -2$$

- Thus the expected value of X is £0.5, and we take this to be the value of the bet.

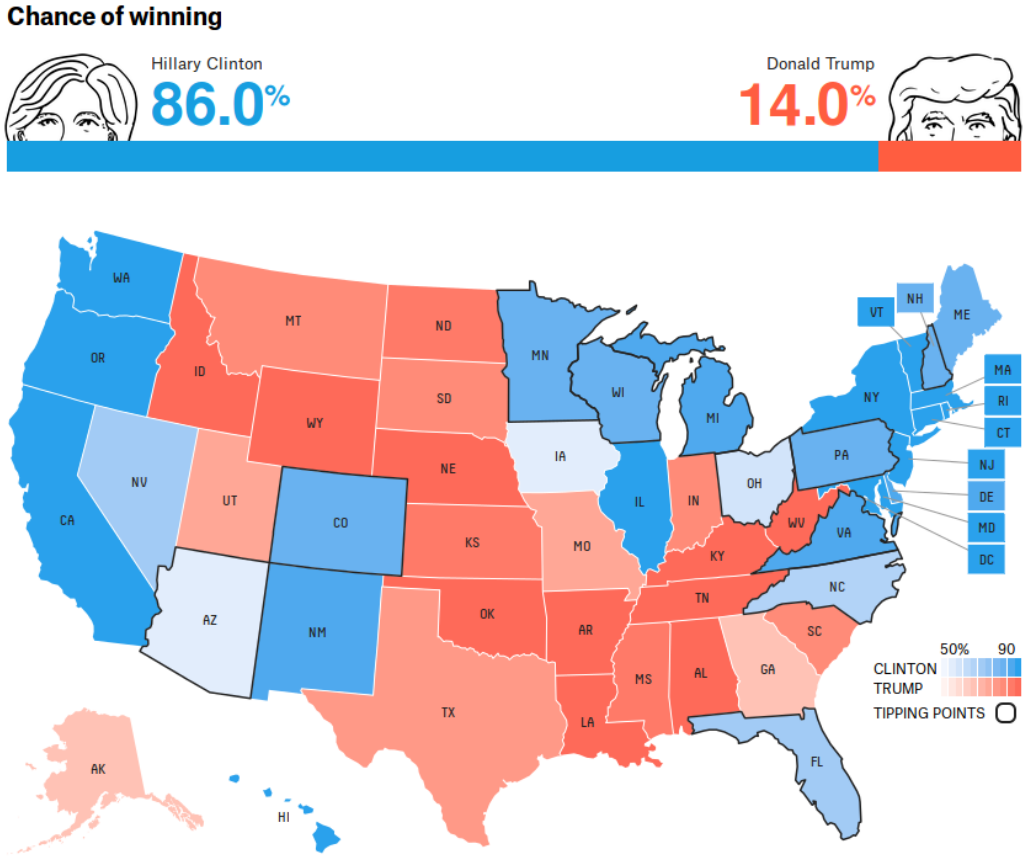
How to decide what to do

- Do you take the bet?
- Compare that 0.5 with not taking the bet.
- Not taking the bet has (expected) value £0

How to decide what to do

- 0.5 is not the value you will get.
- You can think of it as the long run average if you were offered the bet many times.

How to decide what to do



(fivethirtyeight.com)

How to decide what to do

- Another bet: you get £1 if a 2 or a 3 is rolled, £5 if a six is rolled, and pay 3 otherwise.
- The expected value here is:

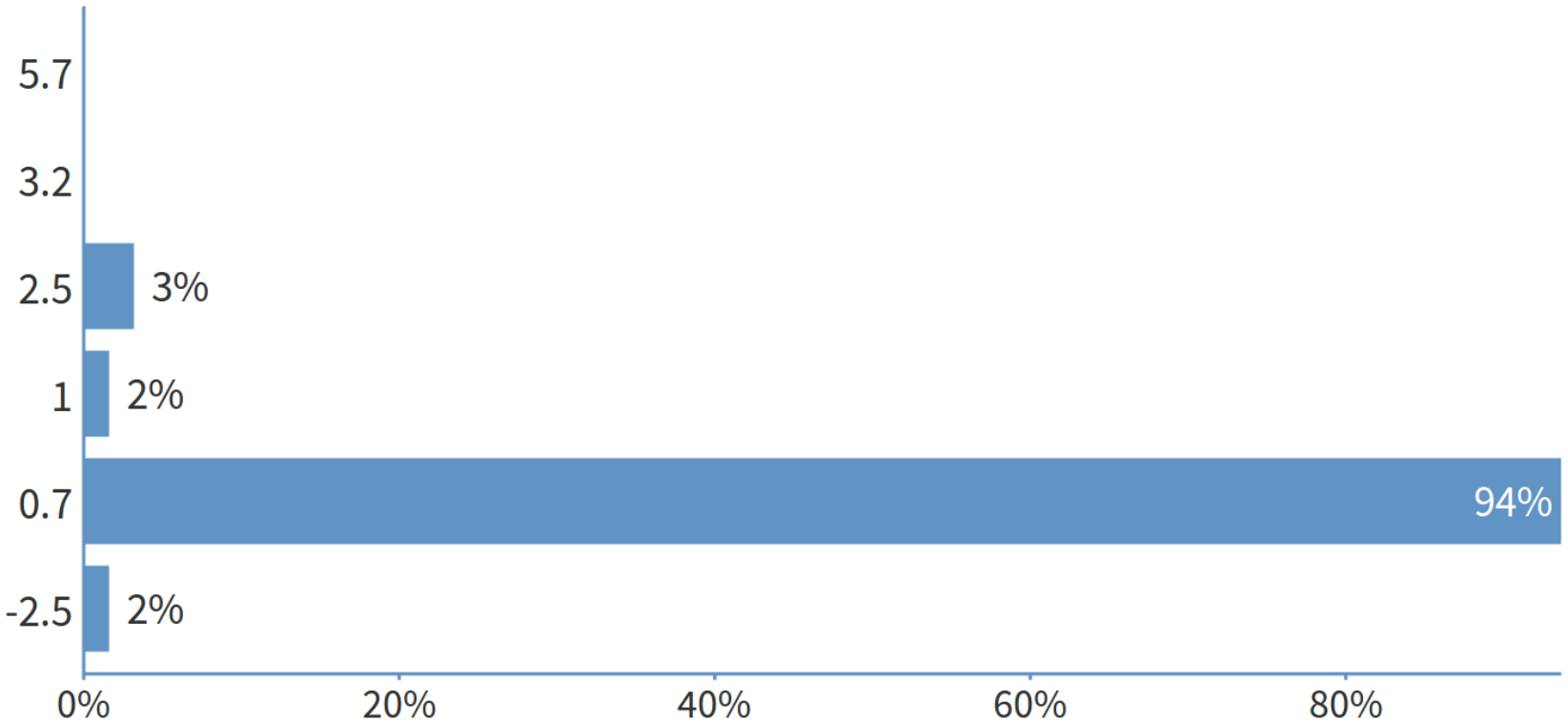
$$E(X) = 0.333 \times 1 + 0.166 \times 5 + 0.5 \times -3$$

which is -0.33.

Example

- Pacman is at a T-junction
- Based on their knowledge, estimates that if they go Left:
  - Probability of 0.3 of getting a payoff of 10
  - Probability of 0.2 of getting a payoff of 1
  - Probability of 0.5 of getting a payoff of -5
- What is the expected value of Left?

Expected value of left



How an agent might decide what to do

- Consider an agent with a set of possible actions A.
- Each $a \in A$ has a set of possible outcomes s_a.
- Which action should the agent pick?

How an agent might decide what to do

- The action a^* which a rational agent should choose is that which maximises the agent's utility.
- In other words the agent should pick:

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

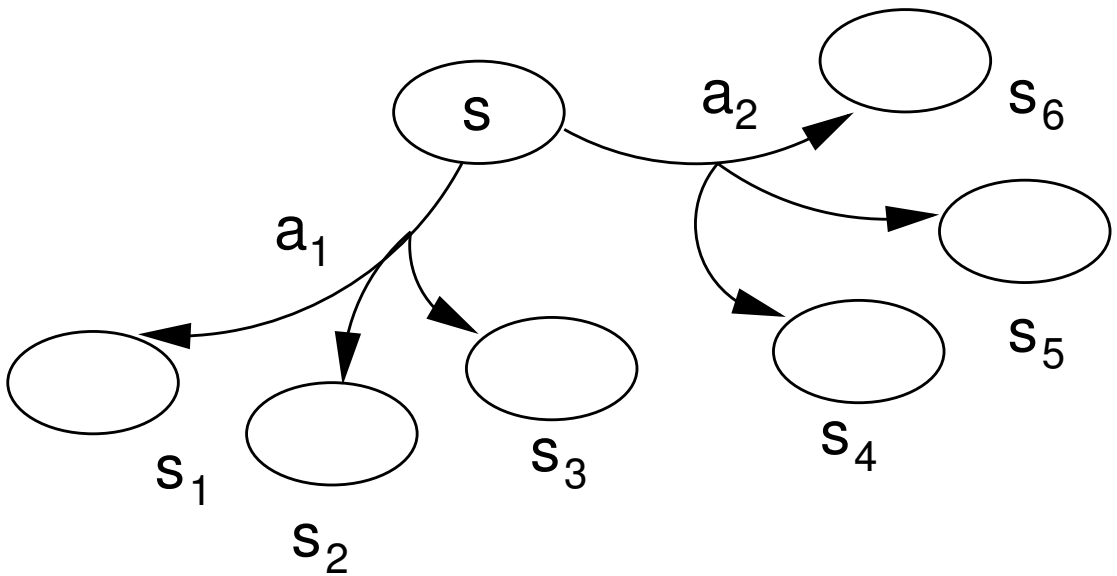
- The problem is that in any realistic situation, we don't know which s_a will result from a given a, so we don't know the utility of a given action.
- Instead we have to calculate the expected utility of each action and make the choice on the basis of that.

How an agent might decide what to do

- In other words, for each action a with a set of outcomes s_a, the agent should calculate:

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

and pick the best.



How an agent might decide what to do

- That is it picks the action that has the greatest expected utility.
  - The right thing to do.

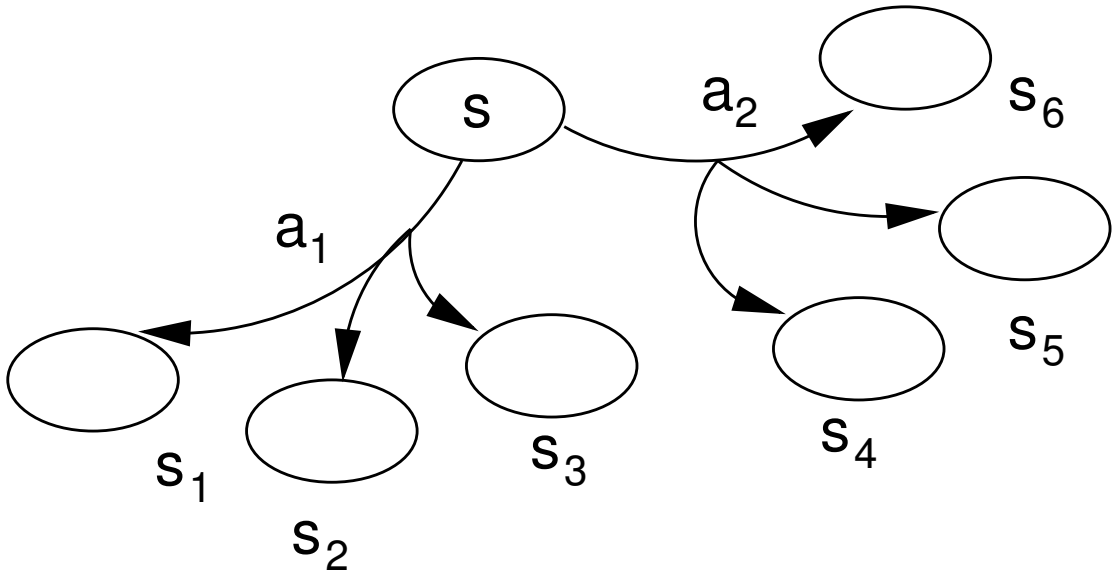
(40 Acres and a Mule Filmworks/Universal Pictures)

- Here “rational” means “rational in the sense of maximising expected utility”.

KING'S College LONDON

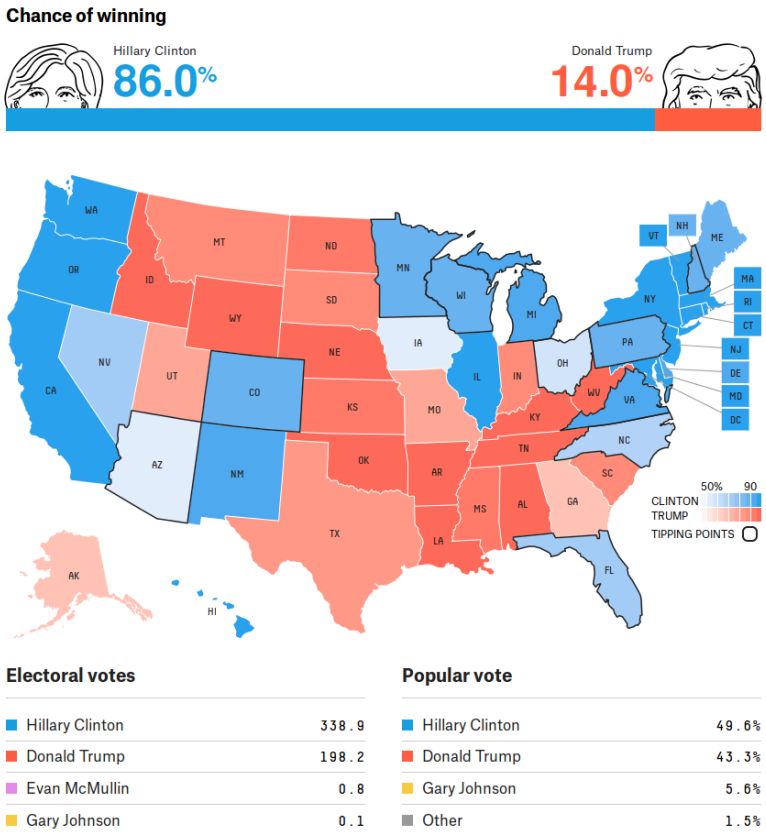
Non-deterministic

- Note that we are dealing with non-deterministic actions here.



- A given action has several possible outcomes.
- We don't know, in advance, which one will happen.

Non-deterministic



Electoral votes	
Hillary Clinton	338.9
Donald Trump	198.2
Evan McMullin	0.8
Gary Johnson	0.1

Popular vote	
Hillary Clinton	49.6%
Donald Trump	43.3%
Gary Johnson	5.6%
Other	1.5%

(fivethirtyeight.com)

- A lot like life.

Other notions of “rational”

- 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.

Other notions of “rational”

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

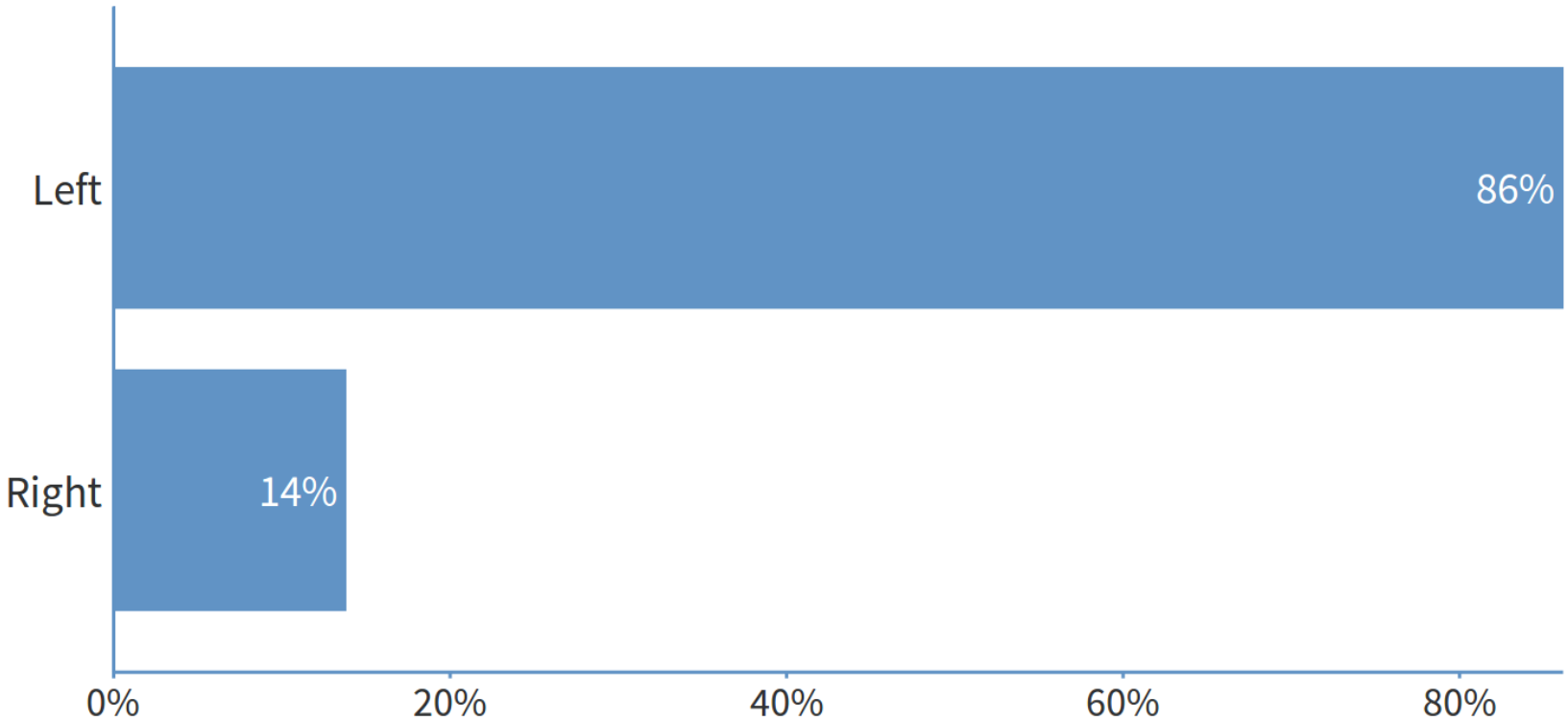
$$a^* = \arg\max_{a \in A} \{\max_{s' \in s_a} u(s')\}$$

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

Example

- Pacman is at a T-junction
- Based on their knowledge, estimates that if they go Left:
  - Probability of 0.3 of getting a payoff of 10
  - Probability of 0.2 of getting a payoff of 1
  - Probability of 0.5 of getting a payoff of -5
- If they go Right:
  - Probability of 0.5 of getting a payoff of -5
  - Probability of 0.4 of getting a payoff of 3
  - Probability of 0.1 of getting a payoff of 15
- Should they choose Left or Right (MEU)?

Example



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.
- The current best option isn't the best thing in the long-run.

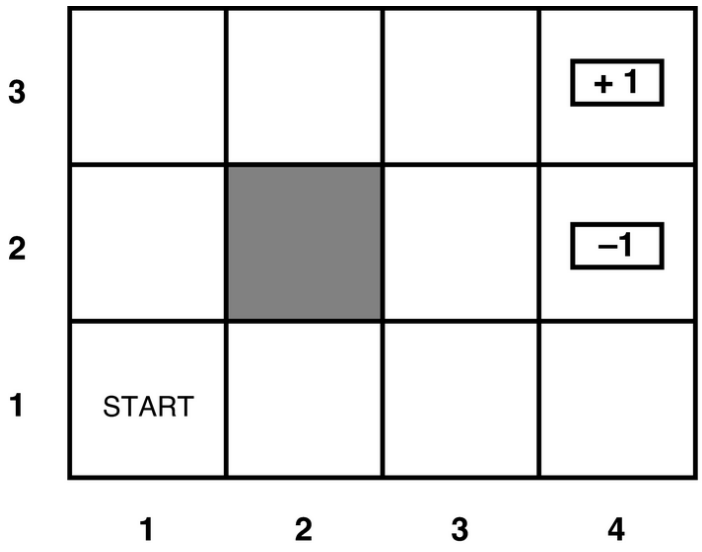
Sequential decision problems

- Otherwise I'd only ever eat cherry pie

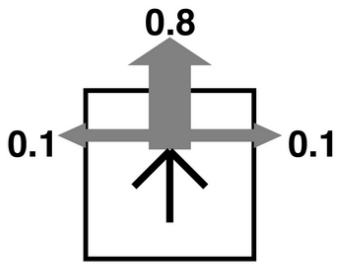
(pillsbury.com)

(Damn fine pie.)

An example



(a)



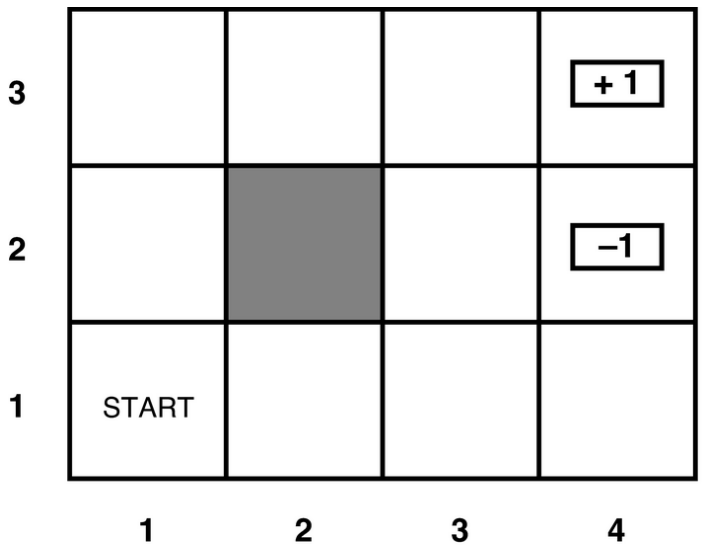
(b)

- The agent has to pick a sequence of actions.

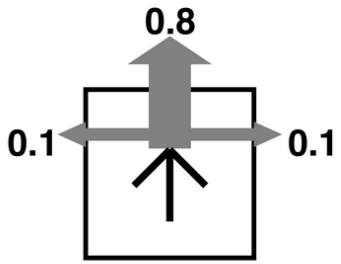
$$A(s) = \{Up, Down, Left, Right\}$$

for all states s.

An example



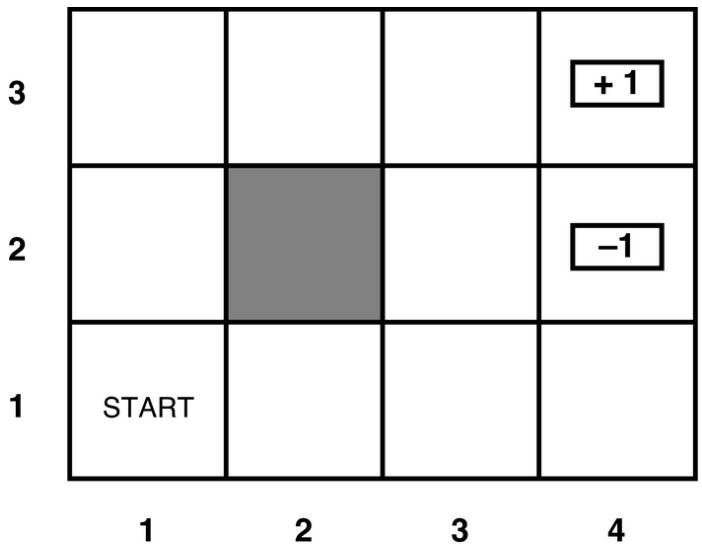
(a)



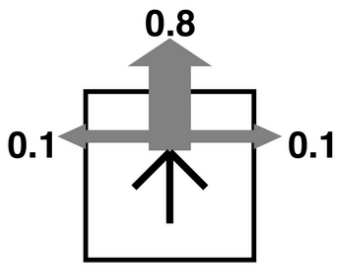
(b)

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

An example



(a)



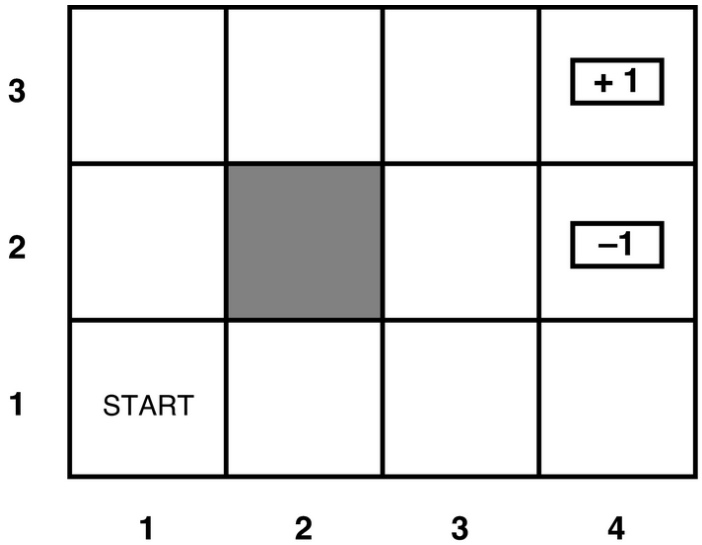
(b)

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

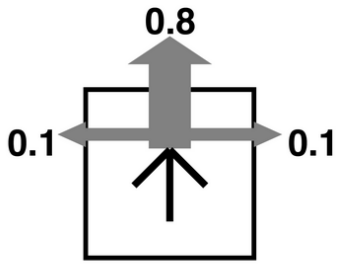
$$\textit{Up}, \textit{Up}, \textit{Right}, \textit{Right}, \textit{Right}$$

- But actions are stochastic.

An example



(a)

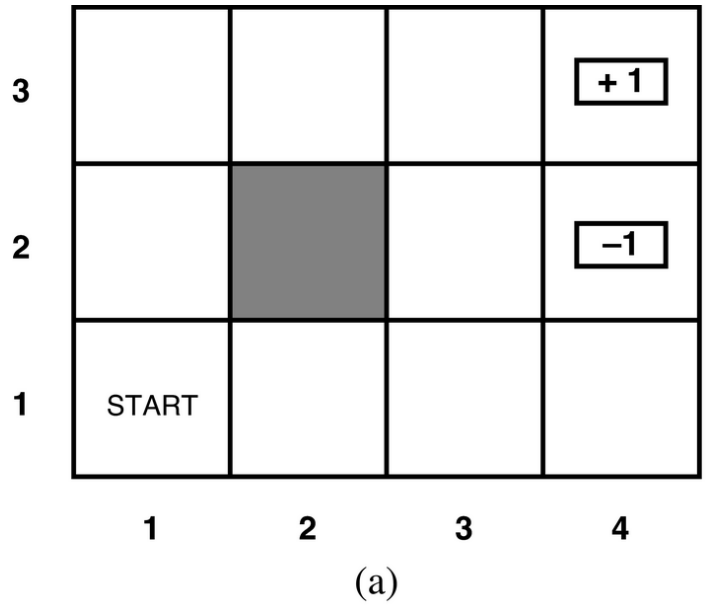


(b)

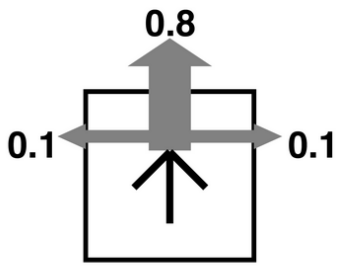
- 80% of the time the agent moves as intended.
- 20% of the time the agent moves perpendicular to the intended direction. Half the time to the left, half the time to the right.
- The agent doesn't move if it hits a wall.

KING'S College LONDON

An example



(a)

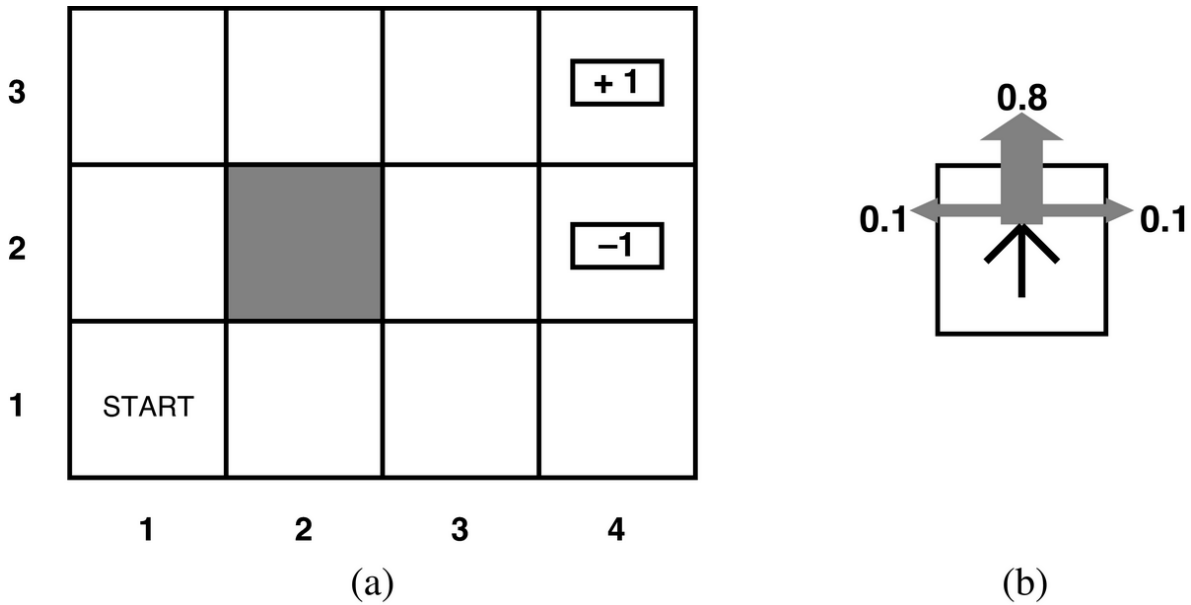


(b)

- So *Up*, *Up*, *Right*, *Right*, *Right* succeeds with probability:

$$0.8^5 = 0.32768$$

An example



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

An example

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

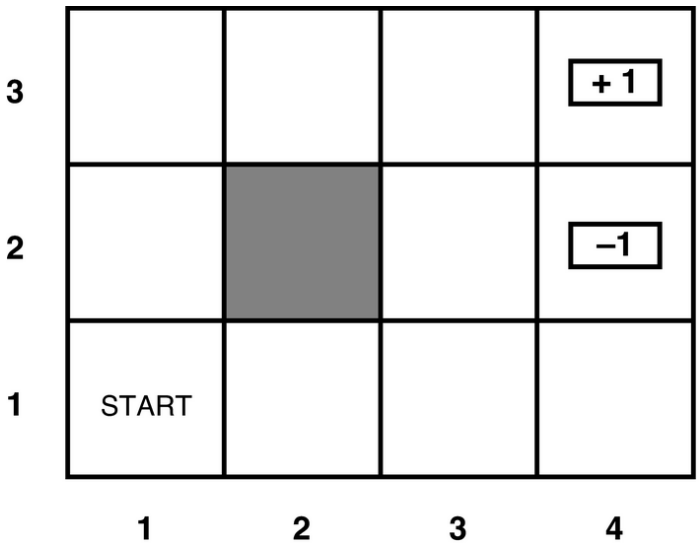
$$P(s'|s,a)$$

  where a is the action that takes the agent from s to s'.
- Transitions are assumed to be (first order) Markovian.
- They only depend on the current and next states.
- 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.

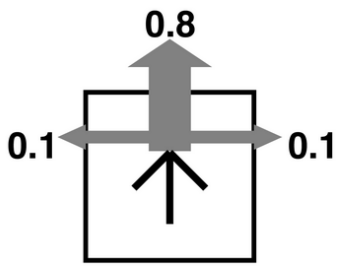
An example

- 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.

An example



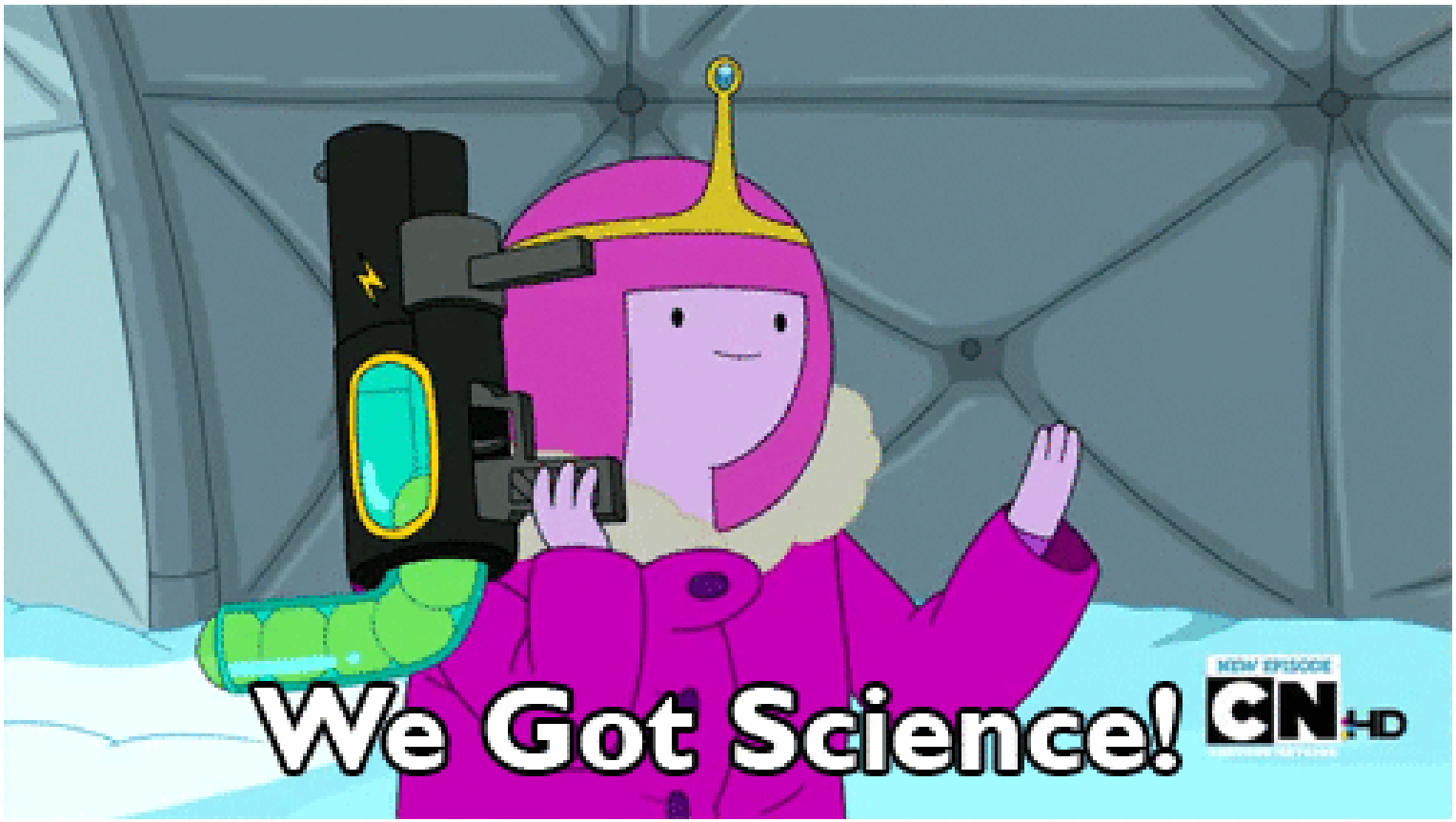
(a)



(b)

- 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).

How do we tackle this?



(Pendleton Ward/Cartoon Network)

Markov decision process

- The overall problem the agent faces here is a Markov decision process (MDP)
- 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 $R(s)$.
- Captures any fully observable non-deterministic environment with a Markovian transition model and additive rewards.

Leslie Pack Kaelbling

Markov decision process

- What does a solution to an MDP look like?

Markov decision process

- 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.

Markov decision process

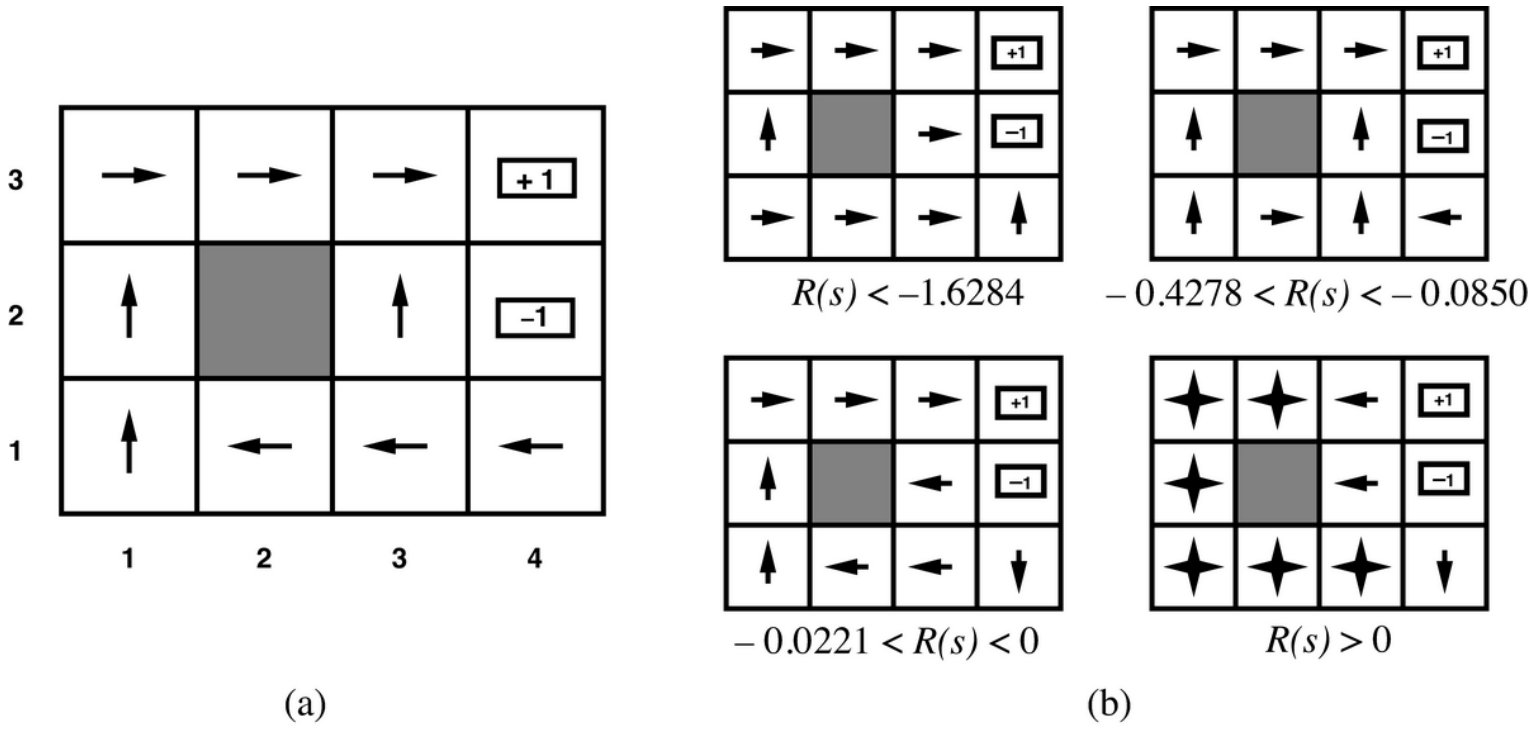
- Naturally we'd prefer not just any policy but the optimum policy.
  - But how to find it?
- Need to compare policies by the reward they generate.
- Since actions are stochastic, policies won't give the same reward every time.
  - So compare the expected value.
- The optimum policy π^* is the policy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

Markov decision process

(40 Acres and a Mule Filmworks/Universal Pictures)

- $\pi^*(s)$ is the right thing.

An example



(a) Optimal policy for the original problem.

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

An example

- $R(s) \leqslant -1.6284$, life is painful so the agent heads for the exit, even if is a bad state.
- $-0.4278 \leqslant R(s) \leqslant -0.0850$, 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.

KING'S College LONDON

How utilities are calculated

- 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 AI module to failing it

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

But are they?

- Not clear that utilities are always stationary.

- In truth, I don't always most want to eat cherry pie.
- Despite this, we will assume that utilities are stationary.

How utilities are calculated

- 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.

KING'S College LONDON

How utilities are calculated

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

How utilities are calculated

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

How utilities are calculated

- Some solutions are:
  - Proper policies
  - Average reward
  - Discounted rewards
- As follows . . .

How utilities are calculated

- Proper policies always end up in a terminal state eventually.
- Thus they have a finite expected utility.

How utilities are calculated

- We can compute the average reward per time step.
- Even for an infinite policy this will (usually) be finite.

How utilities are calculated

- With discounted rewards the utility of an infinite sequence is finite:

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

where $0 \leqslant \gamma < 1$ 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.

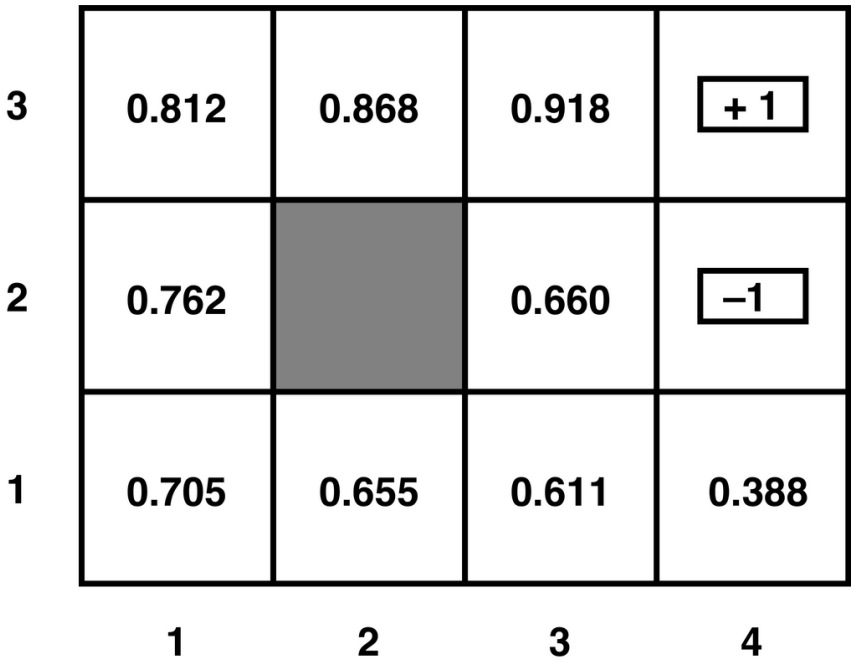
Optimal policies

- The optimal policy is then:

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

- It turns out that this is independent of the state the agent starts in.

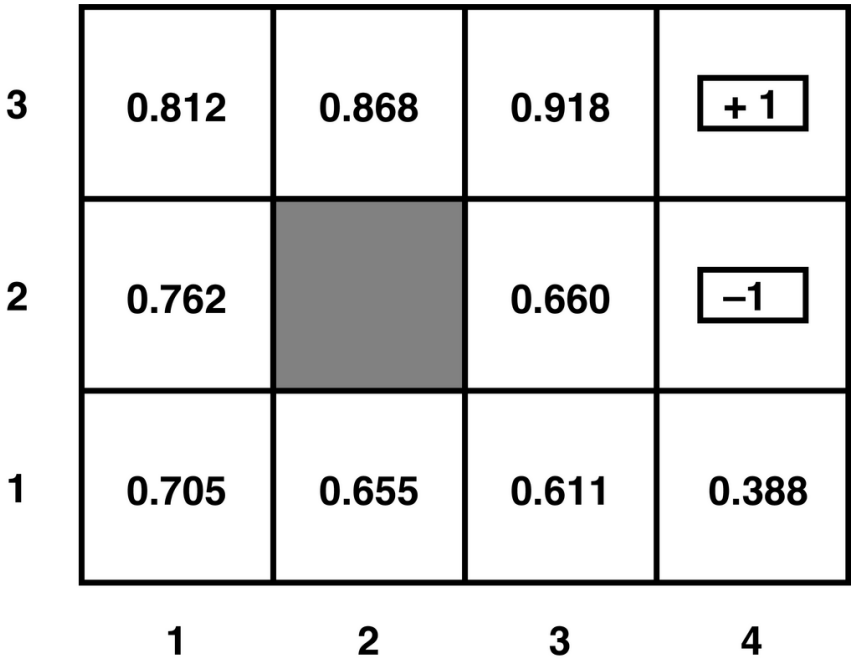
Optimal policies



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

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

Optimal policies

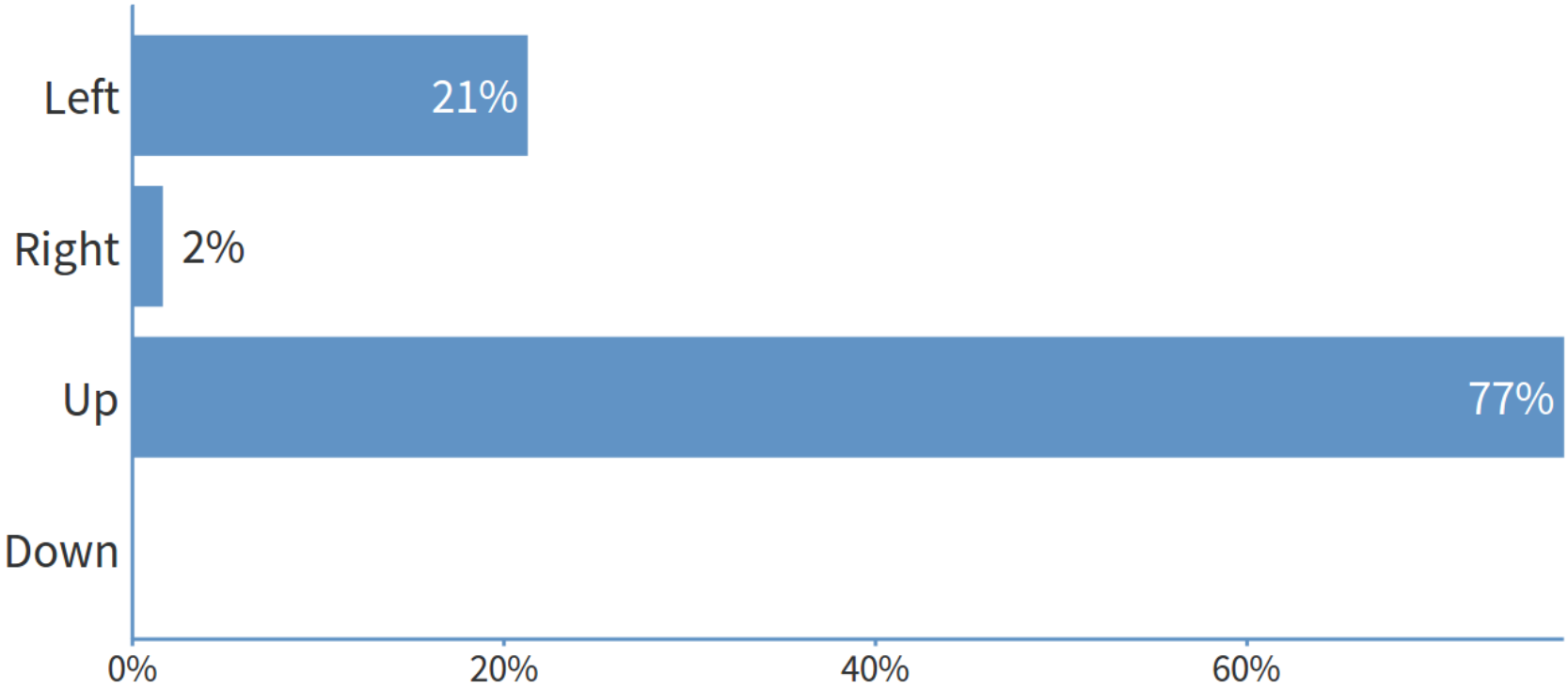


- Here we have the values of states if the agent executes an optimal policy

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

- What should the agent do if it is in (3, 1)?

What should the agent do?



Wrong!

Example

- The answer is *Left*.
- The best action is the one that maximises expected utility.
- (You have to calculate the expected utility of all the actions to see why Left is the best choice.)

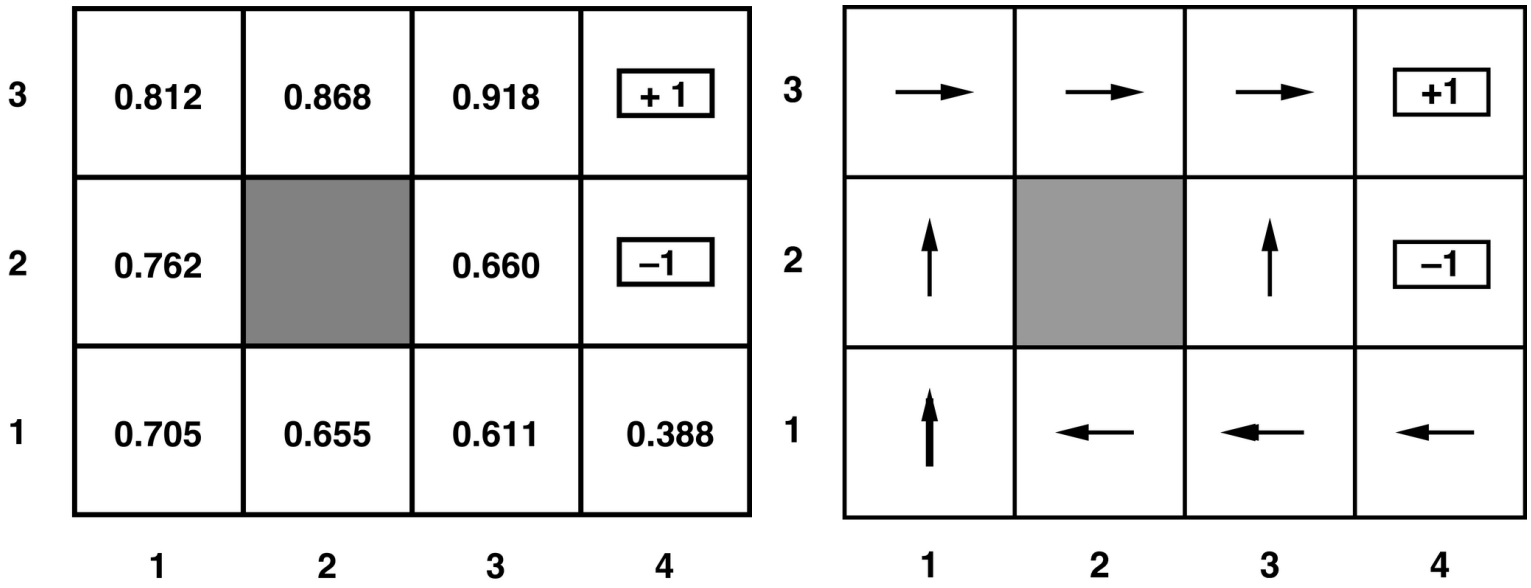
Optimal policies

- 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:

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

- Only have to consider the next step.
- The big question is how to compute $U^{\pi^*}(s)$.

Optimal policies



- Note that this is specific to the value of the reward $R(s)$ for non-terminal states — different rewards will give different values and policies.

Bellman equation

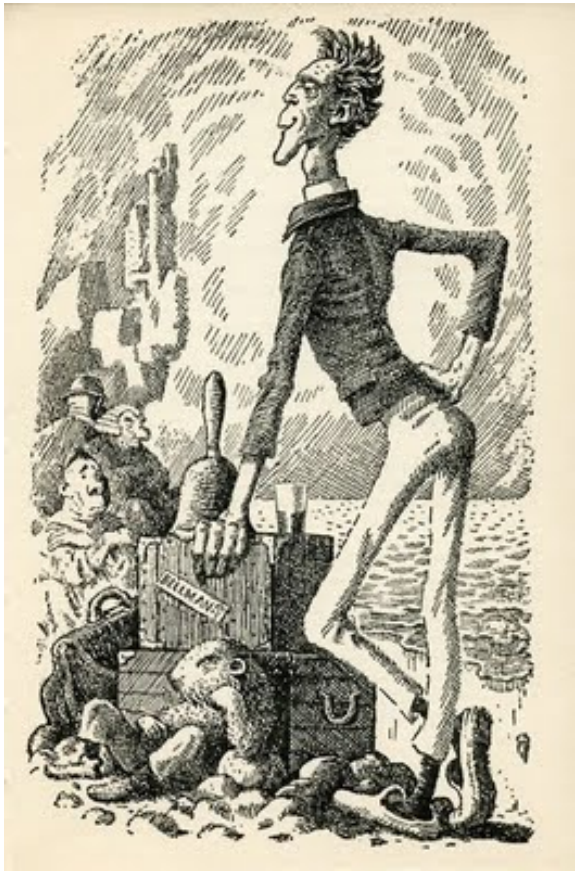
- How do we find the best policy (for a given set of rewards)?
- Turns out that there is a neat way to do this, by first computing the utility of each state.
- We compute this using the Bellman equation

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

- γ is a discount factor.

KING'S College LONDON

Not this Bellman



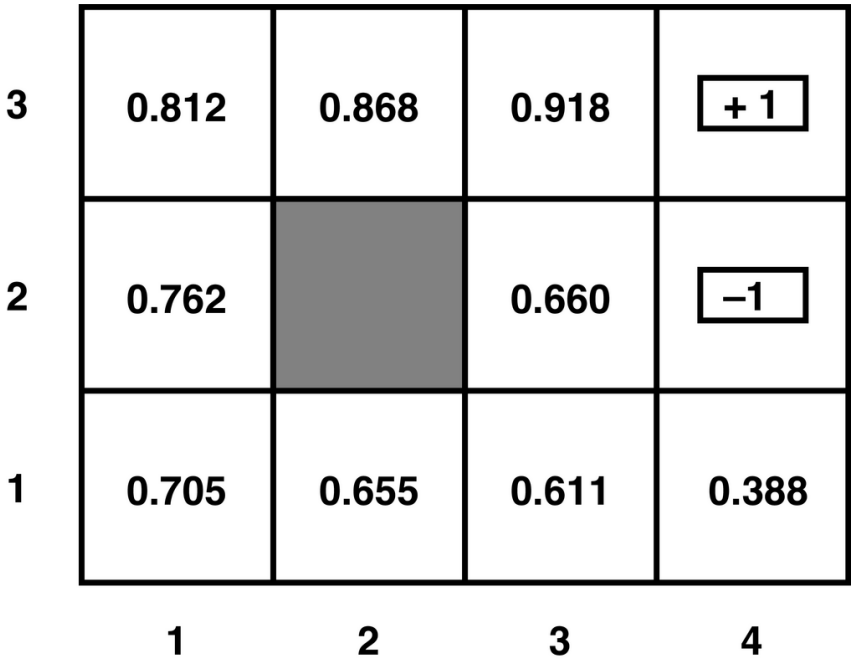
”Just the place for a Snark!” the Bellman cried,
As he landed his crew with care;
Supporting each man on the top of the tide
By a finger entwined in his hair.

”Just the place for a Snark! I have said it twice:
That alone should encourage the crew.
Just the place for a Snark! I have said it thrice:
What I tell you three times is true.”

Lewis Carroll

(Mervyn Peake’s illustrations to “The Hunting of the Snark”).

Bellman equation

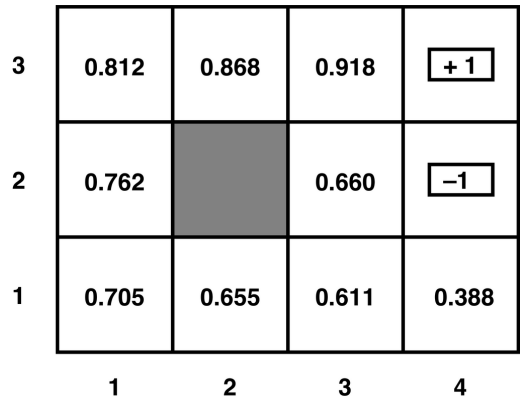


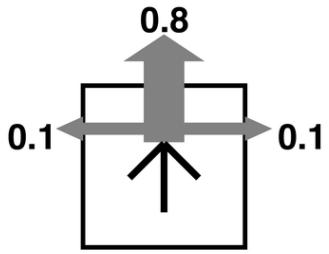
- Apply:

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

and we get:

Bellman equation





$$
\begin{aligned}
U(1,1) \quad = \quad & -0.04 + \\
& \gamma \, max[0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up) \\
& \qquad 0.9U(1,1) + 0.1U(1,2), && (Left) \\
& \qquad 0.9U(1,1) + 0.1U(2,1), && (Down) \\
& \qquad 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)] && (Right)
\end{aligned}
$$

Value iteration

- In an MDP wth n states, we will have n Bellman equations.

(Pendleton Ward/Cartoon Network)

- Hard to solve these simultaneously because of the max operation
  - Makes them non-linear

Value iteration

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

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

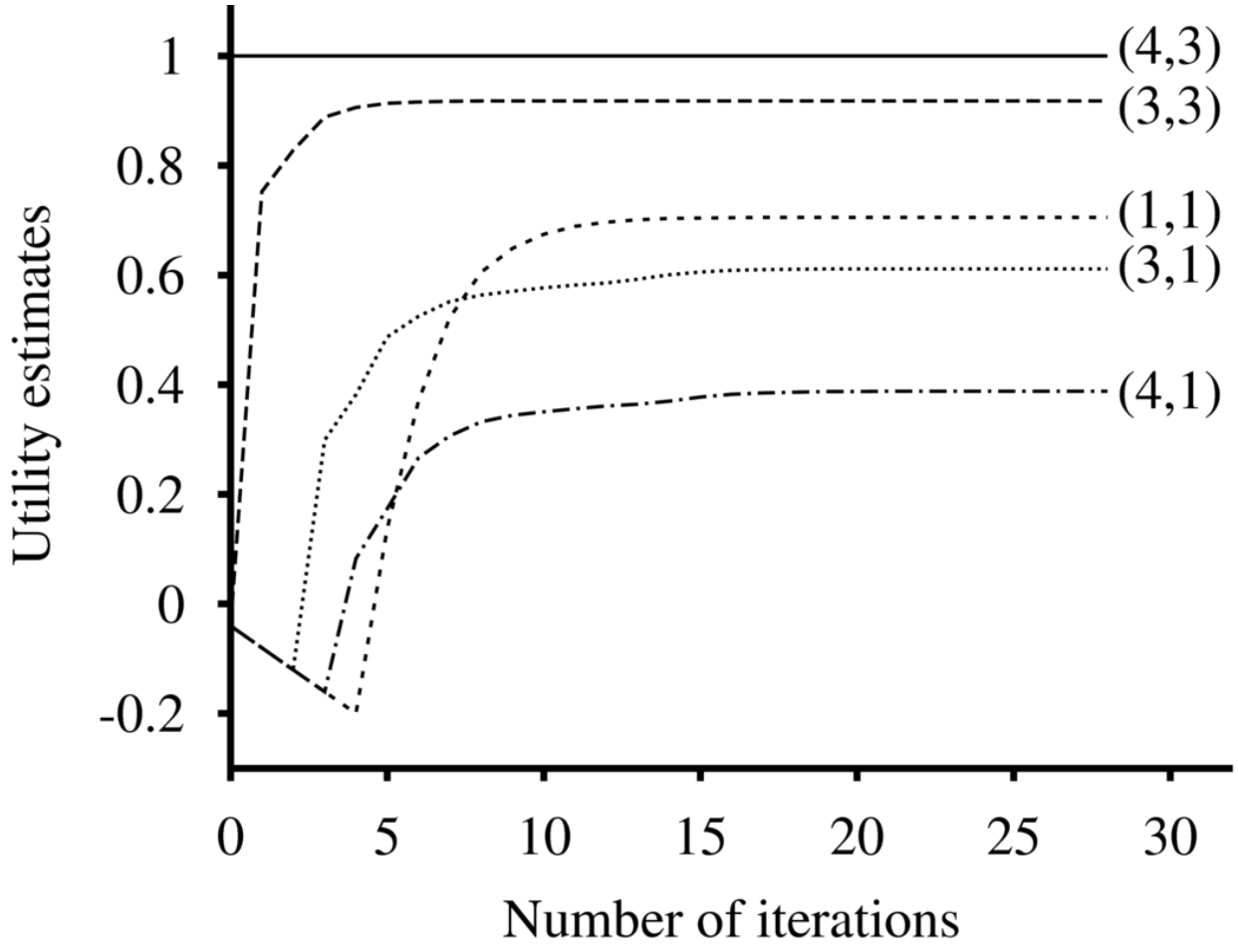
  simultaneously to all the states.
- Continue until the values of states do not change.
- After an infinite number of applications, the values are guaranteed to converge on the optimal values.

Value iteration

procedure VALUE ITERATION
    for s in S **do**
        $U(s) \leftarrow 0$
    end for
    repeat
        $U_{copy} \leftarrow U$
        for s in S **do**
            $U(s) \leftarrow R(s) + \gamma \max_{a \in A(S)} \sum_{s'} P(s'|s,a) U_{copy}(s')$
        end for
    until $U == U_{copy}$
end procedure

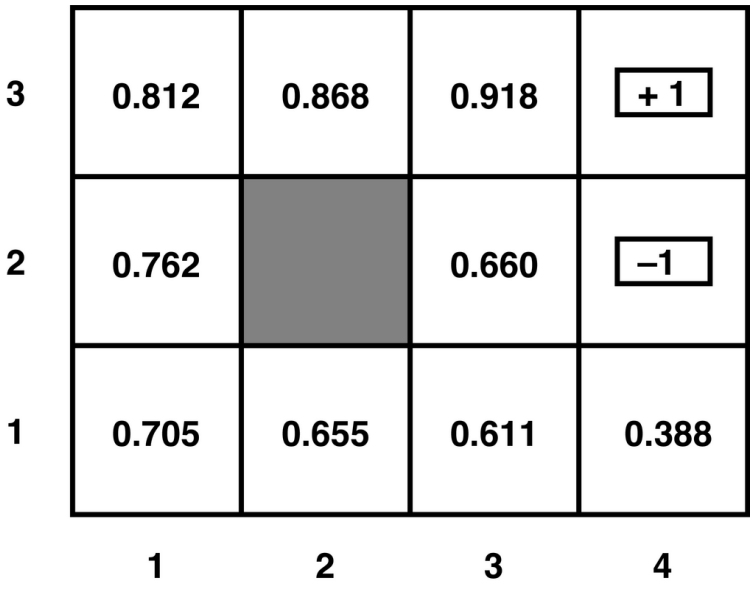
- States, S, reward, $R(s)$, set of actions, $A(s)$ and transition model, $P(s'|s,a)$, are exatcly as defined on page 39.
- γ is the discount factor, as described on page 49.

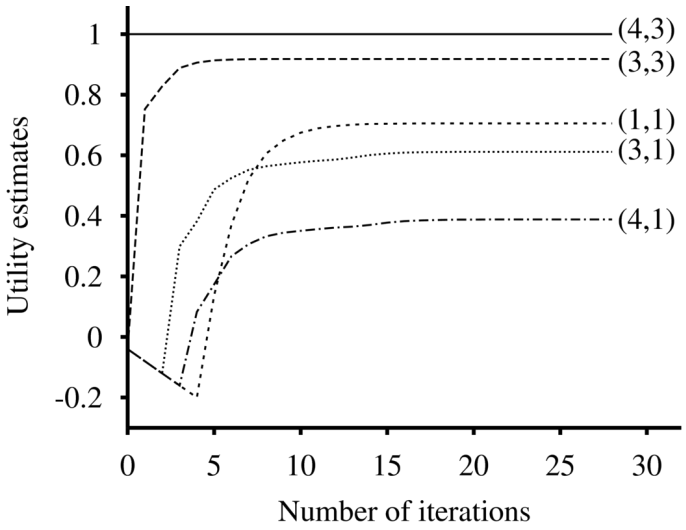
Value iteration



- How the values of states change as updates occur.

Value iteration





- $U(4,3)$ is pinned to 1.
- $U(3,3)$ quickly settles to a value close to 1
- $U(1,1)$ becomes negative, and then grows as positive utility from the goal feeds back to it.

Rewards

- The example so far has a negative reward $R(s)$ for each state.
- Encouragement for an agent not to stick around.
- Can also think of $R(s)$ is being the cost of moving to the next state (where we obatin the utility):

$$R(s) = -c(s, a)$$

  where s is the action used.
- Bellman becomes:

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

- Note that the action can be dependent on the state.

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 $U_i(s)$.
  - Policy improvement
    Given $U_i(s)$, compute π_{i+1}
- We will look at each of these steps in turn.
- But not in order.

Policy improvement

- Easy
- Calculate a new policy π_{i+1} by applying:

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

- For each state we do a one-step MEU lookahead.
- A simple decision.
- Use the values established by policy evaluation.

Policy evaluation

- 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.

Policy iteration

- Put these together to get:
- Starting from some initial policy π_0 we do:
  1. Policy evaluation
     Compute:

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

     for every state.
  2. Policy improvement
     Calculate a new policy π_{i+1} by applying:

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

     for every state s.

Until convergence.

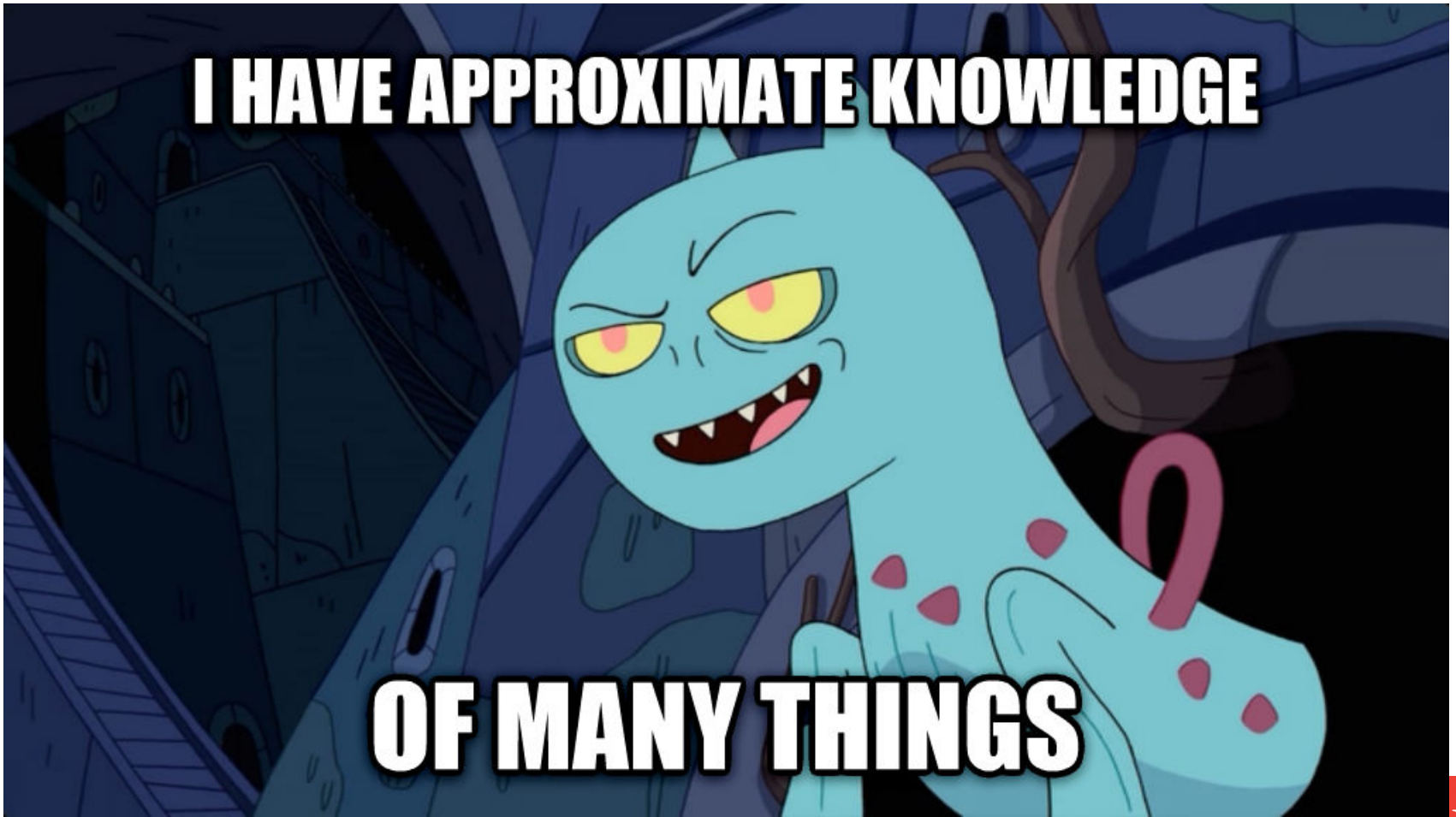
Policy iteration

- 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.

Policy evaluation

- There is a problem with the policy evaluation stage of the policy iteration approach.
- If we have n states, we have n linear equations with n unknowns in the evaluation stage.
- Solution in $O(n^3)$.
- For large n, can be a problem.
- So, an approximate solution.

Approximate?



(Pendleton Ward/Cartoon Network)

Approximate policy evaluation

- Run a simplified value iteration.
- Policy is fixed, so we know what action to do in each state.
- Repeat:

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

a fixed number of times.

Modified policy iteration

- Starting from some initial policy π_0 we do:
  1. Approximate policy evaluation
     Repeat
     $$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$
     a fixed number of times.
  2. Policy improvement
     $$\pi_{i+1}(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'|s, a) U_i(s')$$
     for every state s.

  Until convergance
- Often more efficient than policy iteration or value iteration.

Solving MDPs

- Have covered three methods for solving MDPs
  - Value iteration
    (Exact)
  - Policy iteration
    (Exact)
  - Modified policy iteration
    (Approximate)
- Which to use is somewhat problem specific.

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

Bellman redux

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.

Limitations of MDPs?

(Pendleton Ward/Cartoon Network)

KING'S College LONDON

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.

Partially observable MDPs

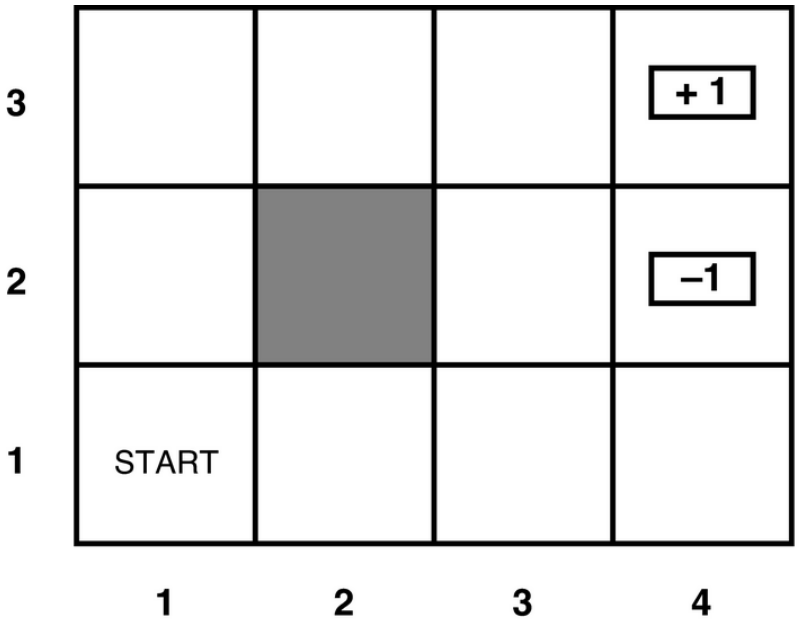
- Basic addition to the MDP model is the sensor model:

$$P(e|s)$$

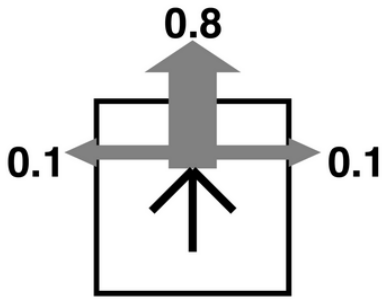
  probability of perceiving e in state s.
- As a result of noise in the sensor model, the agent only has a belief about which state it is in.
- Probability distribution over the possible states.
- "The world is a POMDP"

KING'S College LONDON

Partially observable MDPs



(a)



(b)

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

KING'S College LONDON

Partially observable MDPs

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

Partially observable MDPs

- The agent can compute its current belief as the conditional probability distribution over the states given the sequence of actions and percepts so far.
- We will come across this task again in temporal probabilistic reasoning.
- Filtering
- Computing the state that matches best with a stream of evidence.

Partially observable MDPs

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

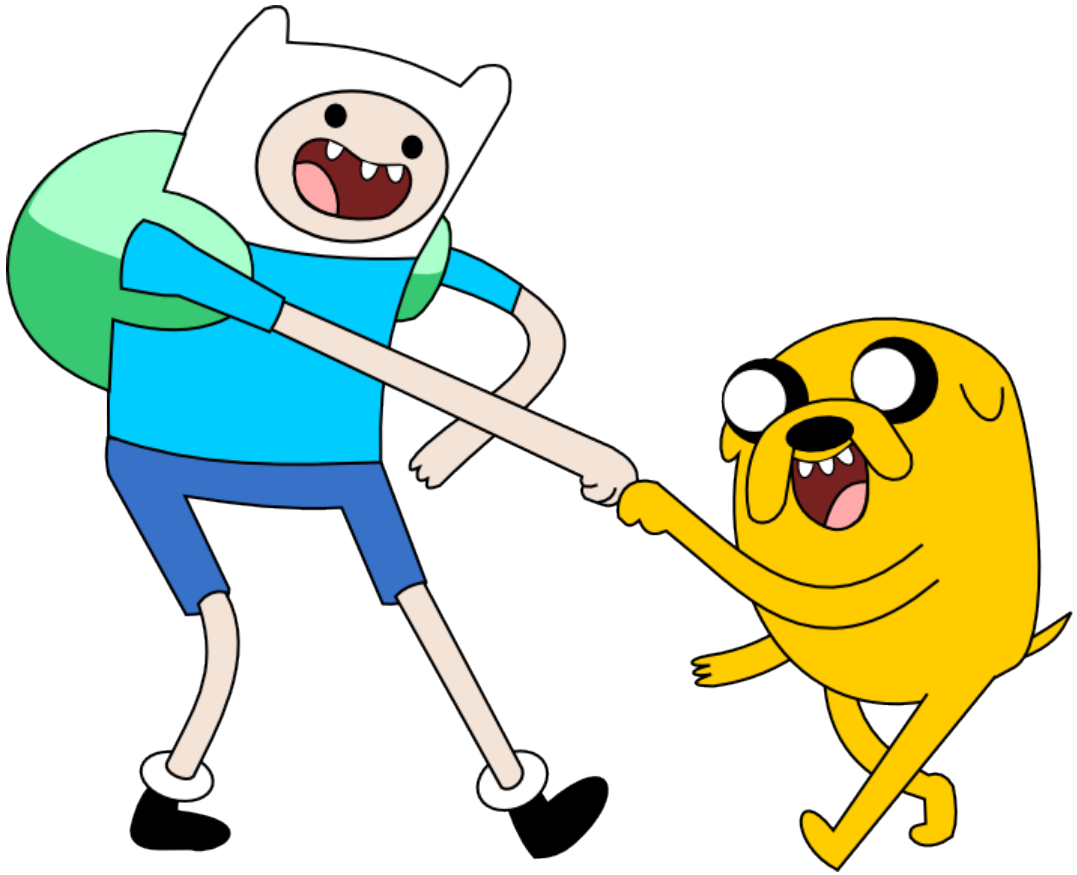
$$b'(s') = \alpha P(e|s') \sum_s P(s'|s,a)b(s)$$

- 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.

Partially observable MDPs

- 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.

Mathematical!



(Pendleton Ward/Cartoon Network)

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.
- This built on the last lecture on probability, and extended that with expected values.
- 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.

KING'S College LONDON