Lecture 8: Markov Decision Processes

Simon Parsons

School of Computer Science
University of Lincoln

Version 1.1

UNIVERSITY OF LINCOLN

Cover

UNIVERSITY OF
LINCOLN

Test

UNIVERSITY OF
LINCOLN

Vaccinate

UNIVERSITY OF
LINCOLN

Isolate

UNIVERSITY OF
LINCOLN

Today

- Quantifying uncertainty
  - Intro to probability, etc.
- Probabilistic reasoning
  - Bayesian networks, causal inference, etc.
- Making complex decisions
  - Intro to complex decision making
  - Markov Decision Processes
  - Reinforcement learning
- Reasoning over time
  - Hidden Markov Models, etc.
- Strategic reasoning
  - Game theory

UNIVERSITY OF LINCOLN

Module Survey

Lincoln STUDENTS' UNION

UNIVERSITY OF LINCOLN

MODULE EVALUATION

Please take part in this short Module Evaluation survey.

It only takes five minutes to complete and your feedback will help us to do more of what you like, as well as improve areas where you think we need to develop.

YOUR MODULE CODE IS CMP9132M

To take part please use the QR code or web address lncn.ac/semamodval to access the survey_

UNIVERSITY OF LINCOLN

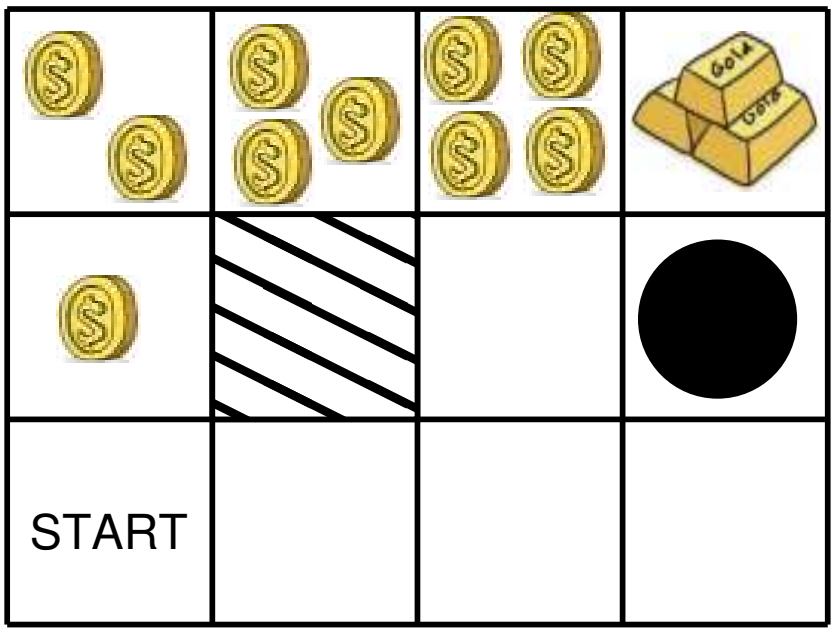
Recap

- The last lecture covered three main ideas.
- First, and most extensively we talk about simple decision making.
  - One shot decisions
  - Should I eat pie or broccoli?
- We also discussed the difficulty of making sequences of decisions, since the greedy approach is often unhelpful.
- Finally we suggested that the utility/reward distinction could be helpful.

UNIVERSITY OF LINCOLN

Recap: reward v. utility

- In the dungeon utility is how much being in a location is worth to you:



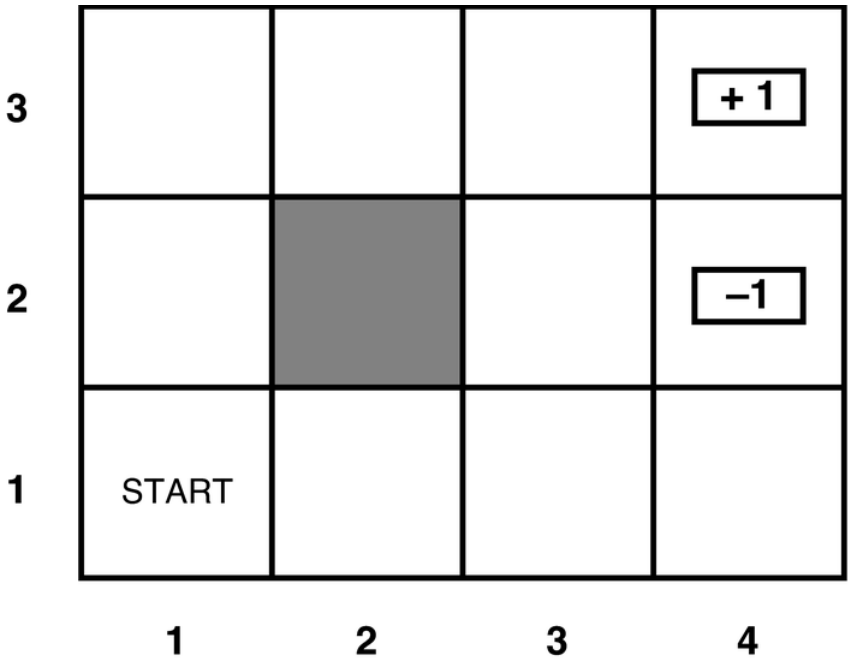
- Utility factors in potential future rewards.

Today

- Question: How do we do this?
- Answer: Markov decision process (MDP).

UNIVERSITY OF LINCOLN

An example

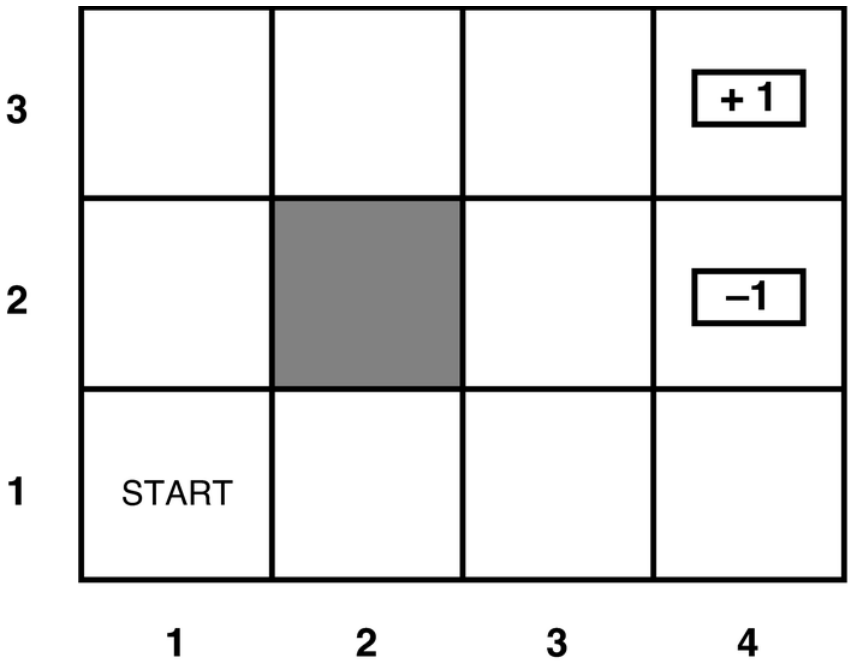


- Here the agent has to pick a sequence of actions.

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

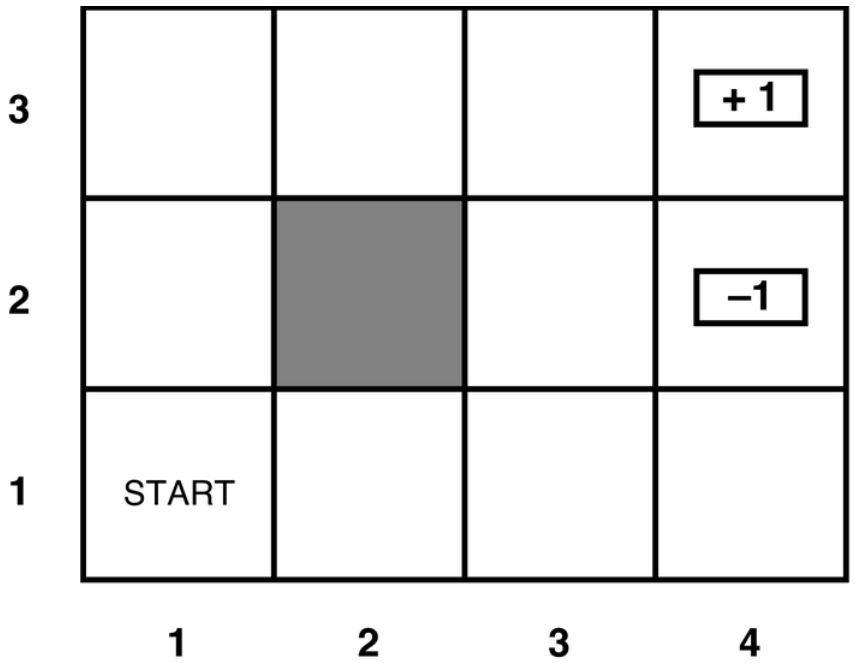
to get from the START to the +1 terminal state.

An example



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

An example



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

$$\mathit{Up}, \mathit{Up}, \mathit{Right}, \mathit{Right}, \mathit{Right}$$

- But actions are stochastic.

UNIVERSITY OF LINCOLN

Aside

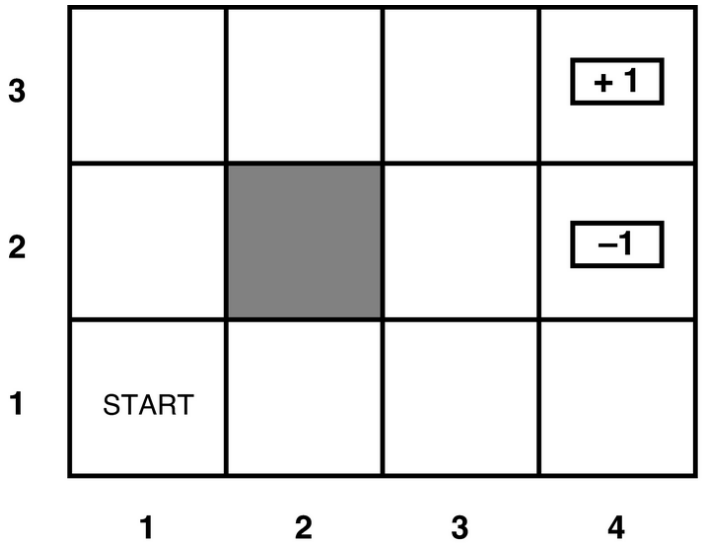
(https://diamondgeezer.blogspot.com/2015/02/five-london-hedge-mazes.html)

- Environment is *fully observable* and *stochastic*.
- Partial observablility comes later.
- Stochastic is a generalisation of deterministic.

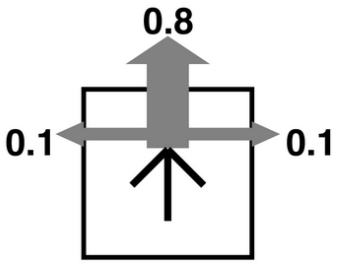
  (Easy to get deterministic from stochastic).

UNIVERSITY OF LINCOLN

An example



(a)

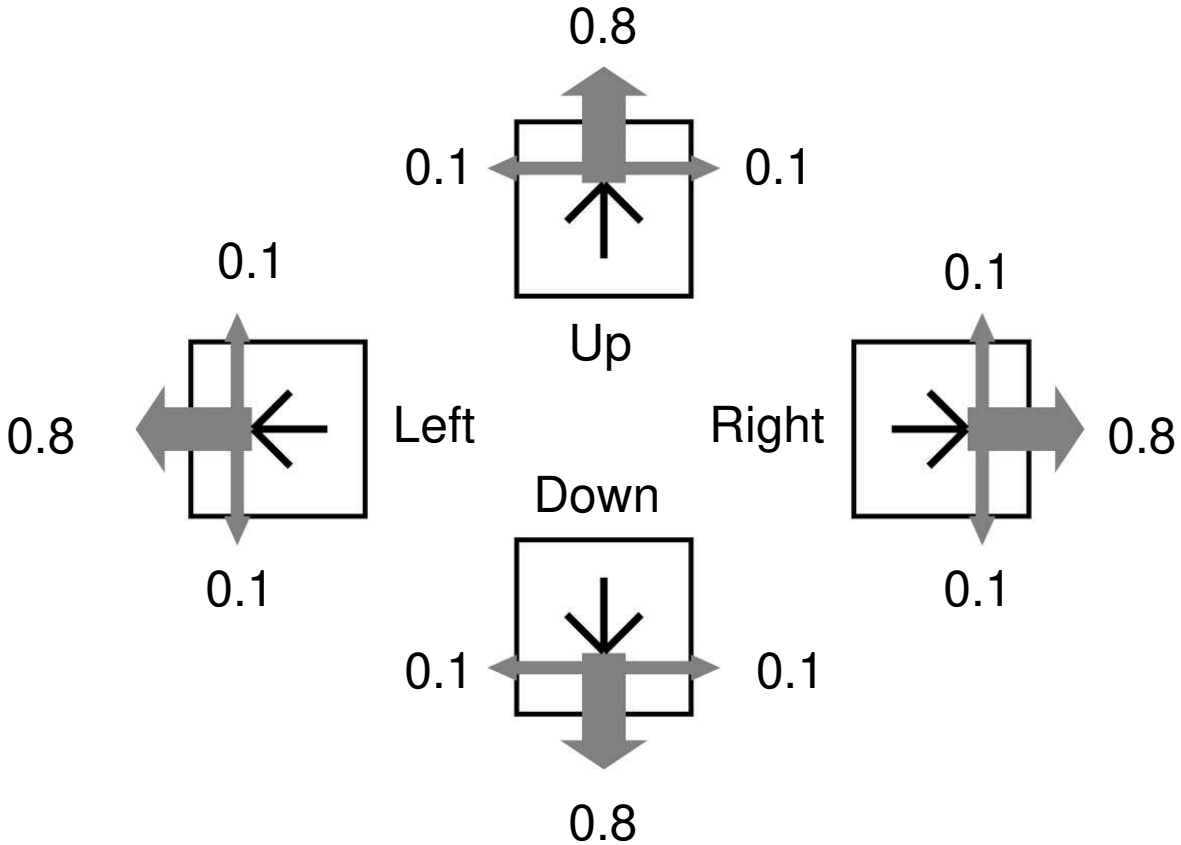


(b)

- 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.

UNIVERSITY OF LINCOLN

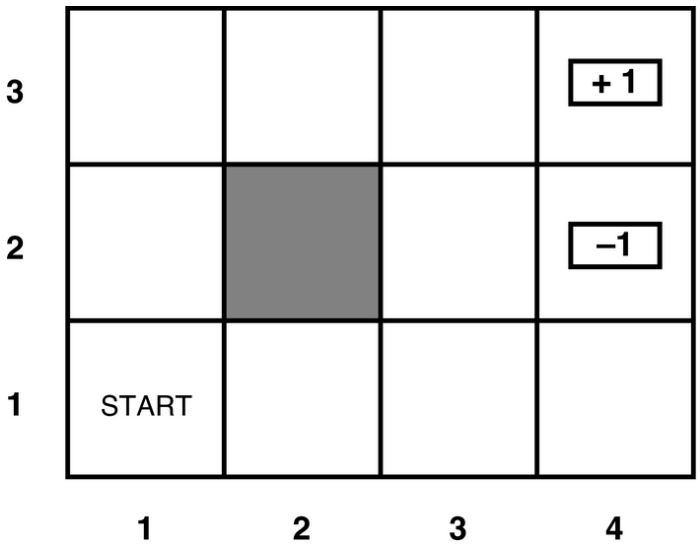
Motion model



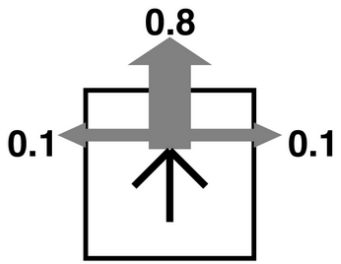
- 80% of the time the agent moves as intended.
- 20% of the time the agent moves perpendicular to the intended direction. Half the time to the left, half the time to the right.
- The agent doesn't move if it hits a wall.

UNIVERSITY OF LINCOLN

Going up?



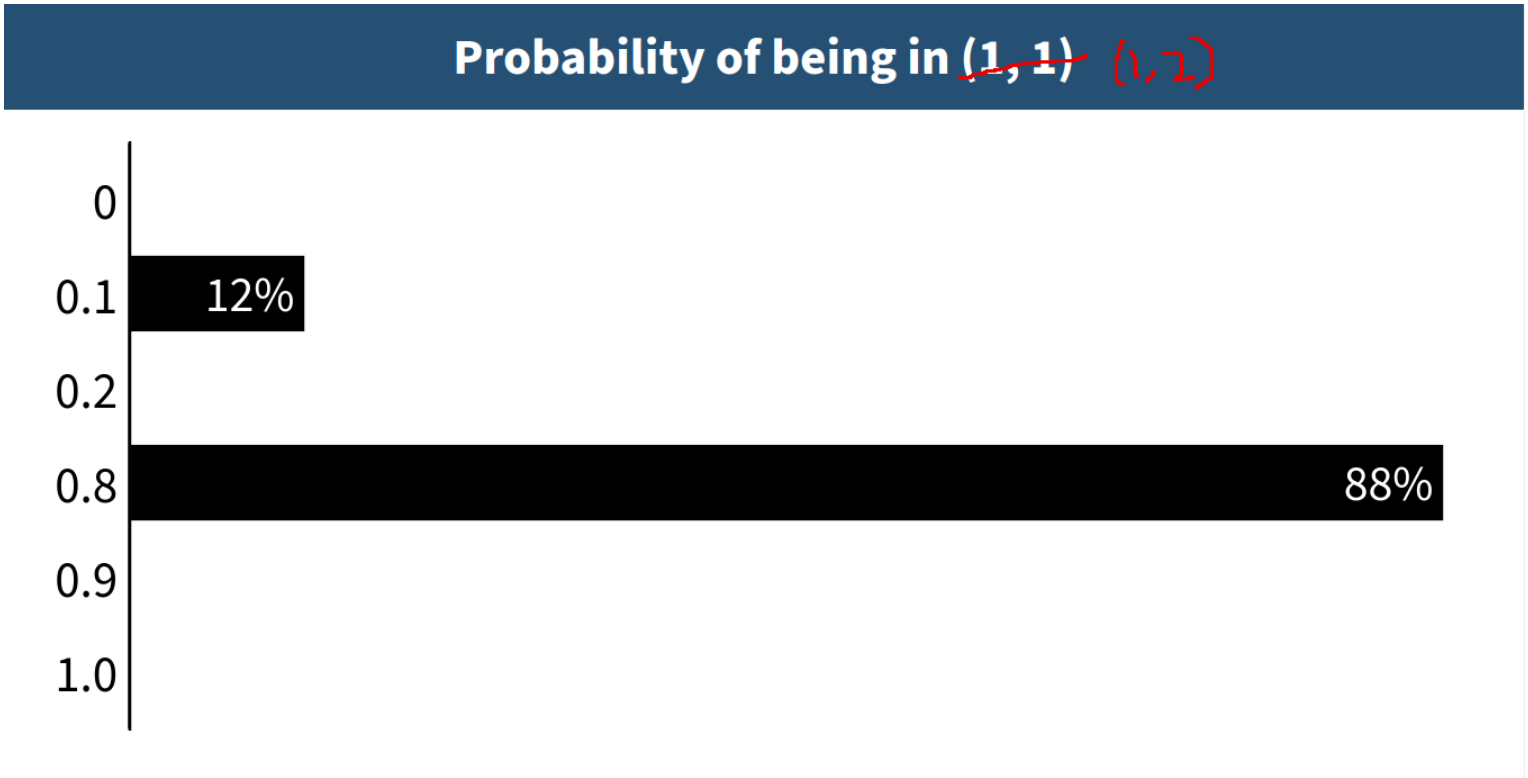
(a)



(b)

- So, if it in (1, 1) and goes Up, what is the probability it ends up in (1, 2)?

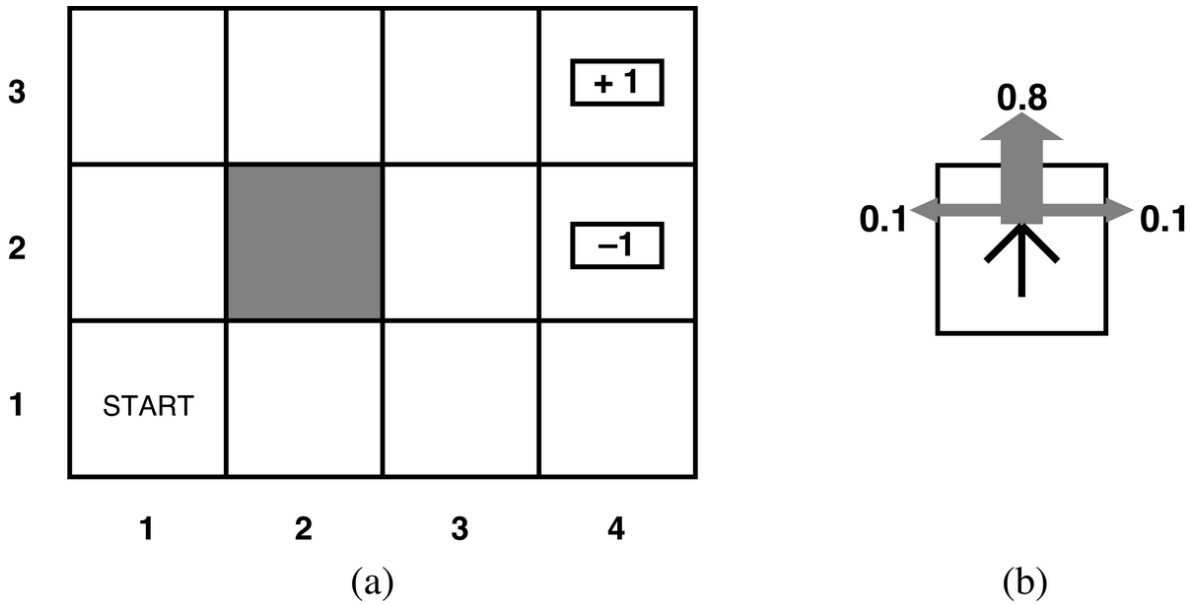
Going up?



- 0.8 is the right answer.

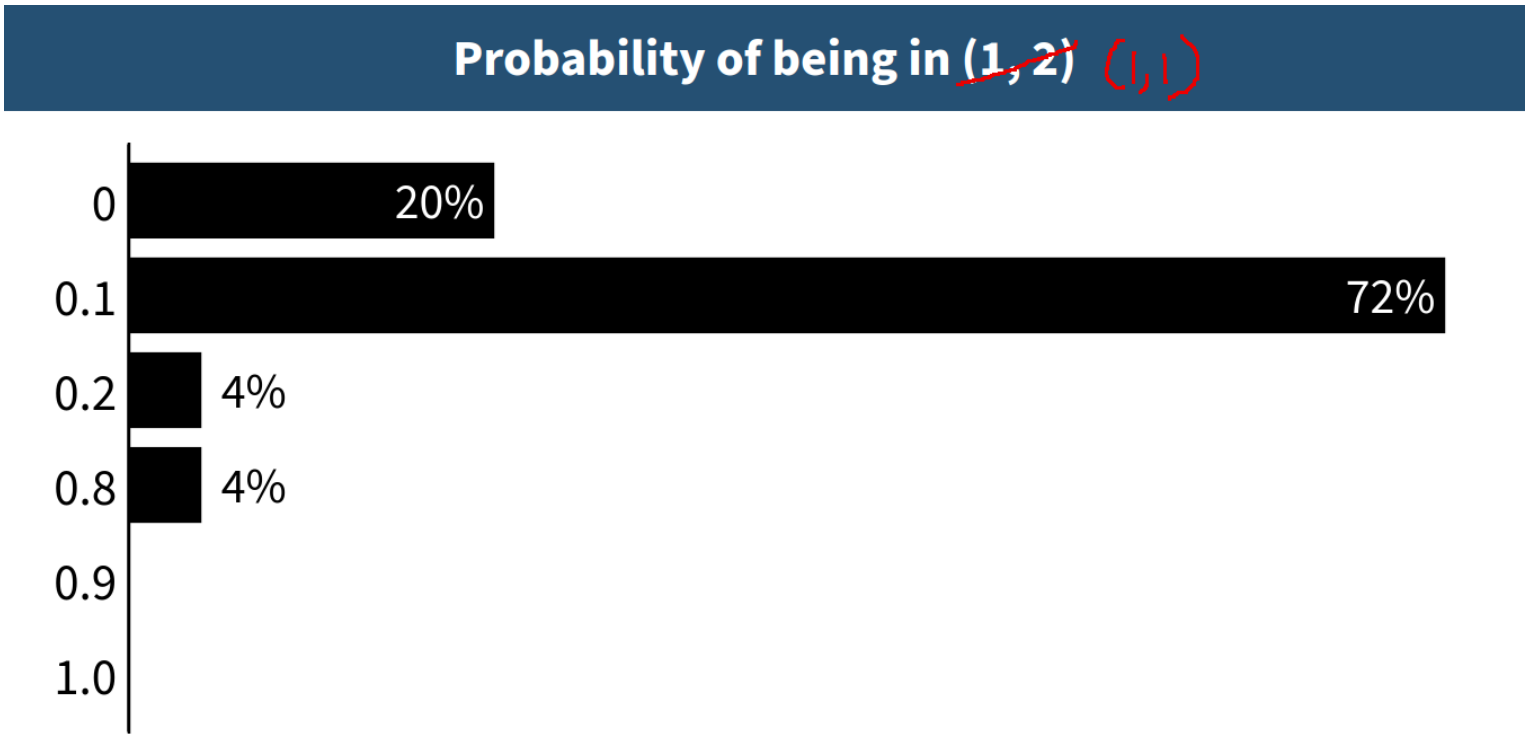
UNIVERSITY OF LINCOLN

Going up?



- So, if it in (1, 1) and goes Up, what is the probability it ends up in (1, 1)?

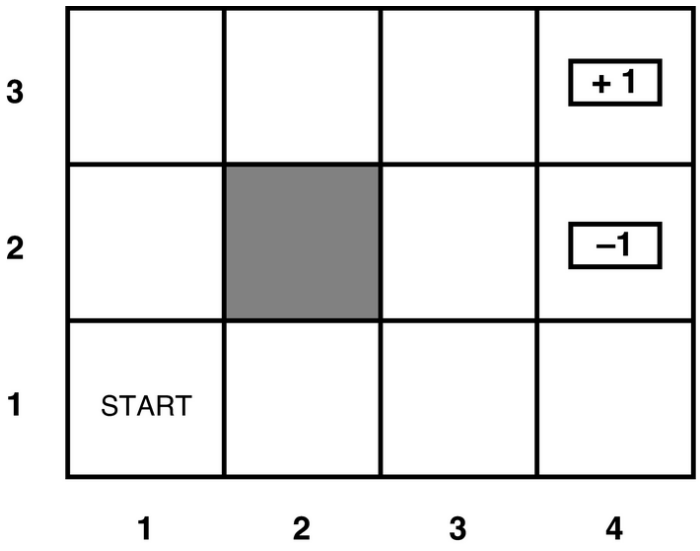
Going up?



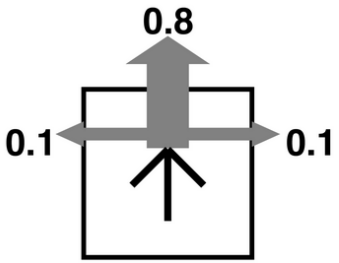
- 0.1 is the right answer.

UNIVERSITY OF LINCOLN

Going right?



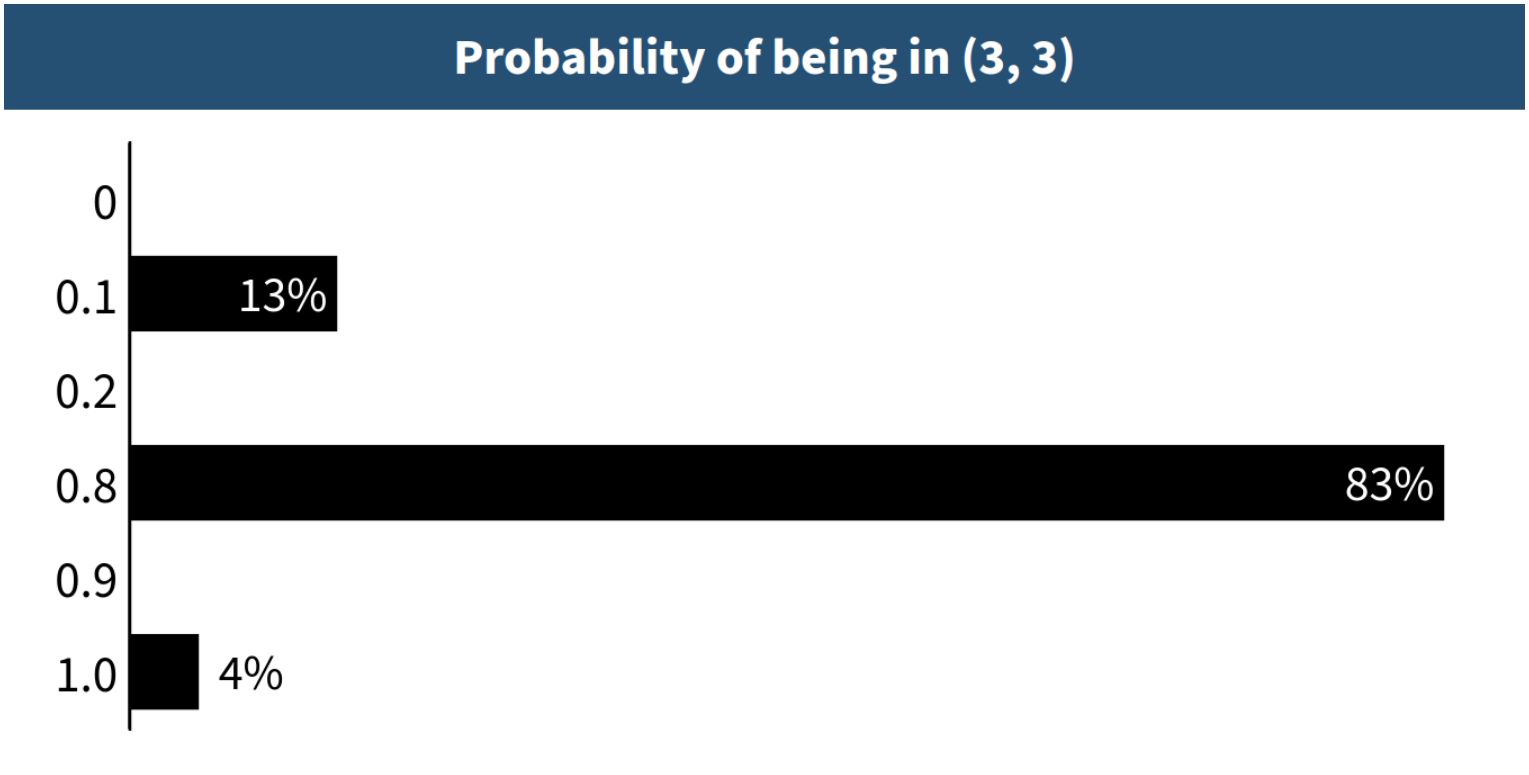
(a)



(b)

- So, if it in (2, 3) and goes Right, what is the probability it ends up in (3, 3)?

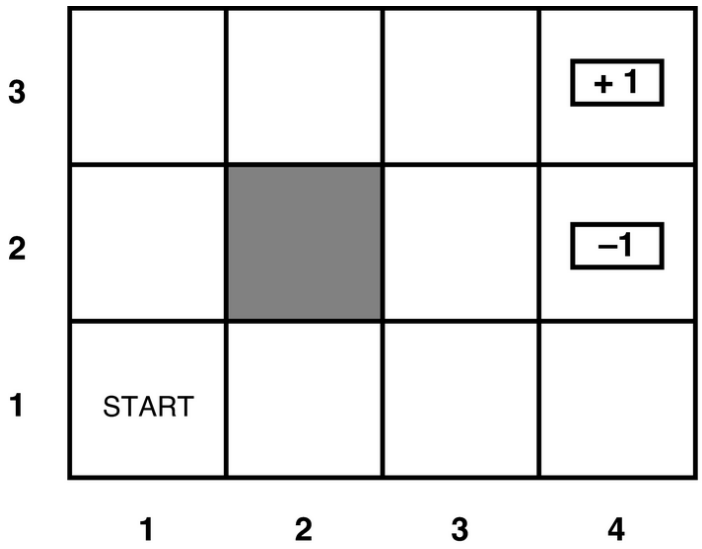
Going right?



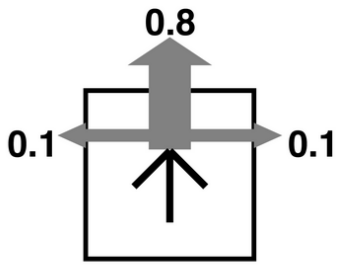
- 0.8 is the right answer.

UNIVERSITY OF LINCOLN

An example



(a)

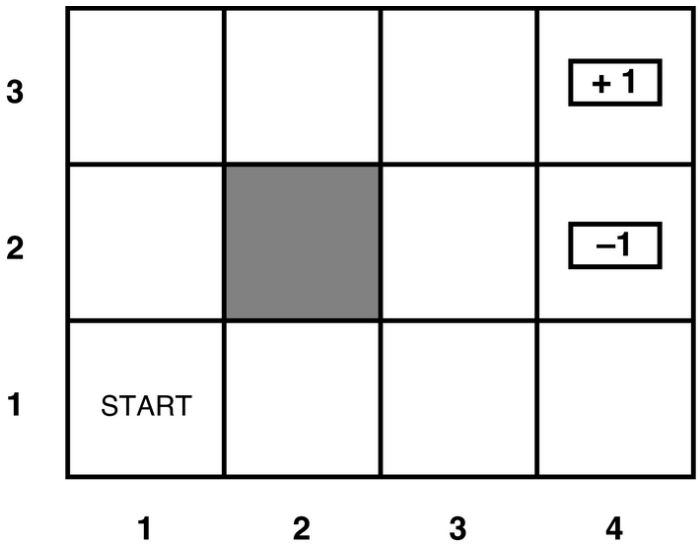


(b)

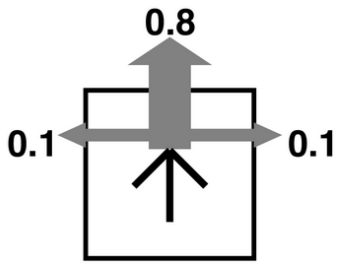
- Because of the stochastic actions, *Up*, *Up*, *Right*, *Right*, *Right* succeeds with probability:

$$0.8^5 = 0.32768$$

An example



(a)



(b)

- 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.

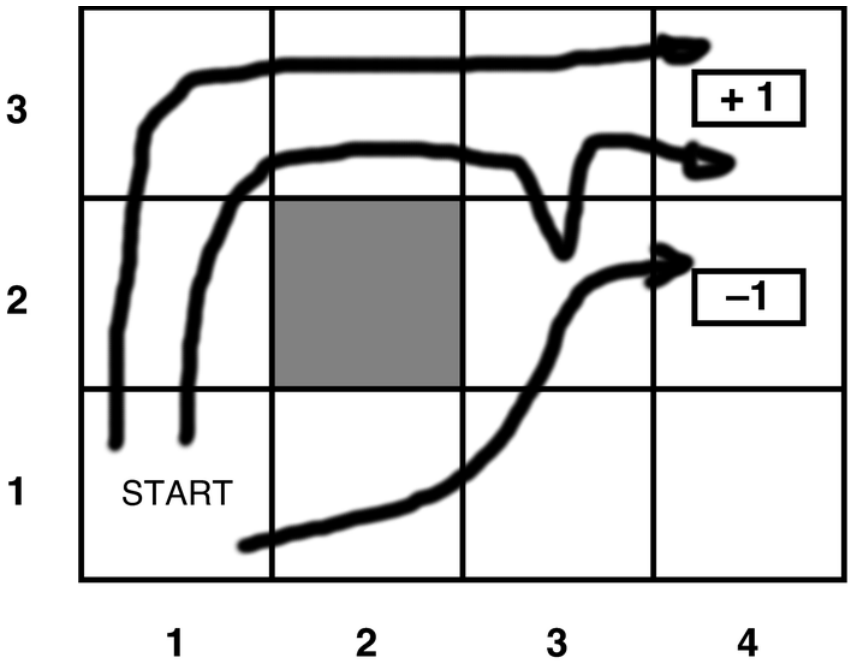
UNIVERSITY OF LINCOLN

An example

- The full description of the problem also has to include the utility function.
- This is defined over sequences of states — what we call *runs*.
- We will assume that in each state s the agent receives a reward $R(s)$.
- This may be positive or negative.

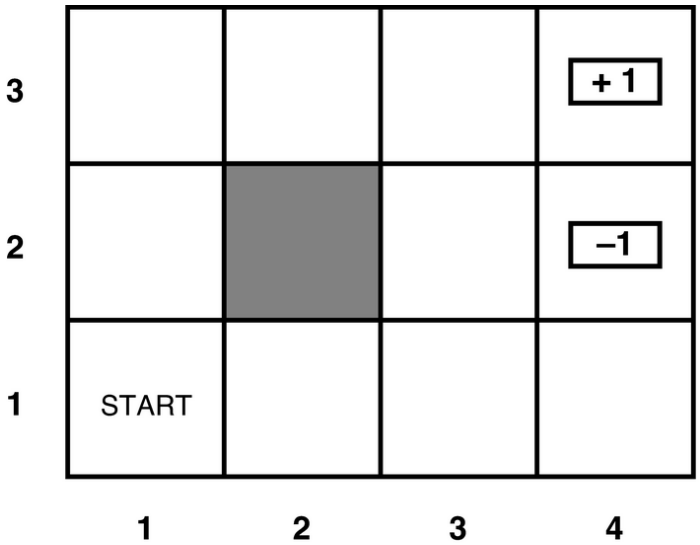
UNIVERSITY OF LINCOLN

Runs

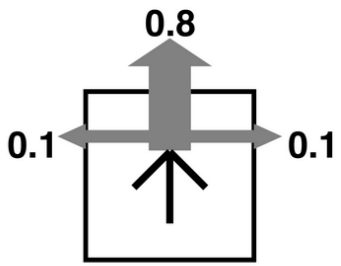


- Three runs.

An example



(a)

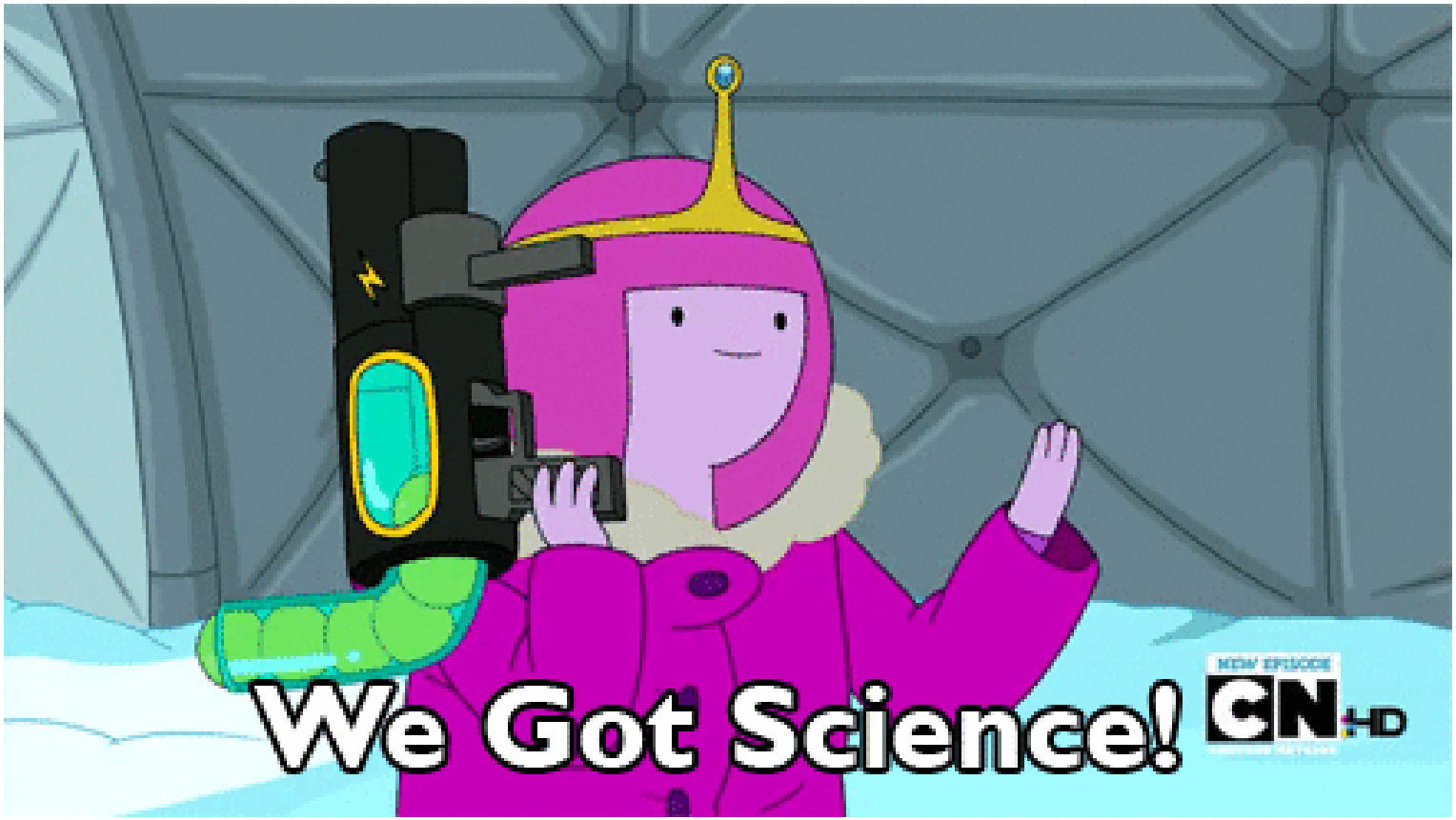


(b)

- The reward for non-terminal states is -0.04.
- We will assume that the utility of a run is the sum of the rewards 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).

UNIVERSITY OF LINCOLN

How do we tackle this?



(Pendleton Ward/Cartoon Network)

UNIVERSITY OF LINCOLN

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

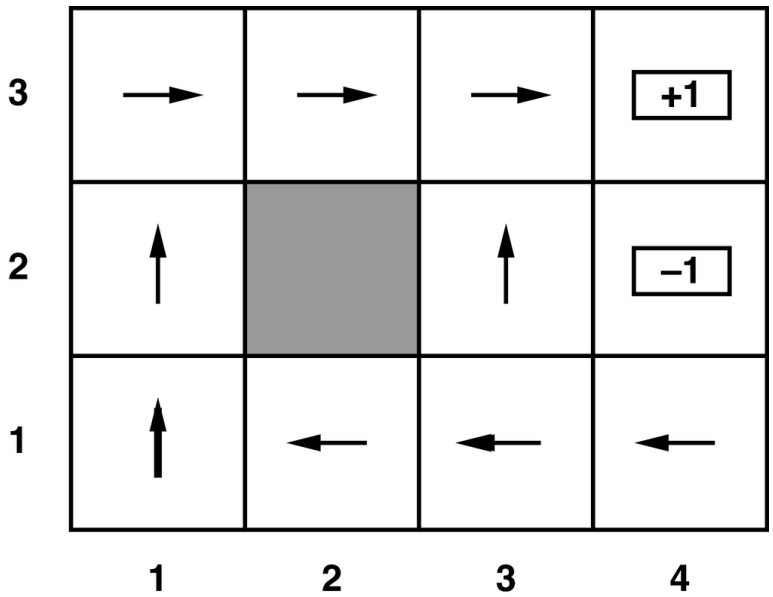
UNIVERSITY OF LINCOLN

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.

$$\pi(s) = a, a \in A(s)$$

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 utility they generate.
- Since actions are stochastic, policies won't give the same utility every time.
  - So compare the expected utility.
- The optimum policy π^* is the policy with the highest expected utility.
- At every stage the agent should do $\pi^*(s)$.

UNIVERSITY OF LINCOLN

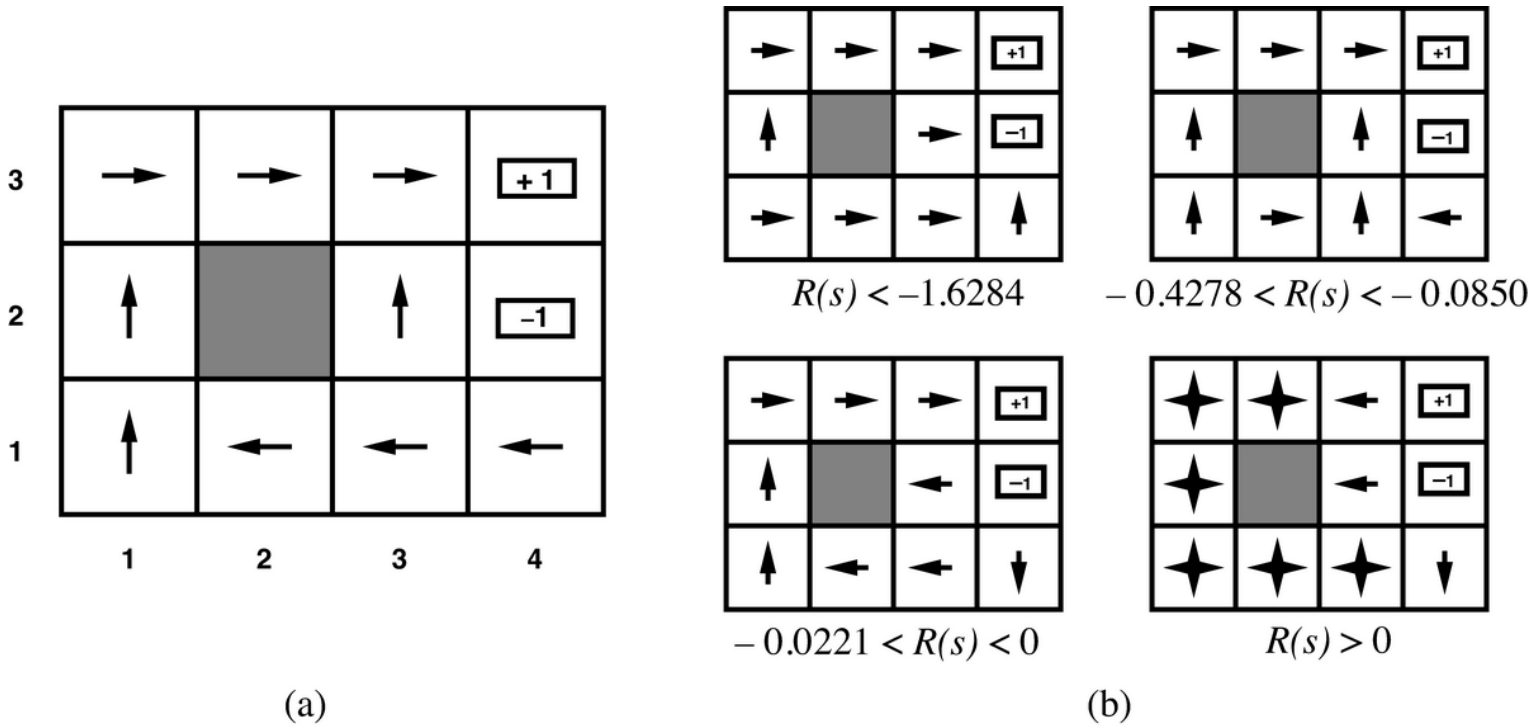
Markov decision process

(40 Acres and a Mule Filmworks/Universal Pictures)

- $\pi^*(s)$ is the right thing.
- But now it is the best action in the best run, not the best myopic single action.

UNIVERSITY OF LINCOLN

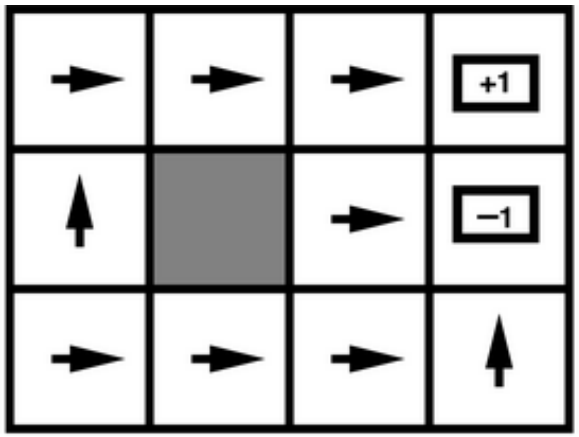
An example



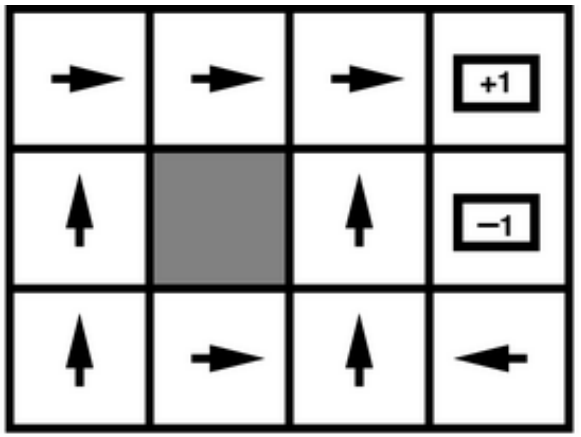
(a) Optimal policy for the original problem.

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

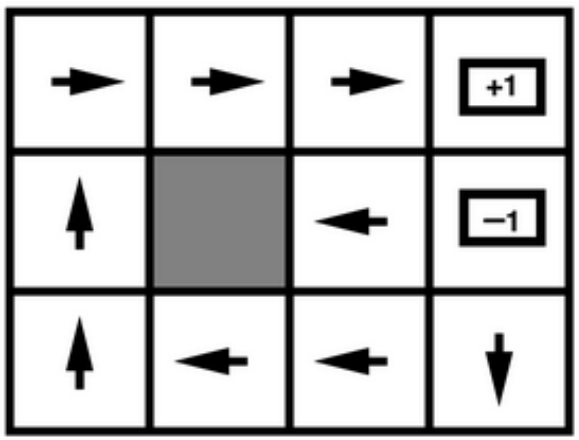
Different rewards



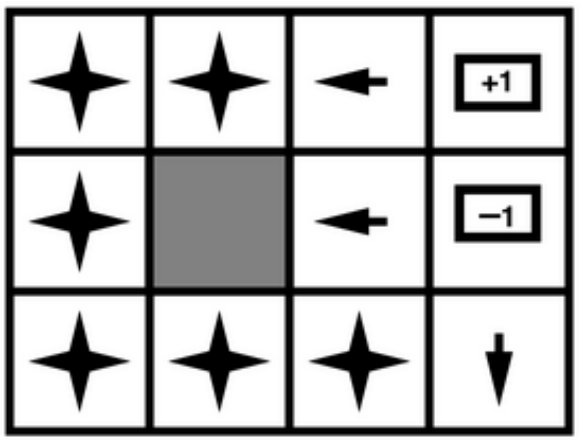
$R(s) < -1.6284$



$-0.4278 < R(s) < -0.0850$



$-0.0221 < R(s) < 0$



$R(s) > 0$

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

- In general we need to compute $U_r([s_0, s_1, \ldots, s_n])$.
- Can just sum rewards along a run to get the utility of the state the run starts from.
  - Not the only way.
- 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.

UNIVERSITY OF LINCOLN

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 AAI module to failing it

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

But are they?

- Not clear that utilities are always stationary.

(pillsbury.com)

- 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.

UNIVERSITY OF LINCOLN

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:

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

  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

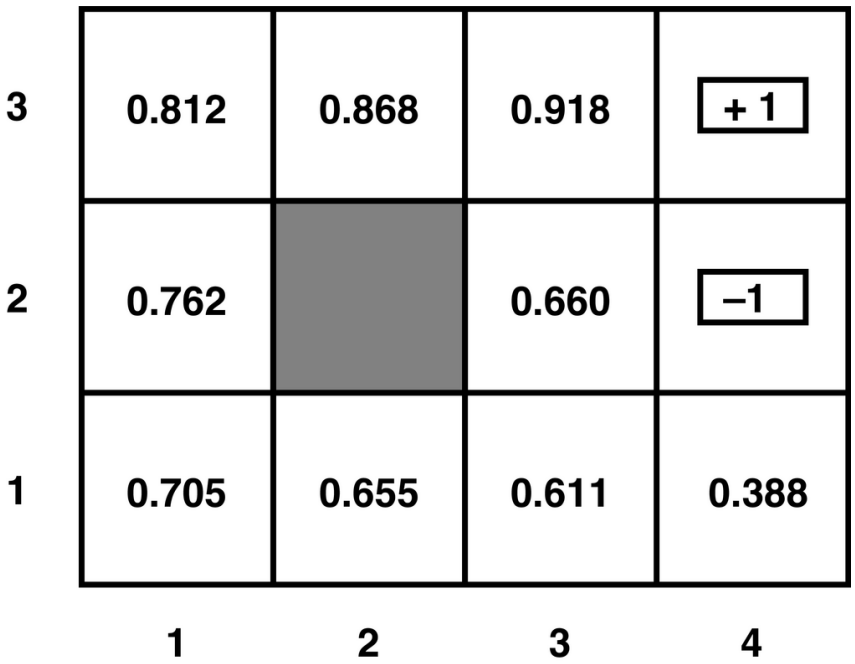
- The optimal policy is then:

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

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

UNIVERSITY OF LINCOLN

Optimal policies



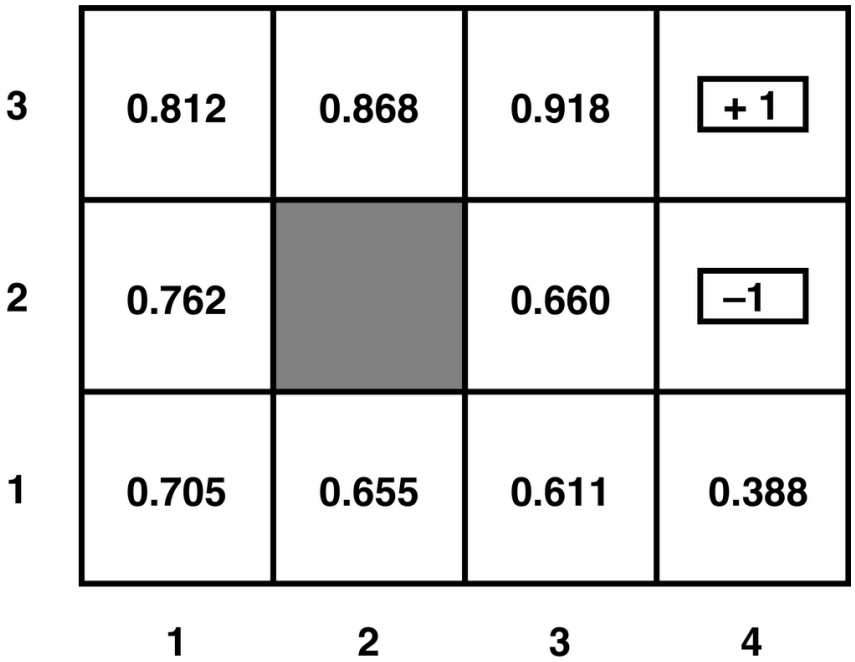
- Here we have the utilities of states if the agent executes an optimal policy

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

- "Optimal utilities"

UNIVERSITY OF LINCOLN

Optimal policies

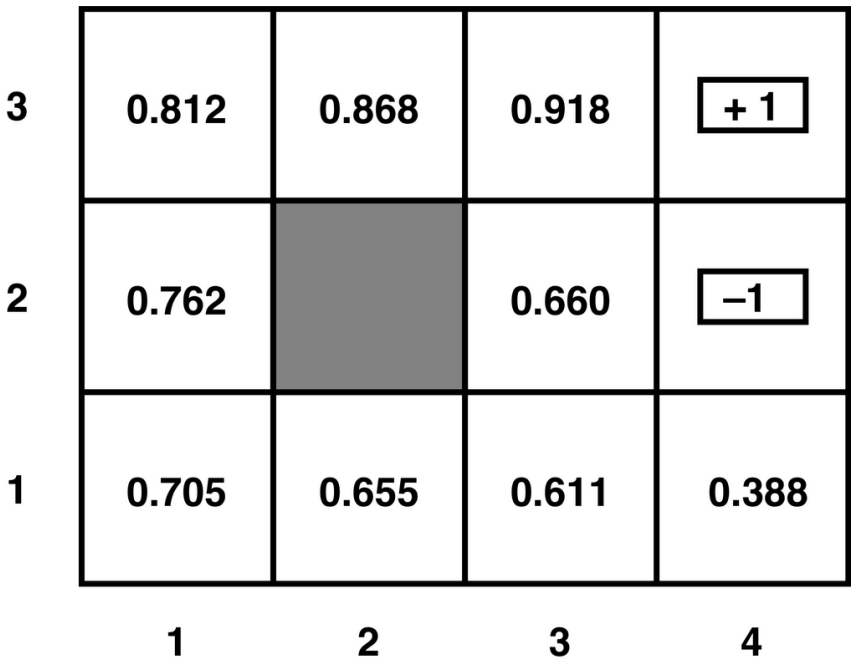


- Here we have the values of states if the agent executes an optimal policy

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

- Yes, we saw these values before.

Optimal policies

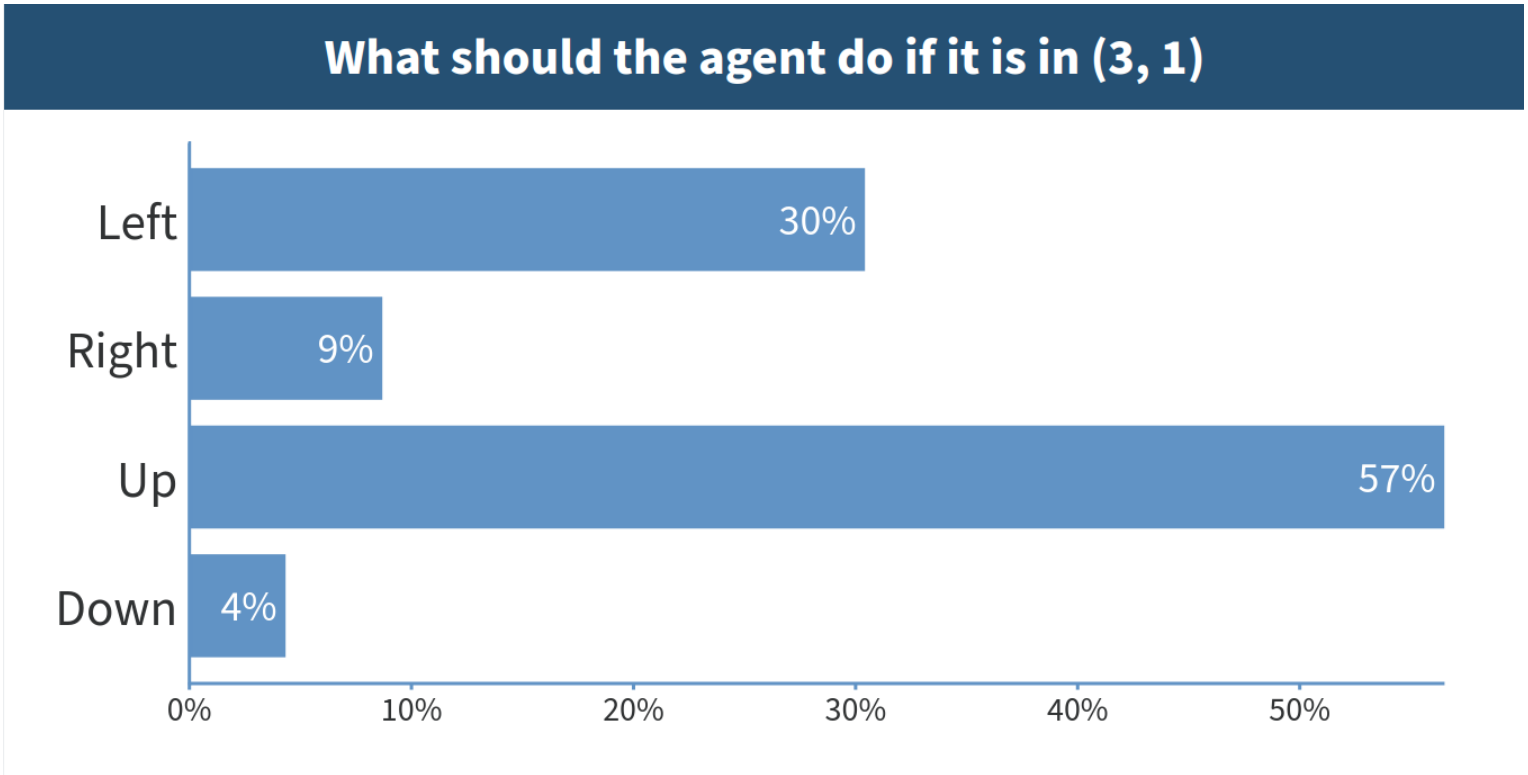


- Here we have the values of states if the agent executes an optimal policy

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

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

What should the agent do?



- Wrong!

UNIVERSITY OF LINCOLN

What should the agent do?

- 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.)

UNIVERSITY OF LINCOLN

Optimal policies

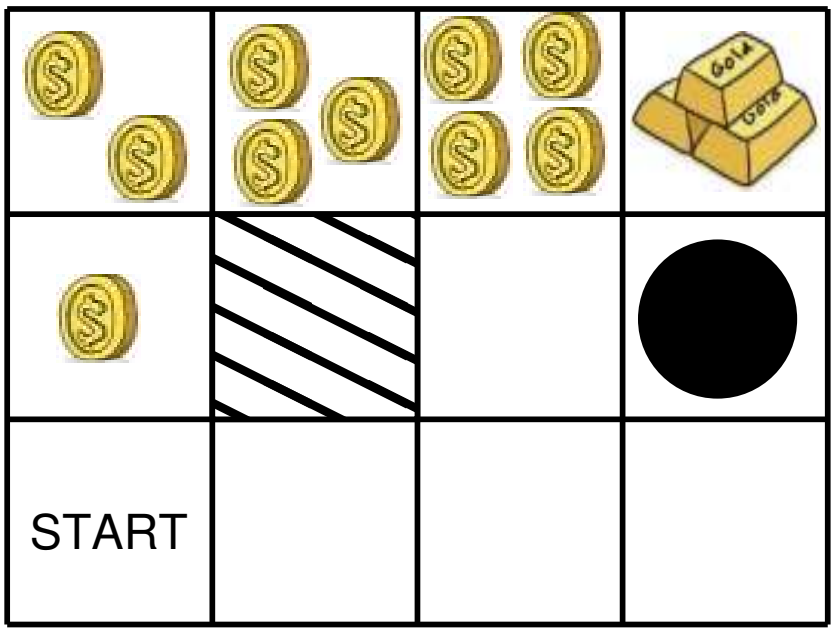
- If we have these values, the agent has a simple decision process
- It just picks the action a that maximises the expected utility of the next state:

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

- Only have to consider the next step.
- This is just what we wanted.

UNIVERSITY OF LINCOLN

Reward v. utility



- We just make the greedy choice at every step.

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.
- This is just what we wanted.

UNIVERSITY OF LINCOLN

Optimal policies

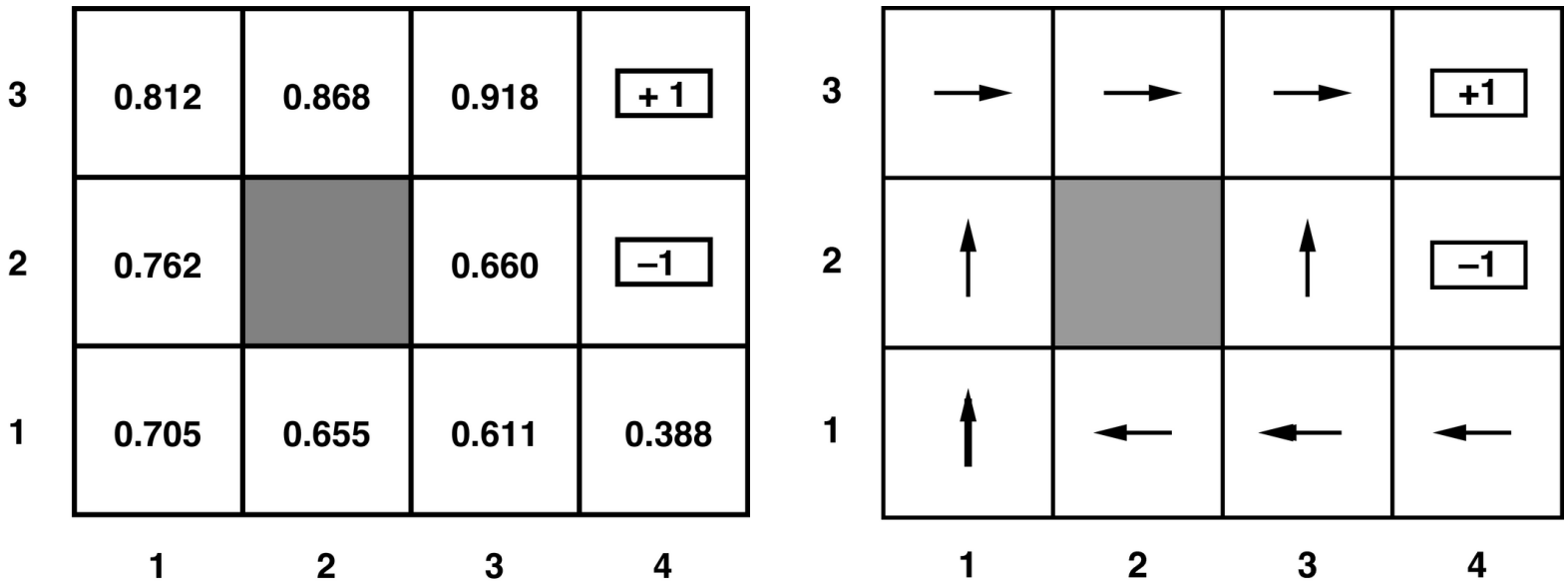
- If we have these values, the agent has a simple decision process
- It just picks the action a that maximises the expected utility of the next state:

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

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

UNIVERSITY OF LINCOLN

Optimal policies



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

Bellman equation

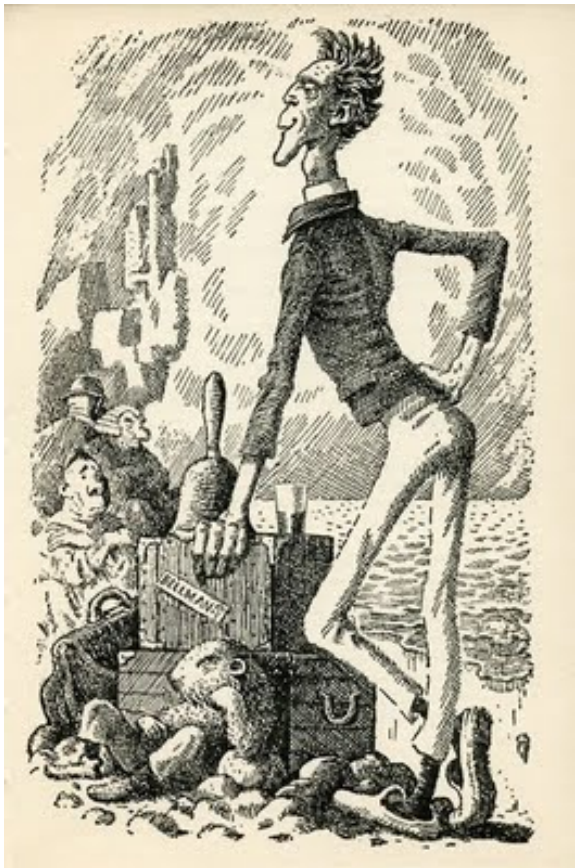
- How do we find the best policy (for a given set of rewards)?
- Turns out that there is a neat way to do this, by first computing the utility of each state.
- We compute this using the Bellman equation

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

- γ is the discount factor.

UNIVERSITY OF LINCOLN

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.

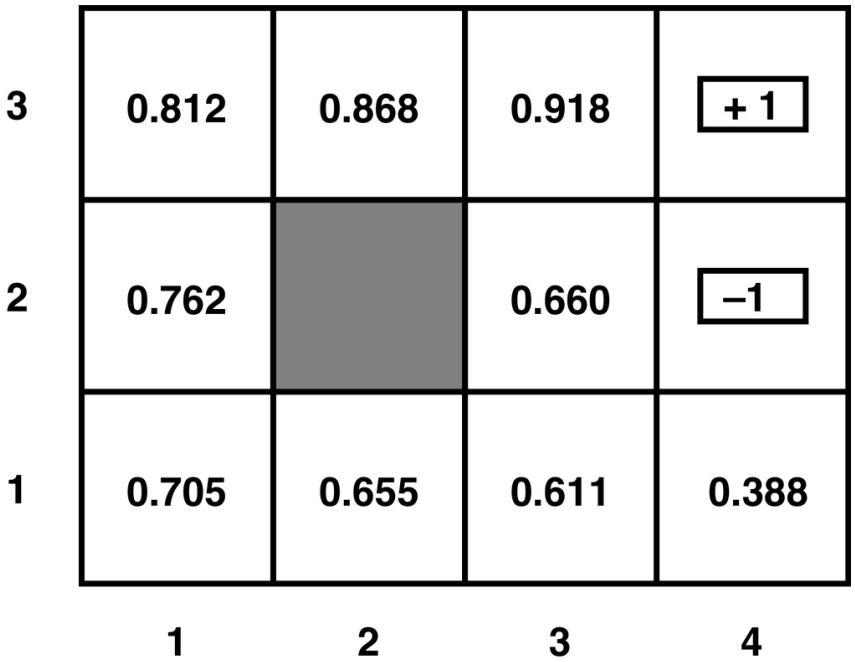
"Just the place for a Snark! I have said it twice:
That alone should encourage the crew.
Just the place for a Snark! I have said it thrice:
What I tell you three times is true."

Lewis Carroll

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

UNIVERSITY OF LINCOLN

Bellman equation

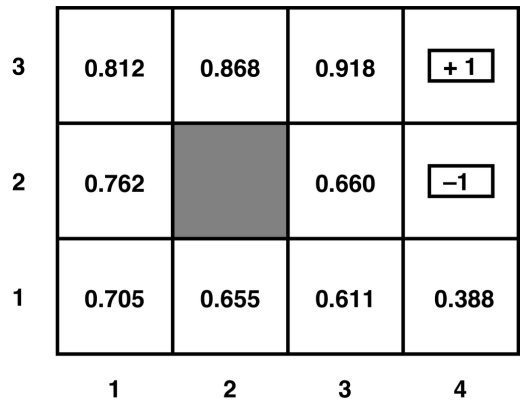


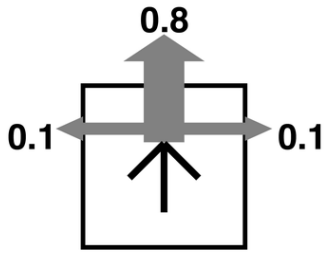
- Apply:

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

and we get:

Bellman equation





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

etc.

Bellman equation

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

- The utility of each state s is a function of the utilities of all states that can be reached by an action $a \in A(s)$.

UNIVERSITY OF LINCOLN

Value iteration

(Pendleton Ward/Cartoon Network)

- In an MDP wth n states, we will have n Bellman equations.
- Hard to solve these simultaneously because of the max operation
  - Makes them non-linear.

UNIVERSITY OF LINCOLN

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.

UNIVERSITY OF LINCOLN

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

UNIVERSITY OF LINCOLN

Value iteration

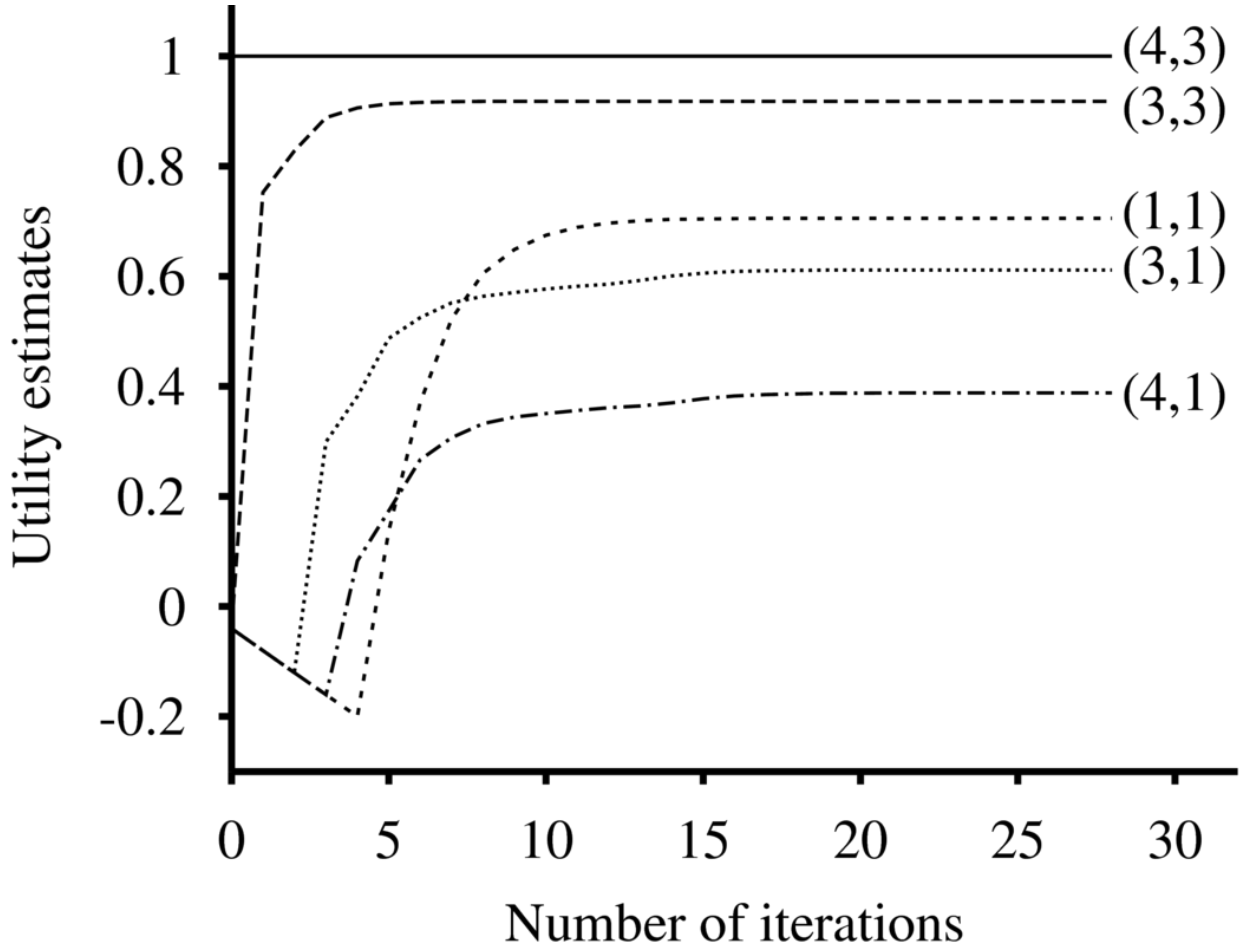
- States, S, reward, $R(s)$, set of actions, $A(s)$ and transition model, $P(s'|s,a)$, are exactly as defined on page 30.
- γ is the discount factor, as described on page 41.

Value iteration

initial	0.000	0.000	0.000	1.000
state	0.000		0.000	-1.000
	0.000	0.000	0.000	0.000
Iteration 1	-0.040	-0.040	0.760	1.000
	-0.040		-0.040	-1.000
	-0.040	-0.040	-0.040	-0.040
Iteration 2	-0.080	0.560	0.832	1.000
	-0.080		0.464	-1.000
	-0.080	-0.080	-0.080	-0.080
Iteration 3	0.392	0.738	0.890	1.000
	-0.120		0.572	-1.000
	-0.120	-0.120	0.315	-0.120
Iteration 4	0.577	0.819	0.906	1.000
	0.250		0.629	-1.000
	-0.160	0.188	0.394	0.100

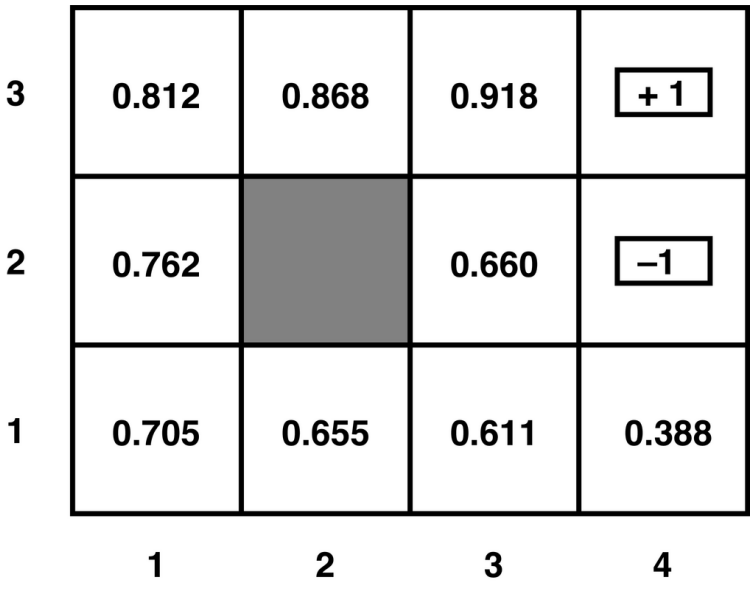
UNIVERSITY OF LINCOLN

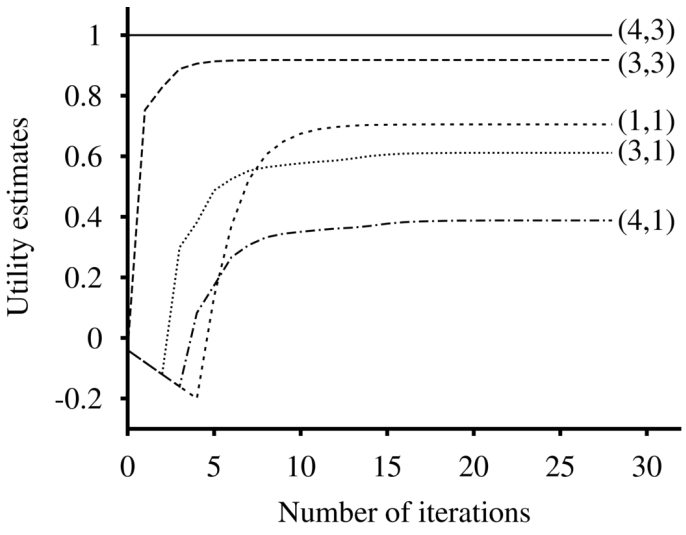
Value iteration



- 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 obtain the utility):

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

  where s is the action used.
- Bellman becomes:

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

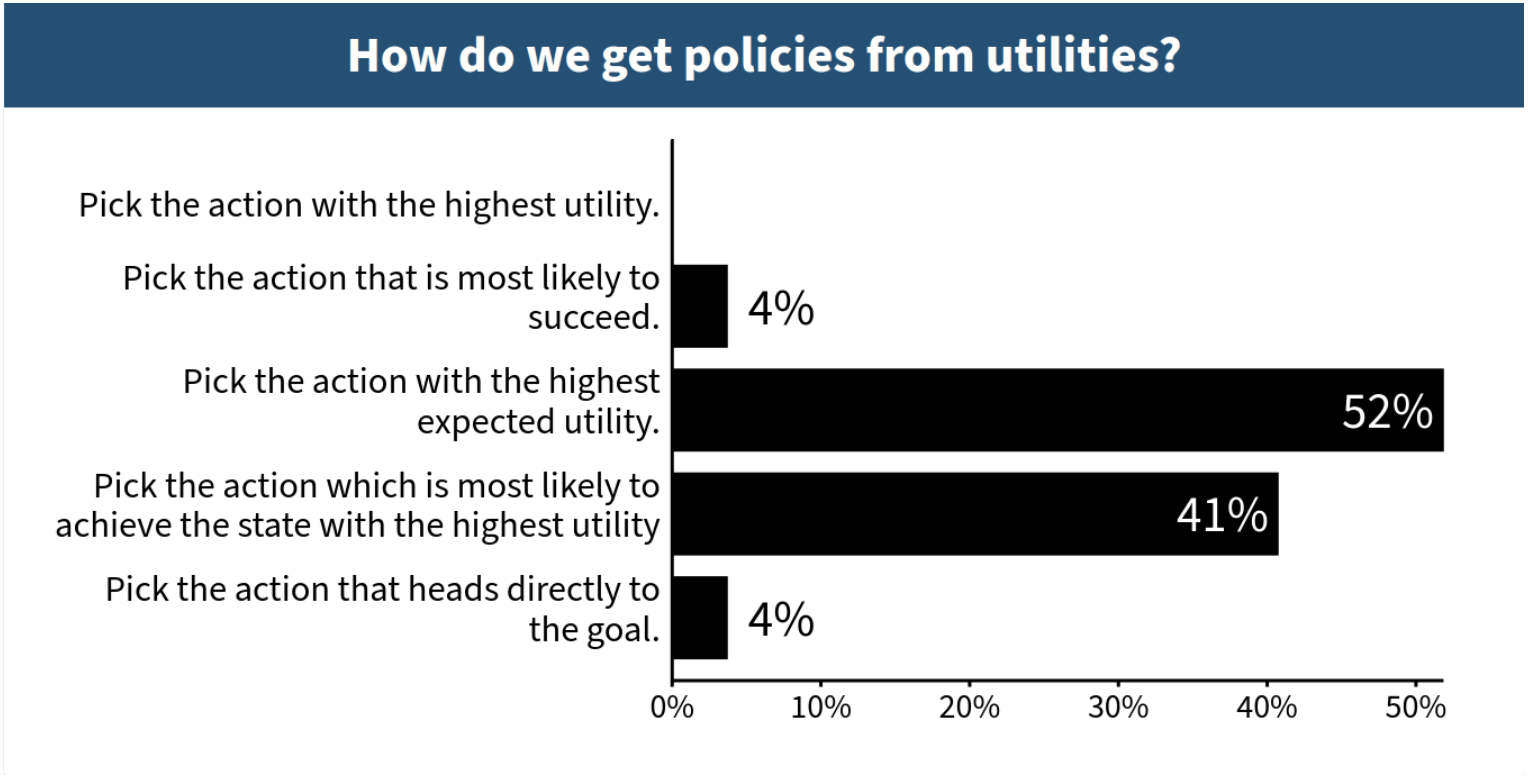
- Note that the action can be dependent on the state.

UNIVERSITY OF LINCOLN

Utility to policy?

- How do we get policies from utilities?

Utility to policy?



- The right answer is by picking the action with the largest expected utility.

Policy iteration

- The utilities are only helpful in getting us to the policy
- Can’t we go direct?

UNIVERSITY OF LINCOLN

Policy iteration

- The utilities are only helpful in getting us to the policy
- Can't we go direct?
- We can.

UNIVERSITY OF LINCOLN

Policy iteration

- Simple to write down an algorithm to do this:
  1. Set $i = 0$.
  2. Pick a policy π_i, and compute its expected utility $EU(\pi_i)$.
  3. Create a new policy π_{i+1} and compute $EU(\pi_{i+1})$.
  4. If $EU(\pi_{i+1}) > EU(\pi_i)$:
     - $i = i + 1$
     - Goto 3

     Else Goto 2
- Rather than compute optimal utility values, policy iteration looks through the space of possible policies.
- This is a simple, and somewhat silly, approach, there are better ones.

UNIVERSITY OF LINCOLN

Policy iteration

- 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.

UNIVERSITY OF LINCOLN

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.

UNIVERSITY OF LINCOLN

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:

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

- This looks rather similar to what we saw with value iteration.

UNIVERSITY OF LINCOLN

Policy evaluation

- Again there are lots of simultaneous equations:

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

- One equation for each state s.
- This time the equations are linear (no max) and so standard linear algebra solutions will work.
- (Call your favourite equation solver.)

UNIVERSITY OF LINCOLN

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.

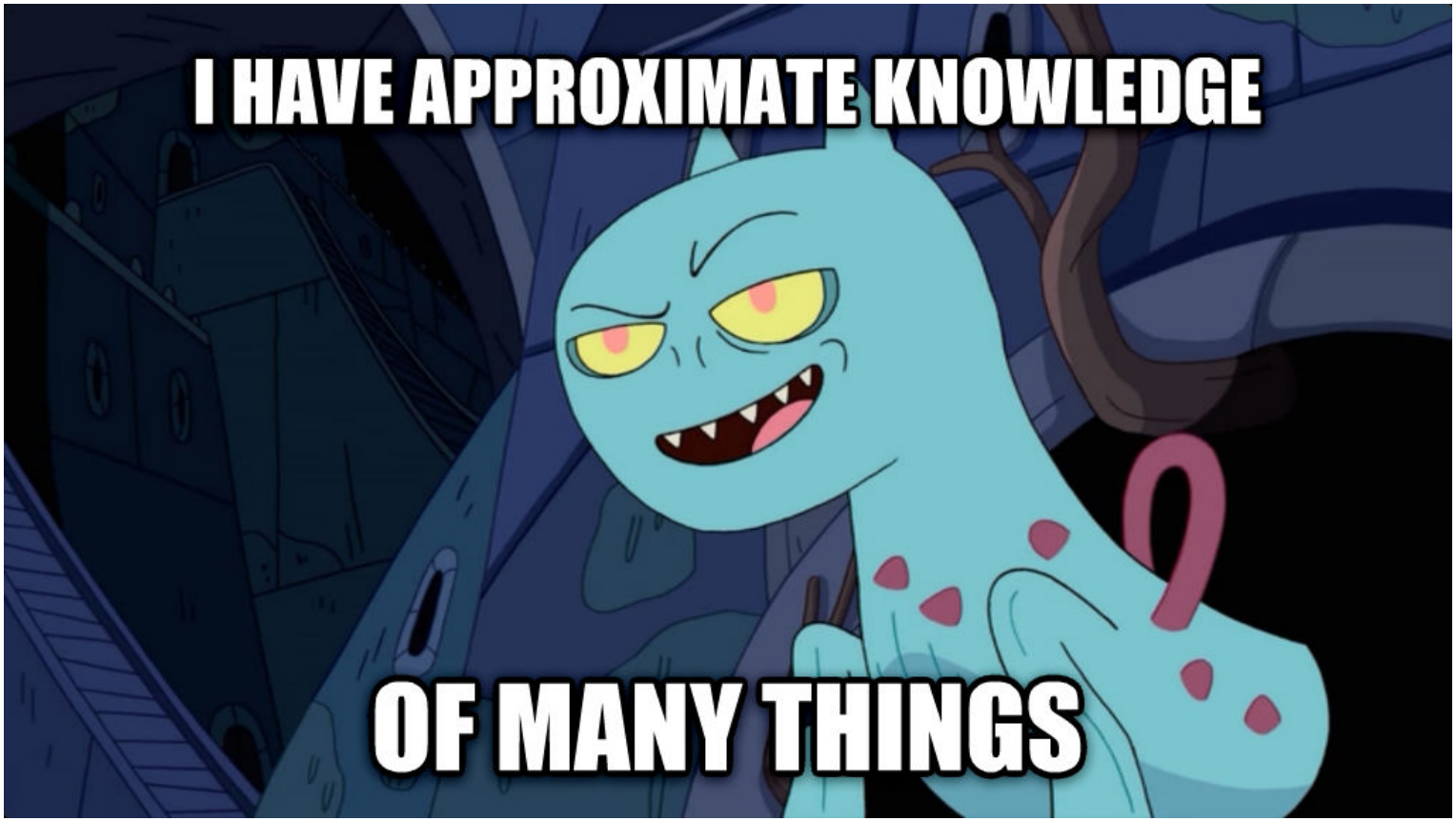
UNIVERSITY OF LINCOLN

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.

UNIVERSITY OF LINCOLN

Approximate?



(Pendleton Ward/Cartoon Network)

UNIVERSITY OF LINCOLN

Approximate policy evaluation

- For policy evaluation we use 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.
- (It won't necessarily converge.)

UNIVERSITY OF LINCOLN

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

Modified policy iteration

- 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 is best 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

UNIVERSITY OF LINCOLN

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.

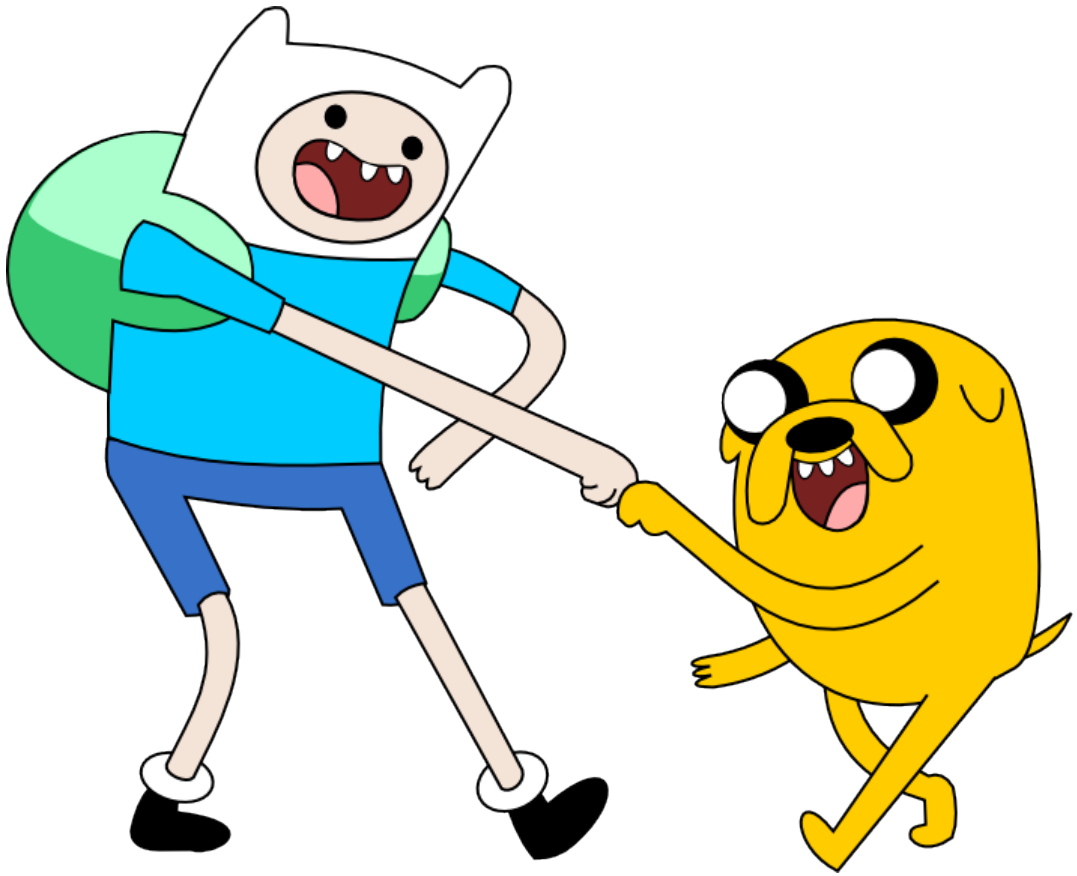
UNIVERSITY OF LINCOLN

Limitations of MDPs?

(Pendleton Ward/Cartoon Network)

UNIVERSITY OF LINCOLN

Mathematical!



(Pendleton Ward/Cartoon Network)

UNIVERSITY OF LINCOLN

Summary

- Today we looked at Markov Decision Processes
- A way to describe (and then study) sequential decision problems.
- We also covered *value iteration*, one of the methods for solving an MDP.
- Solving an MDP gives us a *policy* which describes the “right thing” to do in each state.
- Policy iteration gives us a couple more ways to solve an MDP.

UNIVERSITY OF LINCOLN

Version History

- Version 1.0, 4th December 2021
- Version 1.1, 6th December 2021
  - Added results of the polls.