Markov Decision Processes

Paolo Turrini

Department of Computing, Imperial College London

Introduction to Artificial Intelligence

The lectures

- The agent and the world (**Knowledge Representation**)
  - Actions and knowledge
  - Inference
- Good decisions (**Risk and Decisions**)
  - Chance
  - Gains
- Good decisions in time (**Markov Decision Processes**)
  - Chance and gains in time
  - Patience
  - Finding the best strategy
- Learning from experience (**Reinforcement Learning**)
  - Finding a reasonable strategy

Outline

- Time
- Risk
- Patience

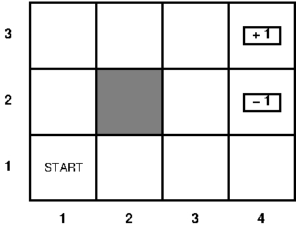
Markov Decision Processes

The right choices still need the right luck

The book

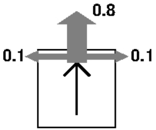
Stuart Russell and Peter Norvig
Artificial Intelligence: a modern approach
Chapter 17

The world



- Start at the starting square
- The game ends when we reach either goal state $+1$ or -1
- Collision results in no movement

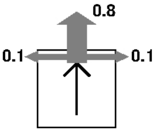
The agent



The agent chooses between $\{Up, Down, Left, Right\}$ and goes:

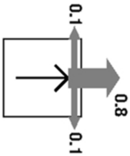
- to the intended direction with probability 0.8
- to the left of the intended direction with probability 0.1
- to the right of the intended direction with probability 0.1

The agent



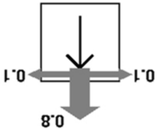
Each time it's like throwing an unfair dice

The agent



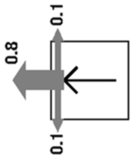
Each time it's like throwing an unfair dice

The agent



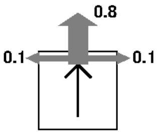
Each time it's like throwing an unfair dice

The agent



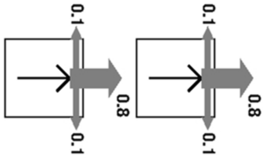
Each time it's like throwing an unfair dice

The agent



Each time it's like throwing an unfair dice

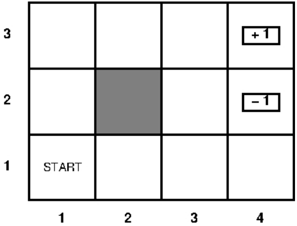
Let's start walking

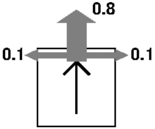


Walking is a repetition of throws:

- The probability that I walk right the first time: 0.8
- The probability that I walk right the second time: 0.8
- The probability that I walk right both times... is a product! 0.8^2

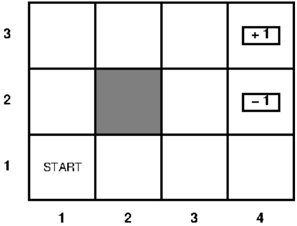
The agent and the world

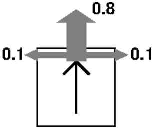




The environment is **fully observable**.

The agent and the world

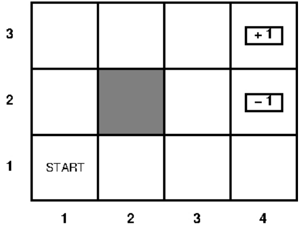


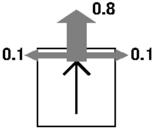


The environment is **fully observable**.

- the agent always knows what the world looks like: e.g., there is a wall, where the wall is, how to get to the wall ...
- the agent always knows their position during the game, even though some trajectories might not be reached with certainty.

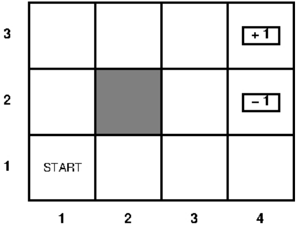
The agent and the world

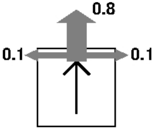




The environment is **Markovian**.

The agent and the world

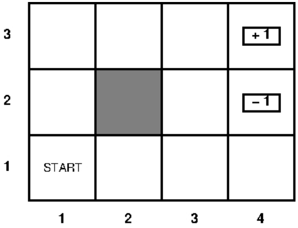


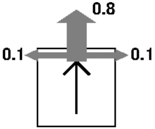


The environment is **Markovian**.

- the probability of reaching a state only depends on the state the agent is in and the action they perform.

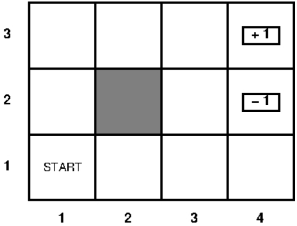
The agent and the world

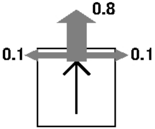




Formally....

The agent and the world

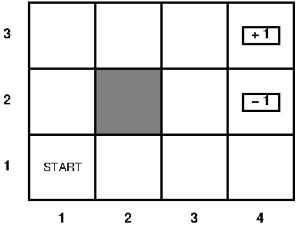


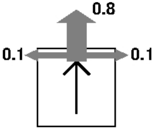


Formally....

- Denote $[x, y]$ the fact that the agent is at square $[x, y]$ now

The agent and the world

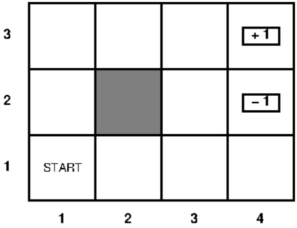


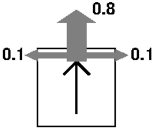


Formally....

- Denote $[x, y]$ the fact that the agent is at square $[x, y]$ now
- Denote $[x, y]_t$ the fact that the agent is at square $[x, y]$ at time t

The agent and the world



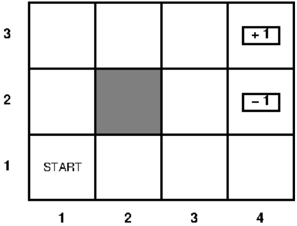


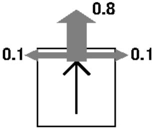
The fact that the environment is Markovian means this:

$$P([x,y]_t \mid Up, [x-1,y]_{t-1}) = P([x,y]_t \mid Up, [x-1,y]_{t-1}, [x',y']_{t-t'})$$

for any non terminal $[x',y']$ and $1 < t' < t$

The agent and the world





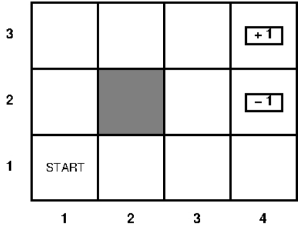
The fact that the environment is Markovian means this:

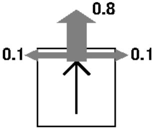
$$P([x,y]_t \mid Up, [x-1,y]_{t-1}) = P([x,y]_t \mid Up, [x-1,y]_{t-1}, [x',y']_{t-t'})$$

for any non terminal $[x',y']$ and $1 < t' < t$

The past does not matter!

The agent and the world





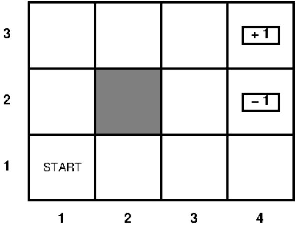
The fact that the environment is Markovian means this:

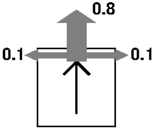
$$P([x,y]_t \mid Down, [x+1,y]_{t-1}) =$$
$$P([x,y]_t \mid Down, [x+1,y]_{t-1}, [x',y']_{t-t'})$$

for any non terminal $[x', y']$ and $1 < t' < t$

The past does not matter!

The agent and the world





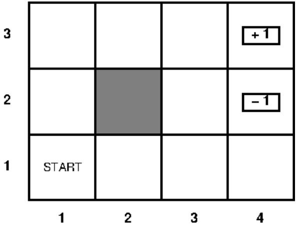
The fact that the environment is Markovian means this:

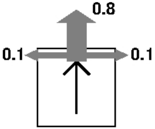
$$P([x, y]_t \mid Right, [x, y-1]_{t-1}) =$$
$$P([x, y]_t \mid Right, [x, y-1]_{t-1}, [x', y']_{t-t'})$$

for any non terminal $[x', y']$ and $1 < t' < t$

The past does not matter!

The agent and the world





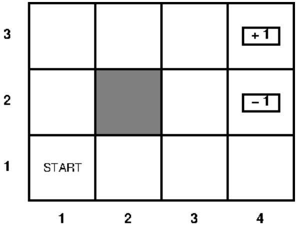
The fact that the environment is Markovian means this:

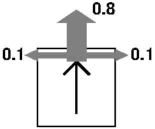
$$P([x,y]_t \mid \mathit{Left}, [x,y+1]_{t-1}) =$$
$$P([x,y]_t \mid \mathit{Left}, [x,y+1]_{t-1}, [x',y']_{t-t'})$$

for any non terminal $[x',y']$ and $1 < t' < t$

The past does not matter!

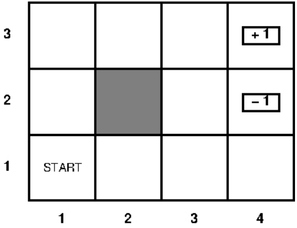
Plans

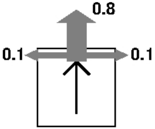




$\{Up, Down, Left, Right\}$ denote the intended directions.

Plans

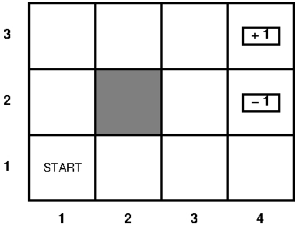


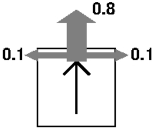


$\{Up, Down, Left, Right\}$ denote the intended directions.

A plan is a finite sequence of **intended** moves, **from the start**.

Plans



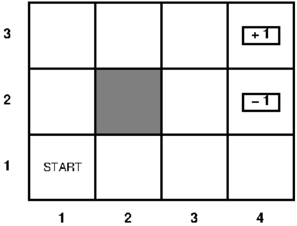


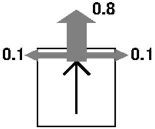
$\{Up, Down, Left, Right\}$ denote the intended directions.

A plan is a finite sequence of **intended** moves, **from the start**.

So $[Up, Down, Up, Right]$ is going to be the plan that, from the starting square, selects the intended moves in the specified order.

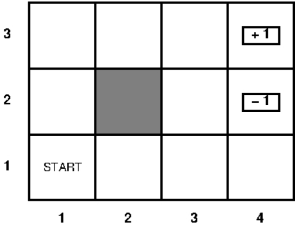
Makings plans

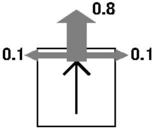




Goal: get to $+1$

Makings plans

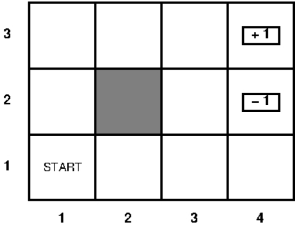


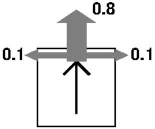


Goal: get to $+1$

Consider the plan $[Up, Up, Right, Right, Right]$.

Makings plans



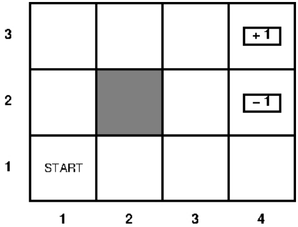


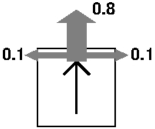
Goal: get to $+1$

Consider the plan $[Up, Up, Right, Right, Right]$.

- With deterministic agents, it gets us to $+1$ with probability 1.
- But what happens to our stochastic agent instead?

Makings plans





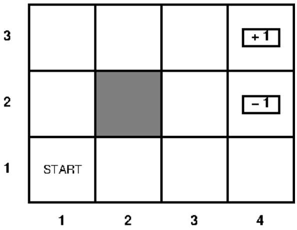
Goal: get to $+1$

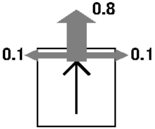
Consider the plan $[Up, Up, Right, Right, Right]$.

- With deterministic agents, it gets us to $+1$ with probability 1.
- But what happens to our stochastic agent instead?

What's the probability that $[Up, Up, Right, Right, Right]$ gets us to $+1$?

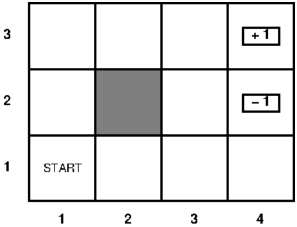
Makings plans

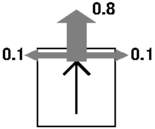




- It's not 0.8^5!
- 0.8^5 is the probability that we get to $+1$ when the plan works!

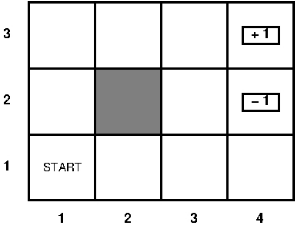
Makings plans

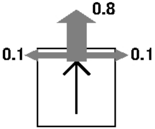




- There is a small chance of [*Up*, *Up*, *Right*, *Right*, *Right*] accidentally reaching the goal by going the other way round!

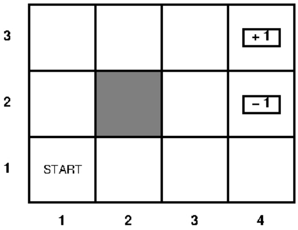
Makings plans

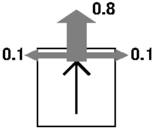




- There is a small chance of $[Up, Up, Right, Right, Right]$ accidentally reaching the goal by going the other way round!
- The probability of this to happen is $0.1^4 \times 0.8 = 0.00008$

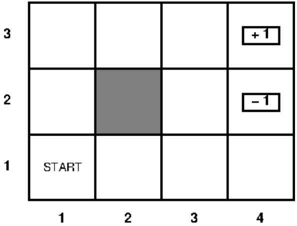
Makings plans

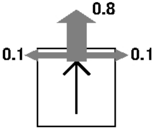




- There is a small chance of $[Up, Up, Right, Right, Right]$ accidentally reaching the goal by going the other way round!
- The probability of this to happen is $0.1^4 \times 0.8 = 0.00008$
- So the probability that $[Up, Up, Right, Right, Right]$ gets us to $+1$ is $0.32768 + 0.00008 = 0.32776$

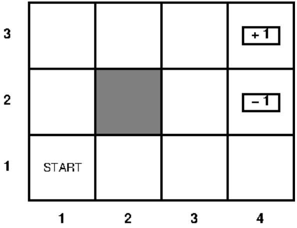
Makings plans

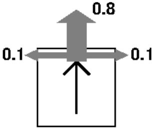




- In this case, the probability of accidental successes doesn't play a significant role. However it might very well do, under different decision models, rewards, environments etc.
- 0.32776 is still less than $\frac{1}{3}$... not great.

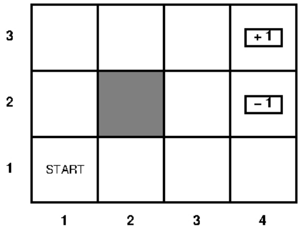
Expected utility

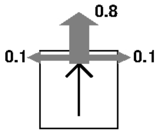




What's the expected utility of $[Up, Up, Right, Right, Right]$?

Expected utility

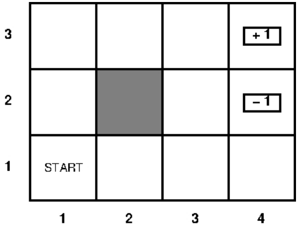


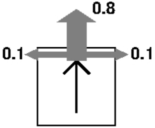


What's the expected utility of [*Up*, *Up*, *Right*, *Right*, *Right*]?

IT DEPENDS

Expected utility

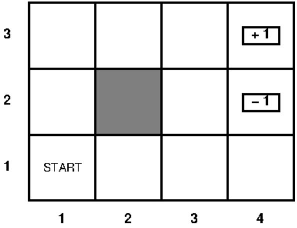


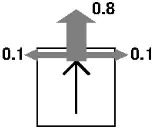


What's the expected utility of $[Up, Up, Right, Right, Right]$?

IT DEPENDS on how we are going to put rewards together!

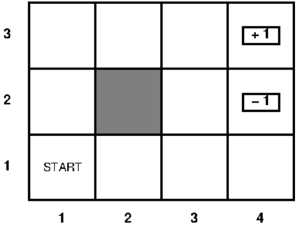
Expected utility

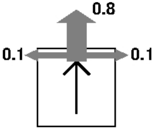




We are going to assume that:

Expected utility

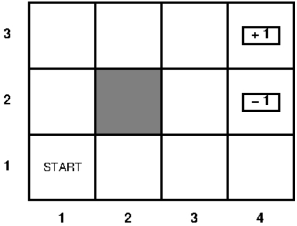


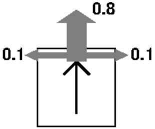


We are going to assume that:

- each sequence hitting a terminal state gets the corresponding value, and 0 otherwise.

Expected utility

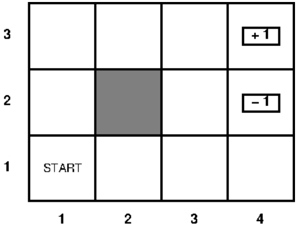


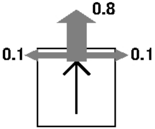


We are going to assume that:

- each sequence hitting a terminal state gets the corresponding value, and 0 otherwise.
- The expected utility of the plan is the sum of the utility of all sequences, weighted with their probability to occur.

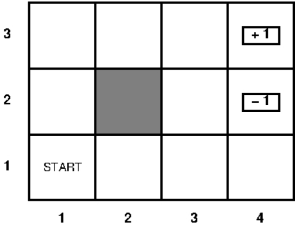
Expected utility

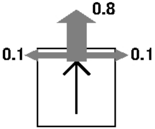




- We only care about the sequences that get us to a terminal state, as the others, no matter how likely, are going to yield 0.

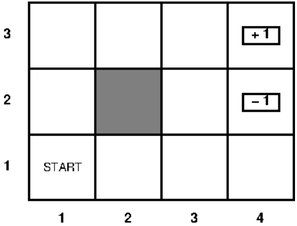
Expected utility

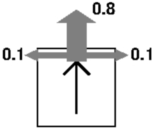




- We only care about the sequences that get us to a terminal state, as the others, no matter how likely, are going to yield 0.
- We already know that the probability of the plan $[Up, Up, Right, Right, Right]$ to get to $+1$ is 0.32776.

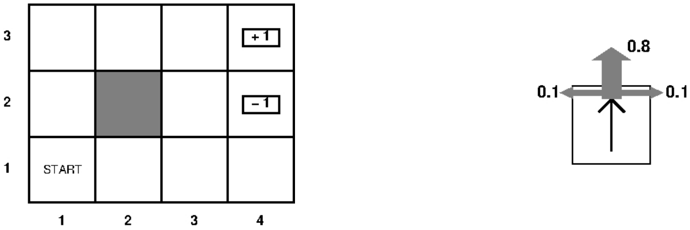
Expected utility





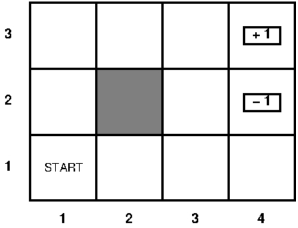
- Now we need to calculate the probability of it to get us to -1.

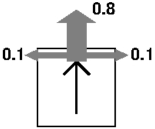
Expected utility



- Now we need to calculate the probability of it to get us to -1.
- It's... quite tricky to calculate, and is 0.014.

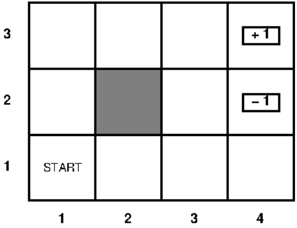
Expected utility

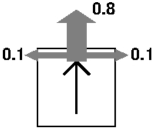




- Now we need to calculate the probability of it to get us to -1.
- It's... quite tricky to calculate, and is 0.014.
- (Thanks to Nicholas Huang and Panayiotis Panayiotou for finding it!)

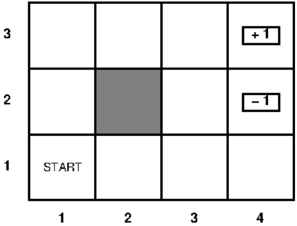
Expected utility

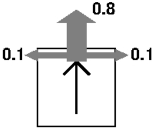




So the expected utility of $[Up, Up, Right, Right, Right]$ is...

Expected utility

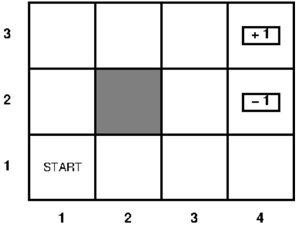


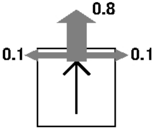


So the expected utility of $[Up, Up, Right, Right, Right]$ is...

$$0.32776 - 0.014$$

Expected utility

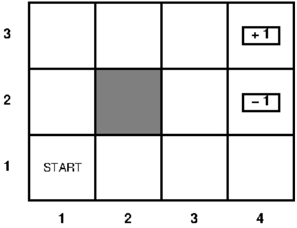


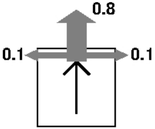


So the expected utility of $[Up, Up, Right, Right, Right]$ is...

$$0.32776 - 0.014 = 0.31376$$

Expected utility





So the expected utility of $[Up, Up, Right, Right, Right]$ is...

$$0.32776 - 0.014 = 0.31376$$:)

Rewards

- In calculating the expected utility of $[Up, Up, Right, Right, Right]$ we only had to consider the reward of the terminal states.

Rewards

- In calculating the expected utility of $[Up, Up, Right, Right, Right]$ we only had to consider the reward of the terminal states.
- Now we complicate things a bit.

Rewards

- In calculating the expected utility of $[Up, Up, Right, Right, Right]$ we only had to consider the reward of the terminal states.
- Now we complicate things a bit.

A reward function is a (utility) function of the form

$$r : S \to \mathbb{R}$$

Rewards

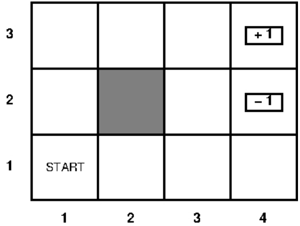
- In calculating the expected utility of $[Up, Up, Right, Right, Right]$ we only had to consider the reward of the terminal states.
- Now we complicate things a bit.

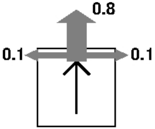
A reward function is a (utility) function of the form

$$r : S \to \mathbb{R}$$

All states, not just the terminal ones, get a reward!

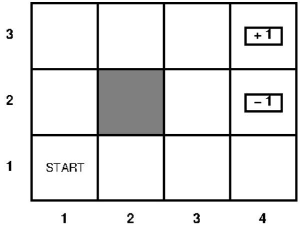
Rewards

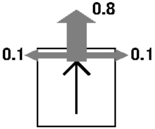




For instance, each non-terminal state:

Rewards

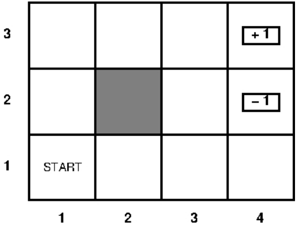


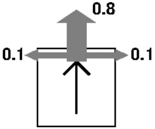


For instance, each non-terminal state:

- has 0 reward, i.e., only the terminal states matter;

Rewards

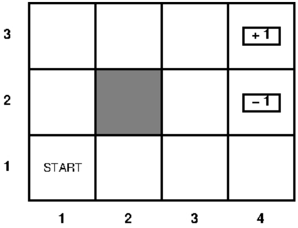


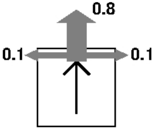


For instance, each non-terminal state:

- has 0 reward, i.e., only the terminal states matter;
- has negative reward, e.g., each move consumes -0.04 of battery;

Rewards

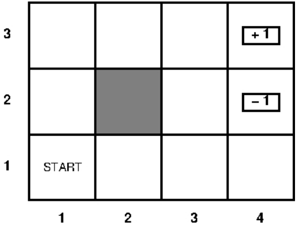


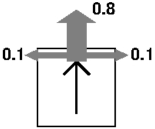


For instance, each non-terminal state:

- has 0 reward, i.e., only the terminal states matter;
- has negative reward, e.g., each move consumes -0.04 of battery;
- has positive reward, e.g., I like wasting battery

Rewards



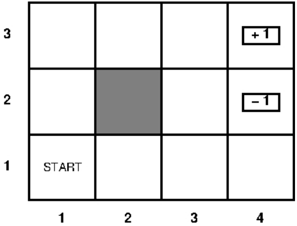


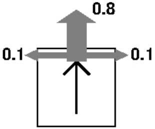
For instance, each non-terminal state:

- has 0 reward, i.e., only the terminal states matter;
- has negative reward, e.g., each move consumes -0.04 of battery;
- has positive reward, e.g., I like wasting battery

Rewards are usually small, negative and uniform at non-terminal states. But the reward function allows for more generality.

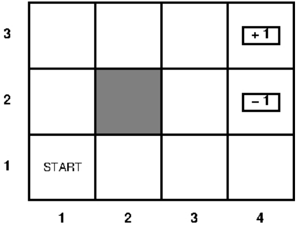
Rewards

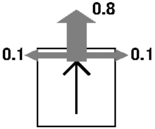




Consider now the following. The reward is:

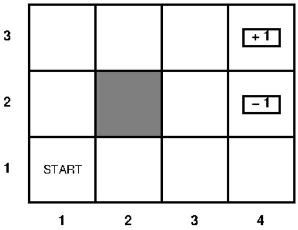
Rewards

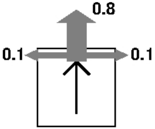




Consider now the following. The reward is:
$+1$ at state $+1$, -1 at state -1, -0.04 in all other states.

Rewards

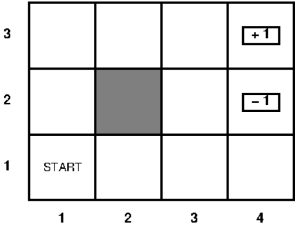


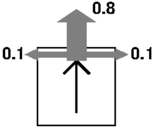


Consider now the following. The reward is:
$+1$ at state $+1$, -1 at state -1, -0.04 in all other states.

What's the expected utility of [*Up*, *Up*, *Right*, *Right*, *Right*]?

Rewards



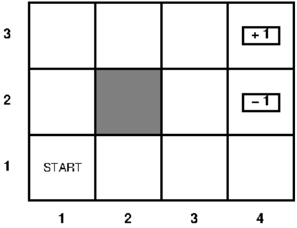


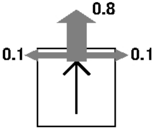
Consider now the following. The reward is:
$+1$ at state $+1$, -1 at state -1, -0.04 in all other states.

What's the expected utility of $[Up, Up, Right, Right, Right]$?

IT DEPENDS

Rewards





Consider now the following. The reward is:
+1 at state +1, -1 at state -1, -0.04 in all other states.

What's the expected utility of $[Up, Up, Right, Right, Right]$?

IT DEPENDS on how we are going to put rewards together!

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

So, if we visit sequence $s_1, s_2, \ldots, s_n$,
returning rewards $r_1, r_2, \ldots, r_n$,

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

So, if we visit sequence $s_1, s_2, \ldots, s_n$,
returning rewards $r_1, r_2, \ldots, r_n$, we want to know:

$$u[r_1, r_2, \ldots r_n]$$

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

So, if we visit sequence $s_1, s_2, \ldots, s_n$,
returning rewards $r_1, r_2, \ldots, r_n$, we want to know:

$$u[r_1, r_2, \ldots r_n]$$

This is just multi-criteria decision-making!

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

So, if we visit sequence $s_1, s_2, \ldots, s_n$,
returning rewards $r_1, r_2, \ldots, r_n$, we want to know:

$$u[r_1, r_2, \ldots r_n]$$

This is just multi-criteria decision-making!
So the question becomes:

Utility of state sequences

Now we need to compare each sequence of state we can encounter, not just the final state!

So, if we visit sequence $s_1, s_2, \ldots, s_n$,
returning rewards $r_1, r_2, \ldots, r_n$, we want to know:

$$u[r_1, r_2, \ldots r_n]$$

This is just multi-criteria decision-making!
So the question becomes:

Which states are important and why?

Comparing states

Many ways of comparing states:

Comparing states

Many ways of comparing states:

- summing all the rewards

Comparing states

Many ways of comparing states:

- summing all the rewards
- giving priority to the immediate rewards

Comparing states

Many ways of comparing states:

- summing all the rewards
- giving priority to the immediate rewards
- . . .

Utility of state sequences

We are going to assume only one axiom,

Utility of state sequences

We are going to assume only one axiom, stationary preferences on reward sequences:

Utility of state sequences

We are going to assume only one axiom, stationary preferences on reward sequences:

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r_0', r_1', r_2', \ldots] \Leftrightarrow [r_0, r_1, r_2, \ldots] \succ [r_0', r_1', r_2', \ldots]$$

In words...

Utility of state sequences

We are going to assume only one axiom,
stationary preferences on reward sequences:

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots] \;\Leftrightarrow\; [r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

In words...

- If we have a preference over two sequences that start in the same way, then we should keep the same preference after the first move.
- The other way round: if we have a preference over two sequences, then we should keep the same preference over the same sequences with the addition of one same initial state.

Utility of state sequences

Theorem

If we accept the previous axiom, there is only one way to combine rewards over time.

Utility of state sequences

Theorem

If we accept the previous axiom, there is only one way to combine rewards over time.

Discounted utility function:

Utility of state sequences

Theorem

If we accept the previous axiom, there is only one way to combine rewards over time.

Discounted utility function:

$u([s_0, s_1, s_2, \ldots]) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \cdots$

where $\gamma \in [0, 1]$ *is the* **discount factor**

Utility of state sequences

Theorem

If we accept the previous axiom, there is only one way to combine rewards over time.

Discounted utility function:

$$u([s_0, s_1, s_2, \ldots]) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \cdots$$

where $\gamma \in [0, 1]$ is the **discount factor**

Notice: additive utility function $u([s_0, s_1, s_2, \ldots]) = r(s_0) + r(s_1) + r(s_2) + \cdots$ is just a discounted utility function where $\gamma = 1$.

Discount factor

γ is a measure of the agent patience. How much more they value a gain of five today than a gain of five tomorrow, the day after etc...

- Used everywhere in AI, game theory, cognitive psychology
- A lot of experimental research on it
- Variants: discounting the discounting! I care more about the difference between today and tomorrow than the difference between some distant moment and the day after that!

Discounting

- $\gamma = 1$ we don't care about today
- $\gamma = 0$ we don't care about tomorrow

Discounting

- $\gamma = 1$ we don't care about today
- $\gamma = 0$ we don't care about tomorrow

Tomorrow is a lottery!

Discounting

- $\gamma = 1$ we don't care about today
- $\gamma = 0$ we don't care about tomorrow

Tomorrow is a lottery!

e.g., you transfer money to UK, thinking you are smart, and then Brexit happens...

Markov Decision Process

A Markov Decision Process is a sequential decision problem for a:

Markov Decision Process

A Markov Decision Process is a sequential decision problem for a:

- fully observable environment

Markov Decision Process

A Markov Decision Process is a sequential decision problem for a:

- fully observable environment
- with stochastic actions

Markov Decision Process

A Markov Decision Process is a sequential decision problem for a:

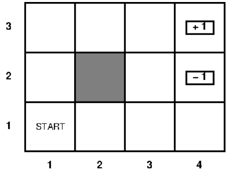
- fully observable environment
- with stochastic actions
- with a Markovian transition model

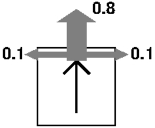
Markov Decision Process

A Markov Decision Process is a sequential decision problem for a:

- fully observable environment
- with stochastic actions
- with a Markovian transition model
- and with discounted (possibly additive) rewards

MDPs formally





Definition

Let s be a state and a and action

Model $P(s'|s,a)$ = probability that a in s leads to s'

Reward function $r(s)$ (or $r(s,a)$, $r(s,a,s')$) =

$$\begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Value of plans

The expected utility (or value) of a plan p, from state s is:

Value of plans

The expected utility (or value) of a plan p, from state s is:

$$v^p(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$$

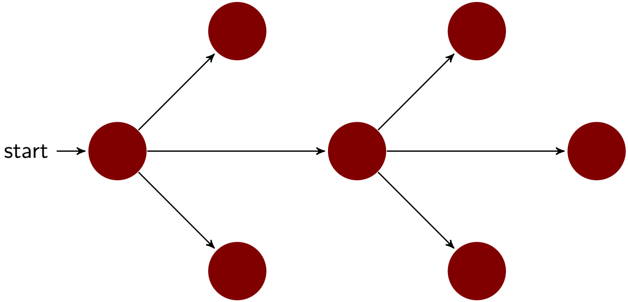
Where S_t is a random variable and the expected utility is calculated on the probability distribution over the discounted state sequences determined by s and p.

Value of plans: the recipe

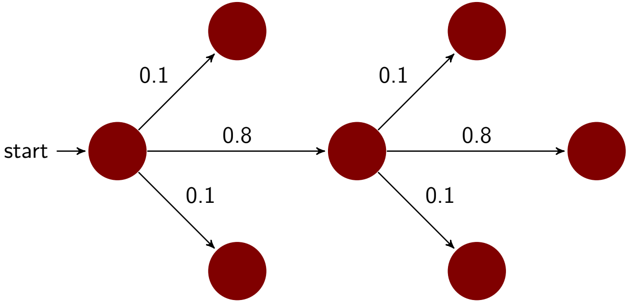
$$v^p(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$$

1. Calculate the utility of the sequences you can actually perform, with the appropriately discounted rewards, times the probability of reaching them.
2. Add these numbers
3. Forget about the rest

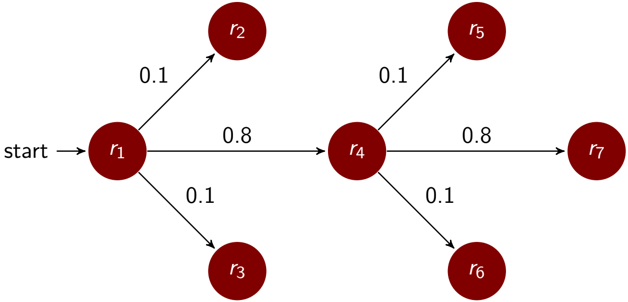
Value of plans: the recipe



Value of plans: the recipe

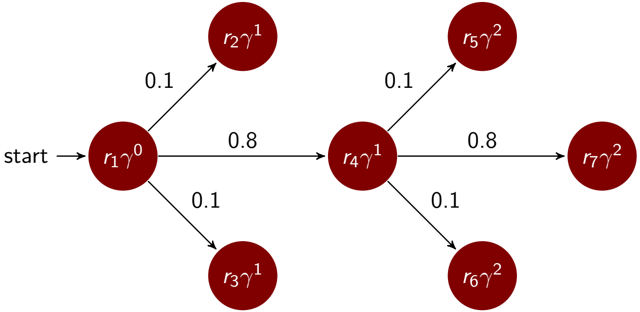


Value of plans: the recipe



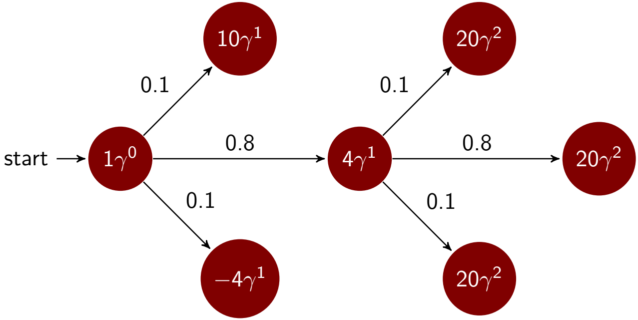
The real value of rewards depends on the agent's patience.
(as much as the real value of money depends on the attitude towards risk)

Value of plans: the recipe



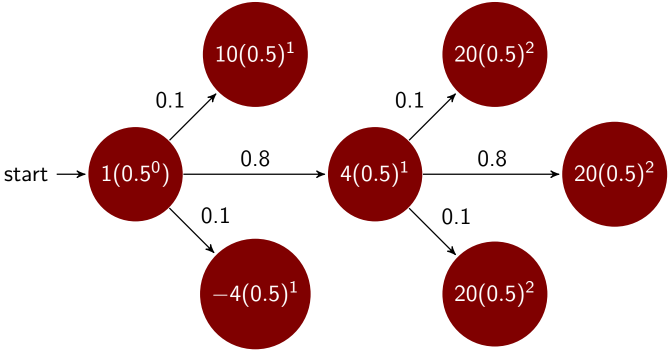
Multiplicative discounting: γ^n after n steps.

Value of plans: the recipe



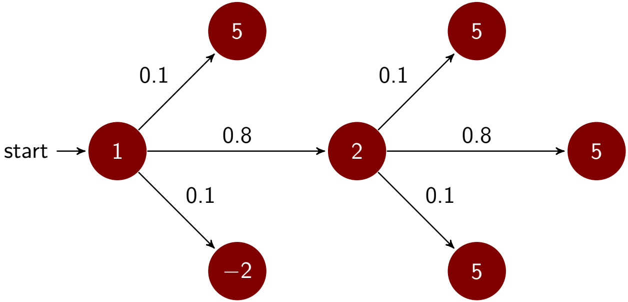
Multiplicative discounting: γ^n after n steps.

Value of plans: the recipe



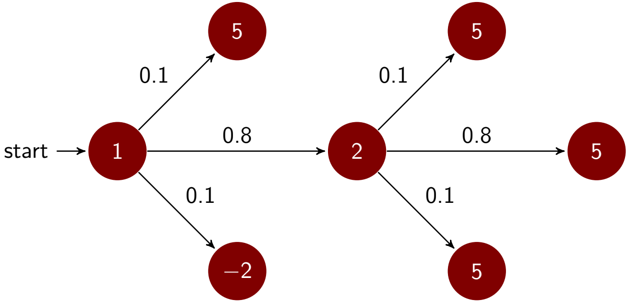
$\gamma = 0.5$

Value of plans: the recipe



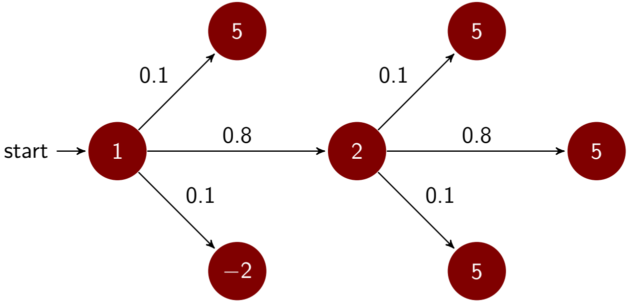
$\gamma = 0.5$

Value of plans: the recipe



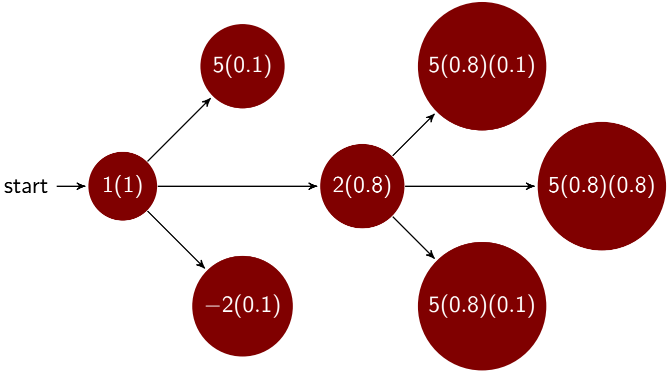
And now?

Value of plans: the recipe



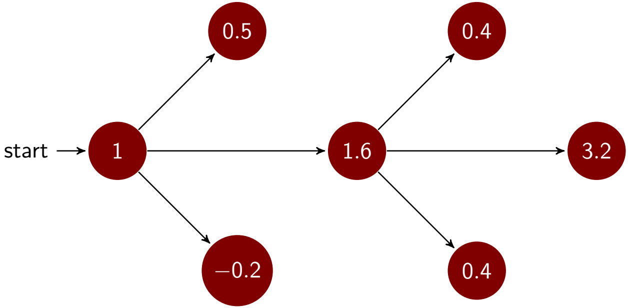
And now?
We include the probabilities...

Value of plans: the recipe



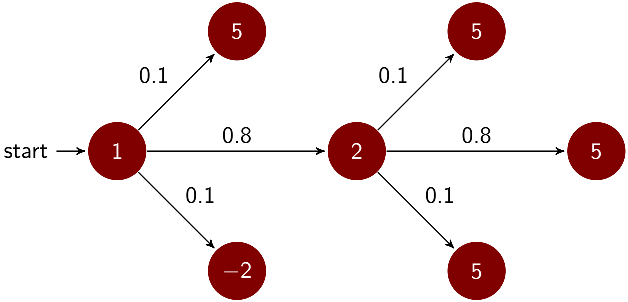
Probabilities of sequences:
to *discount* further the already discounted rewards!

Value of plans: the recipe



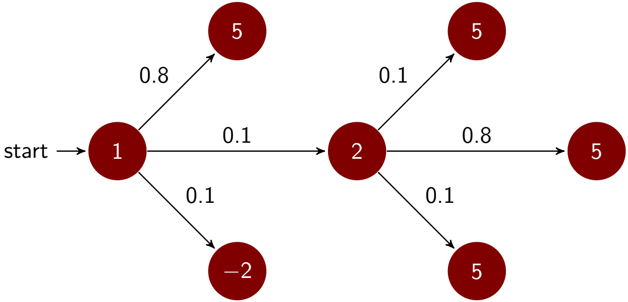
Expected utility of this intended course of actions (not considering the rest = assuming it's zero reward everywhere else) is:
6.9

Value of plans: the recipe



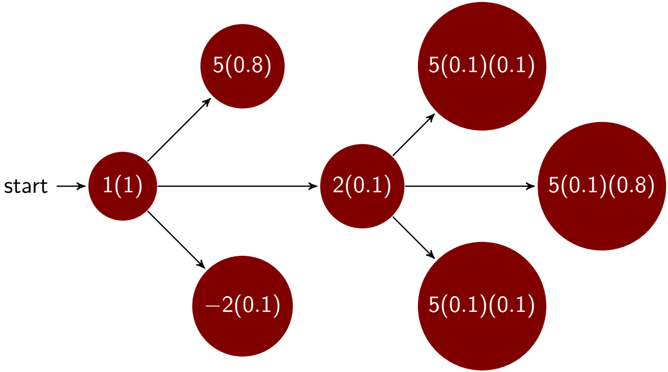
Let's see what happens if we go up instead...

Value of plans: the recipe



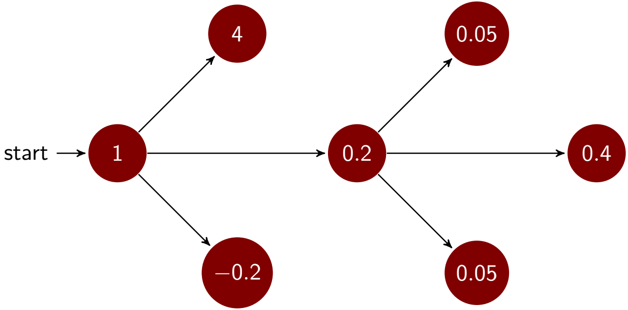
Let's see what happens if we go up instead...

Value of plans: the recipe



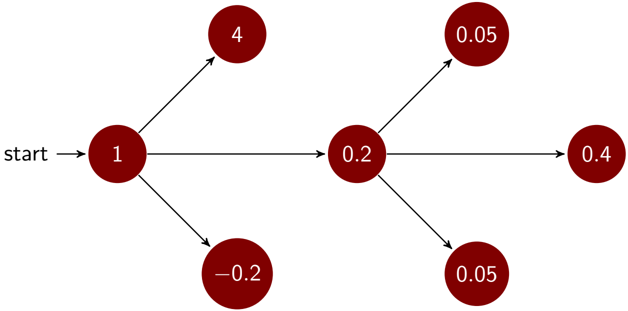
Including probabilities...

Value of plans: the recipe



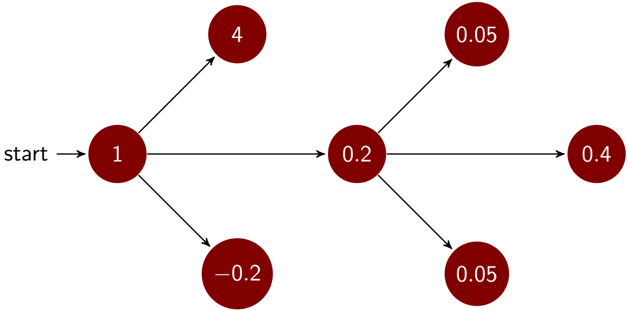
Including probabilities...

Value of plans: the recipe



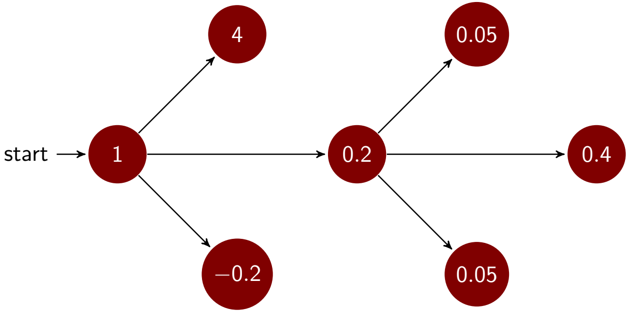
Summing up: 5.5

Value of plans: the recipe



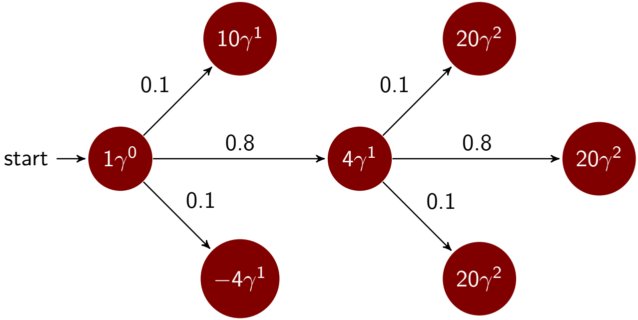
This means that switching to *Up* is dominated by going right.

Value of plans: the recipe



This means that switching to *Up* is dominated by going right.
Same reasoning for going down: lower expected utility!

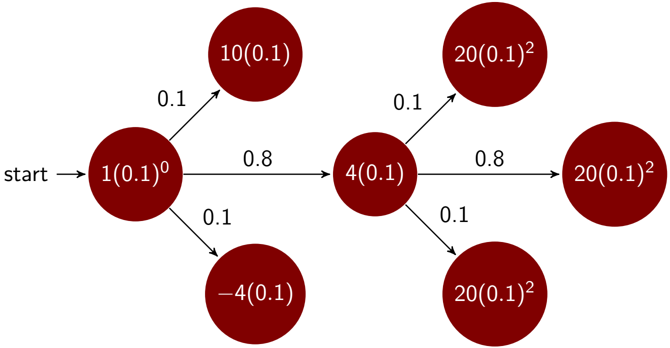
Value of plans: the recipe



Now I'm going to be very impatient.

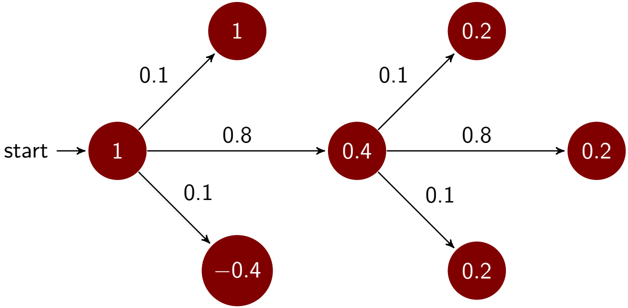
$\gamma = 0.1$

Value of plans: the recipe



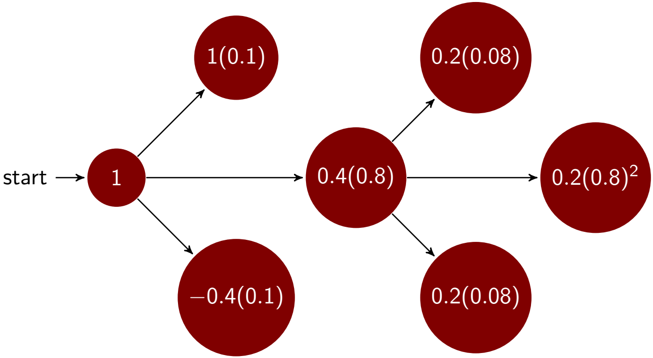
Now I'm going to be very impatient.
$\gamma = 0.1$

Value of plans: the recipe



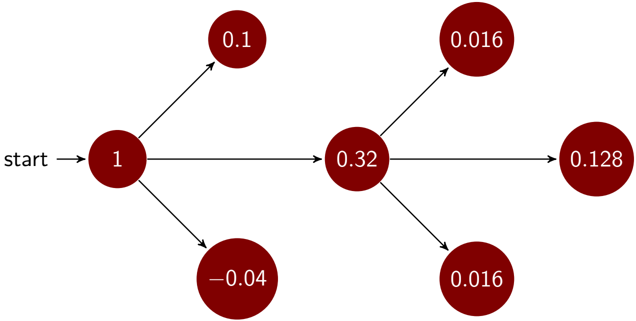
Can you already see what's going on?

Value of plans: the recipe



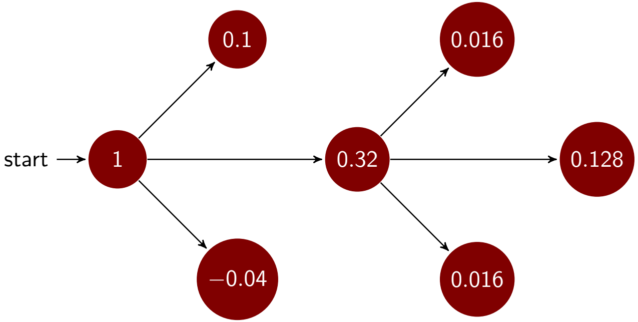
Let's include the probabilities

Value of plans: the recipe



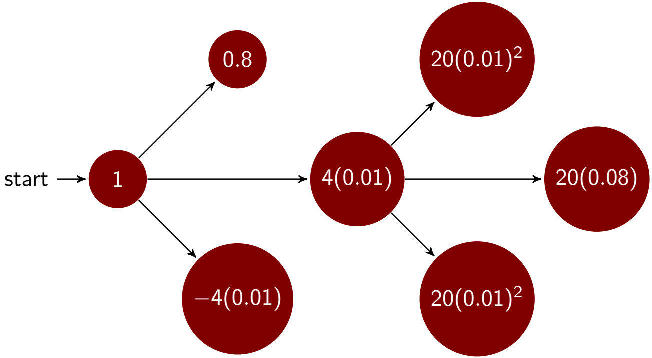
Notice the impact of discounting on negative rewards:
In the end, it's all gonna be zero!

Value of plans: the recipe



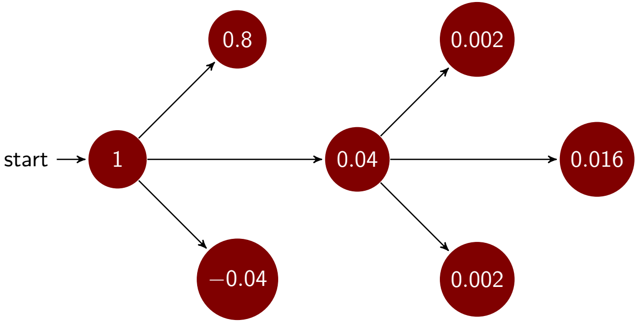
The expected utility at the starting state is: 1.54

Value of plans: the recipe



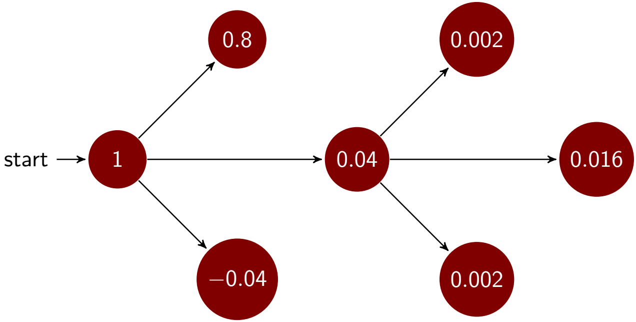
Let's go up

Value of plans: the recipe



The expected utility at the starting state is: 1.82

Value of plans: the recipe



Now Up is better!

What's going on here?

$$v^p(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$$

- First I told you to put together the sequences

What's going on here?

$$v^p(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$$

- First I told you to put together the sequences
- Then I showed a method that only looks at states

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$$

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$

$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$
$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$$

$$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$$
$$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$$

$$p(s_1)(\gamma)(r_1)(p(s_2) + p(s_2')) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2) =$$

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$$

$$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$$
$$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$$

$$p(s_1)(\gamma)(r_1)(p(s_2) + p(s_2')) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2) =$$

$$p(s_1)(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2)$$

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$

$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$
$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$

$p(s_1)(\gamma)(r_1)(p(s_2) + p(s_2')) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2) =$

$p(s_1)(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2)$

:)

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$

$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$
$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$

$p(s_1)(\gamma)(r_1)(p(s_2) + p(s_2')) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2) =$

$p(s_1)(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2)$

:)
Notice how this works no matter the number of outcoming edges!

It's the same thing!

Take a state s_1 with two outcoming edges: s_2 and s_2'.

$$p(s_1)p(s_2)((\gamma)(r_1) + (\gamma^2)(r_2)) + p(s_1)p(s_2')((\gamma)(r_1) + (\gamma^2)(r_2')) =$$

$$p(s_1)p(s_2)(\gamma)(r_1) + p(s_1)p(s_2)(\gamma^2)(r_2)$$
$$+p(s_1)p(s_2')(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') =$$

$$p(s_1)(\gamma)(r_1)(p(s_2) + p(s_2')) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2) =$$

$$p(s_1)(\gamma)(r_1) + p(s_1)p(s_2')(\gamma^2)(r_2') + p(s_1)p(s_2)(\gamma^2)(r_2)$$

:)
Notice how this works no matter the number of outcoming edges!
:)))

Today's class

- Markov Decision Processes
- Walking
- Planning
- Comparing plans

Coming next

- Plans when plans don't work out
- Plans when plans for plans that don't work out don't work out
- Etc.

Coming next

- Plans when plans don't work out
- Plans when plans for plans that don't work out don't work out
- Etc.