Making Complex Decisions

Paolo Turrini

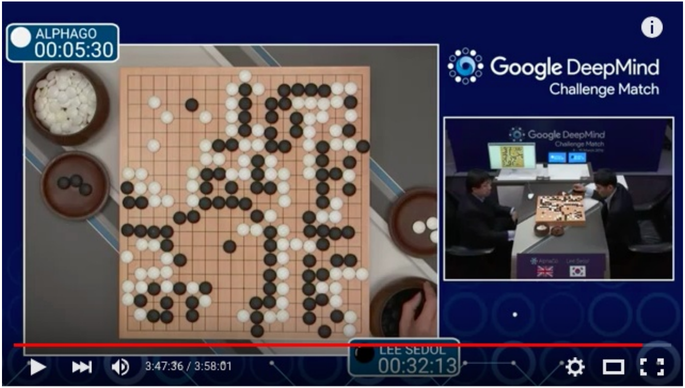
Department of Computing, Imperial College London

Introduction to Artificial Intelligence
2nd Part

AlphaGo beats World Go Champion

AlphaGo beats World Go Champion

AlphaGo beats World Go Champion



AlphaGo beats World Go Champion

Algorithm vs intuition

The five-day battle is being seen as a major test of what scientists and engineers have achieved in the sphere of artificial intelligence.

Go is a 3,000-year old Chinese board game and is considered to be a lot more complex than chess where artificial intelligence scored its most famous victory to date when IBM's Deep Blue beat grandmaster Gary Kasparov in 1997.

But experts say Go presents an entirely different challenge because of the game's incomputable number of move options which means that the computer must be capable of human-like "intuition" to prevail.

AlphaGo beats World Go Champion

Algorithm vs intuition

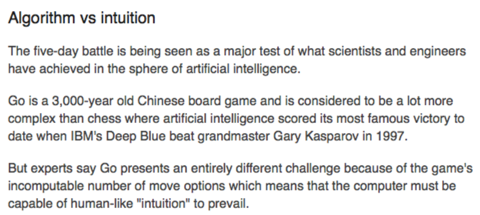
The five-day battle is being seen as a major test of what scientists and engineers have achieved in the sphere of artificial intelligence.

Go is a 3,000-year old Chinese board game and is considered to be a lot more complex than chess where artificial intelligence scored its most famous victory to date when IBM's Deep Blue beat grandmaster Gary Kasparov in 1997.

But experts say Go presents an entirely different challenge because of the game's incomputable number of move options which means that the computer must be capable of human-like "intuition" to prevail.

Welcome to scientific journalism!

AlphaGo beats World Go Champion



Algorithm vs intuition

The five-day battle is being seen as a major test of what scientists and engineers have achieved in the sphere of artificial intelligence.

Go is a 3,000-year old Chinese board game and is considered to be a lot more complex than chess where artificial intelligence scored its most famous victory to date when IBM's Deep Blue beat grandmaster Gary Kasparov in 1997.

But experts say Go presents an entirely different challenge because of the game's incomputable number of move options which means that the computer must be capable of human-like "intuition" to prevail.

Welcome to scientific journalism!

It's the number of possible positions the fundamental difference, together with the branching factor.

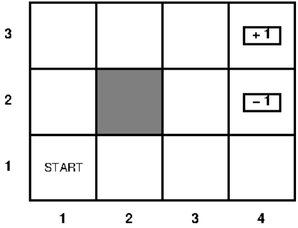
Outline

- Time
- Patience
- Risk

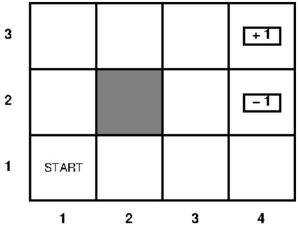
The main reference

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

The World

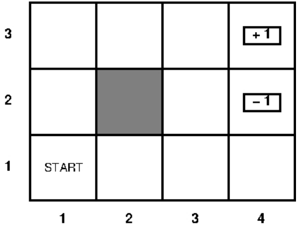


The World



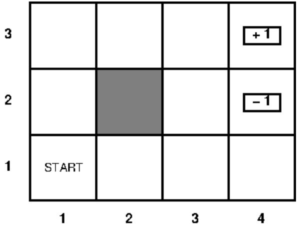
- Begin at the start state

The World



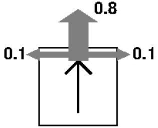
- Begin at the start state
- The game ends when we reach either goal state $+1$ or -1

The World

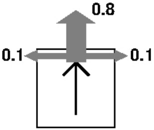


- Begin at the start state
- The game ends when we reach either goal state $+1$ or -1
- Collision results in no movement

The Agent

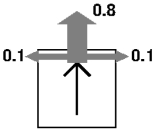


The Agent



The agent goes:

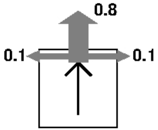
The Agent



The agent goes:

- towards the intendend direction with probability 0.8

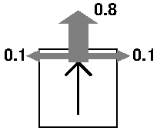
The Agent



The agent goes:

- towards the intendend direction with probability 0.8
- to the left of the intended direction with probability 0.1

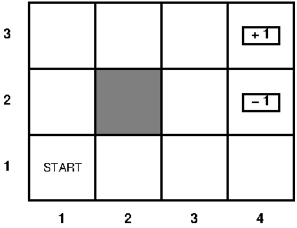
The Agent

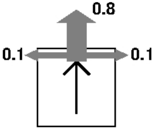


The agent goes:

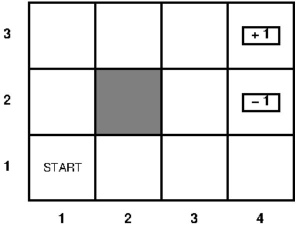
- towards the intendend direction with probability 0.8
- to the left of the intended direction with probability 0.1
- to the right of the intended direction with probability 0.1

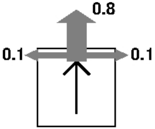
The Agent and the World





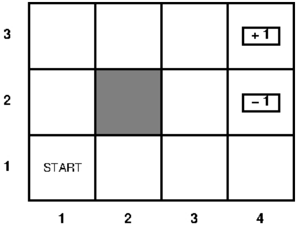
The Agent and the World

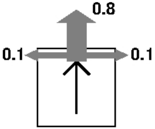




The environment is **fully observable**:

The Agent and the World

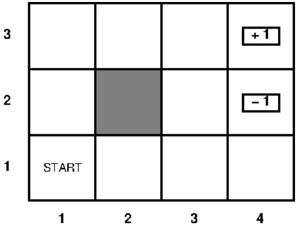


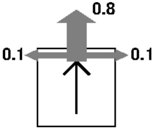


The environment is **fully observable**:

- 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 and the World

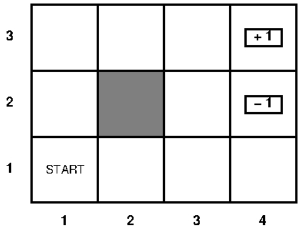


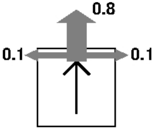


The environment is **fully observable**:

- 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 his or her position during the game, even though some trajectories might not be reached with certainty.

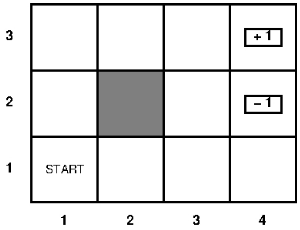
The Agent and the World

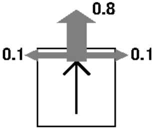




The environment is **Markovian**:

The Agent and the World

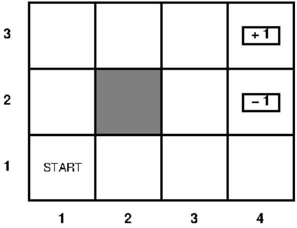


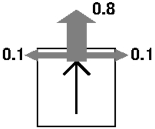


The environment is **Markovian**:

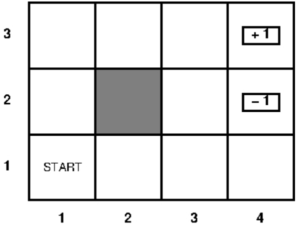
- the probability of reaching a state, only depends on the state the agent is in and the action she performs.

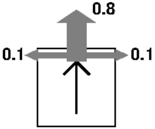
The Agent and the World





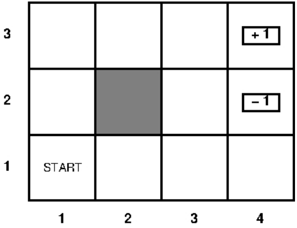
The Agent and the World

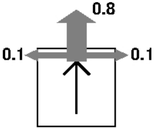




- $[x, y]_t$ is the fact that the agent is at square $[x, y]$ at time t

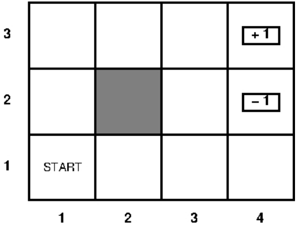
The Agent and the World

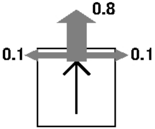




- $[x, y]_t$ is the fact that the agent is at square $[x, y]$ at time t
- $(x, y)_t$ is the fact that the agent *intends* to go to $[x, y]$ at time t

The Agent and the World

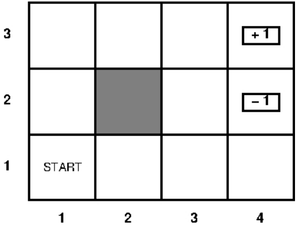


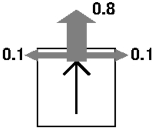


- $[x, y]_t$ is the fact that the agent is at square $[x, y]$ at time t
- $(x, y)_t$ is the fact that the agent *intends* to go to $[x, y]$ at time t

$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}) =$$

The Agent and the World

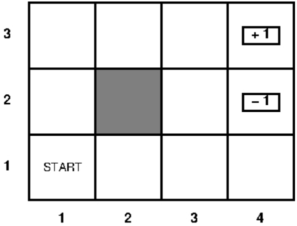


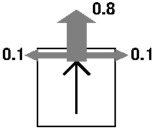


- $[x, y]_t$ is the fact that the agent is at square $[x, y]$ at time t
- $(x, y)_t$ is the fact that the agent *intends* to go to $[x, y]$ at time t

$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}) =$$
$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}, [x-5, y-6]_{t-20}) =$$

The Agent and the World

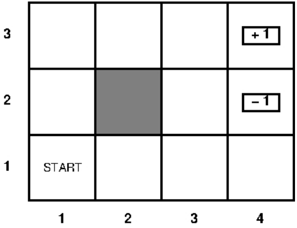


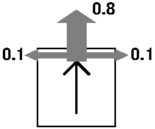


- $[x, y]_t$ is the fact that the agent is at square $[x, y]$ at time t
- $(x, y)_t$ is the fact that the agent *intends* to go to $[x, y]$ at time t

$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}) =$$
$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}, [x-5, y-6]_{t-20}) =$$
$$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}, (x-4, y-6)_{t-20})$$

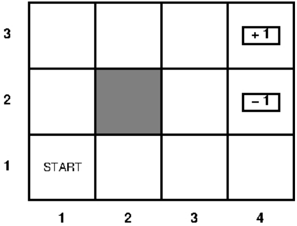
The Agent and the World

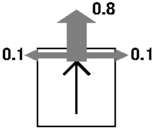




These properties allow us to make plans.

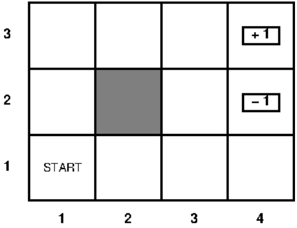
The Agent and the World

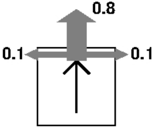




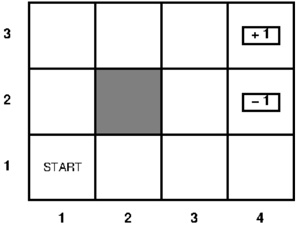
These properties allow us to make plans.
E.g., plans with determistic agents: as we know
$P([x, y]_t \mid (x, y)_{t-1}, [x-1, y]_{t-1}) = 1$

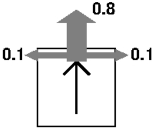
Lets make plans then





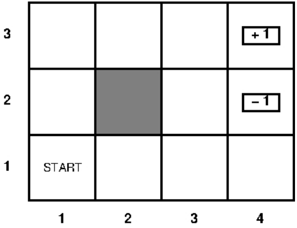
Lets make plans then

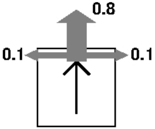




$\{Up, Down, Left, Right\}$ to denote the intended directions.

Lets make plans then

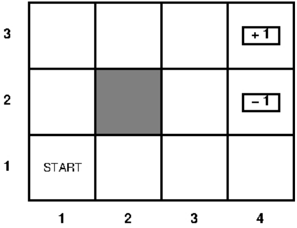


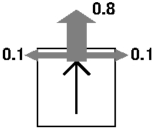


$\{Up, Down, Left, Right\}$ to denote the intended directions.

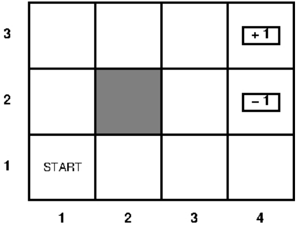
So $[Up, Down, Up, Right]$ is going to be the plan that, **from the starting state**, executes the moves n the specified order.

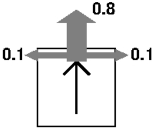
Makings plans





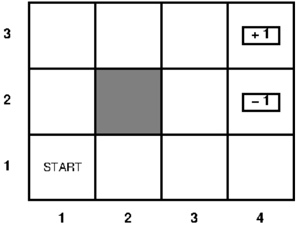
Makings plans

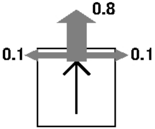




Goal: get to $+1$

Makings plans

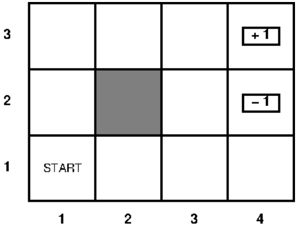


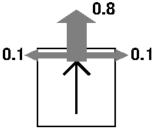


Goal: get to $+1$

Consider the plan $[\mathit{Up}, \mathit{Up}, \mathit{Right}, \mathit{Right}, \mathit{Right}]$:

Makings plans



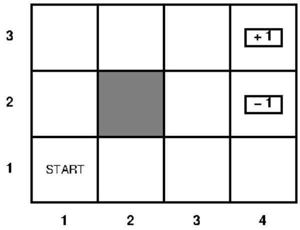


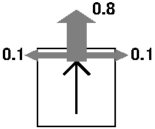
Goal: get to $+1$

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

- With deterministic agents, it gets us to $+1$ with probability 1.

Makings plans



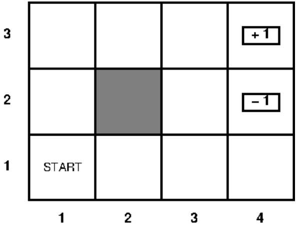


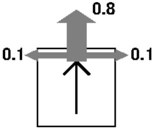
Goal: get to $+1$

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

- With deterministic agents, it gets us to $+1$ with probability 1.
- But now?

Makings plans





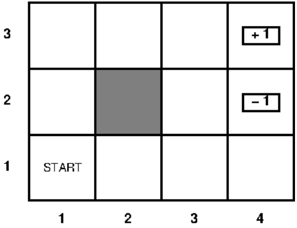
Goal: get to $+1$

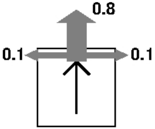
Consider the plan $[Up, Up, Right, Right, Right]$:

- With deterministic agents, it gets us to $+1$ with probability 1.
- But now?

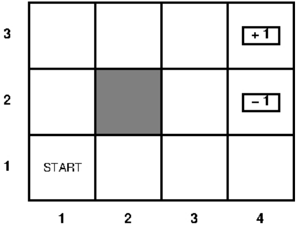
What's the probability that $[Up, Up, Right, Right, Right]$ gets us to $+1$?

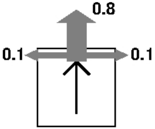
Makings plans





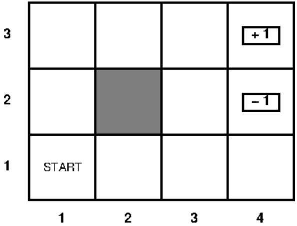
Makings plans

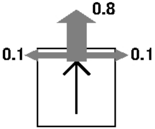




- It's not 0.8^5!

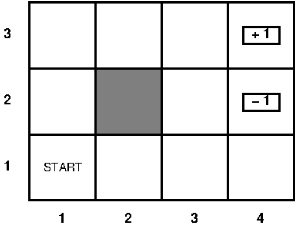
Makings plans

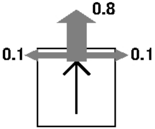




- It's not 0.8^5!
- 0.8^5 is the probability that we get to $+1$ actually using the intended plan $[Up, Up, Right, Right, Right]$

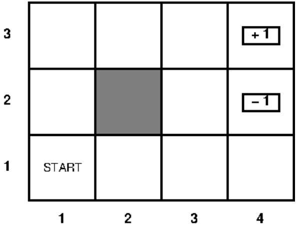
Makings plans

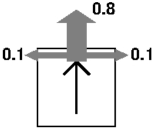




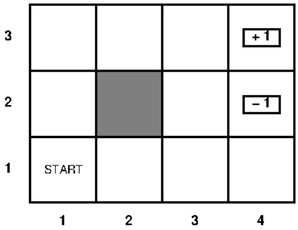
- It's not 0.8^5!
- 0.8^5 is the probability that we get to $+1$ actually using the intended plan $[Up, Up, Right, Right, Right]$
- $0.8^5 = 0.32768$: this means that we do not even get there 1 time out of 3.

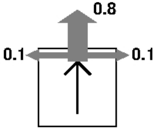
Makings plans





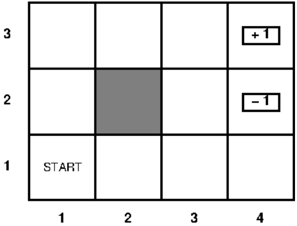
Makings plans

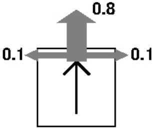




- There is a small chance of [*Up*, *Up*, *Right*, *Right*, *Right*] accidentally reaching the goal by going the other way round!

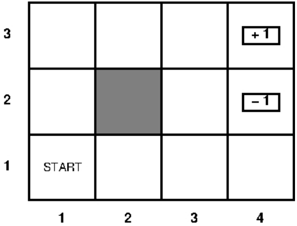
Makings plans

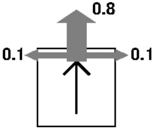




- There is a small chance of $[Up, Up, Right, Right, Right]$ accidentally reaching the goal by going the other way round!
- The probability of this to happen is $0.1^4 \times 0.8 = 0.00008$

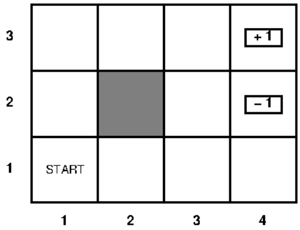
Makings plans

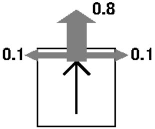




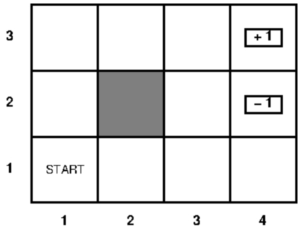
- There is a small chance of $[Up, Up, Right, Right, Right]$ accidentally reaching the goal by going the other way round!
- The probability of this to happen is $0.1^4 \times 0.8 = 0.00008$
- So the probability that $[Up, Up, Right, Right, Right]$ gets us to $+1$ is $0.32768 + 0.00008 = 0.32776$

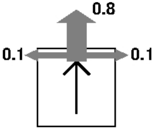
Makings plans





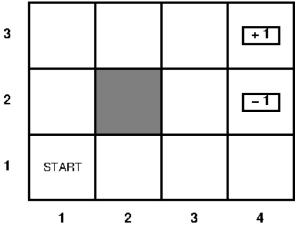
Makings plans

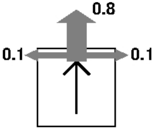




- In this case, the probability of accidental successes doesn't play a significant role. However it might very well, under different decision models, rewards, environments etc.

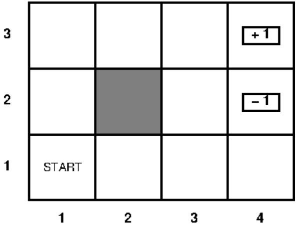
Makings plans

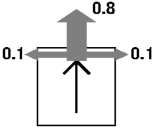




- In this case, the probability of accidental successes doesn't play a significant role. However it might very well, under different decision models, rewards, environments etc.
- 0.32776 is still less than $\frac{1}{3}$, so we don't seem to be doing very well.

Rewards

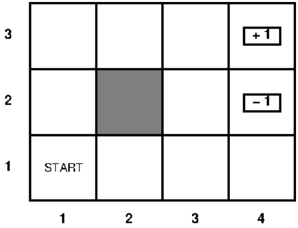


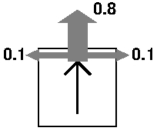


We introduce a utility function

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

Rewards



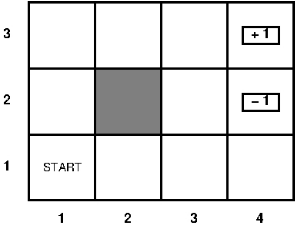


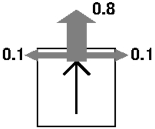
We introduce a utility function

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

r stands for rewards. To avoid confusion with established terminology, we also call it a reward function.

Rewards



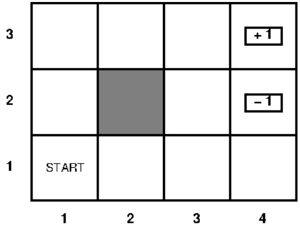


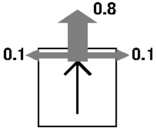
We introduce a utility function

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

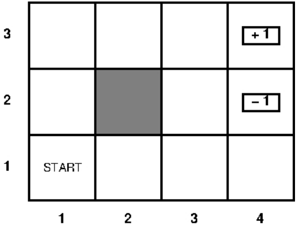
r stands for rewards. To avoid confusion with established terminology, we also call it a reward function.

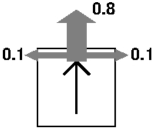
Terminology





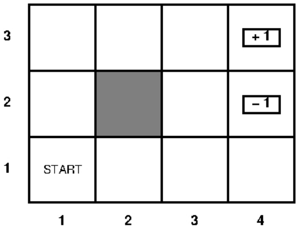
Terminology

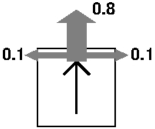




rewards for local utilities, assigned to states - denoted r

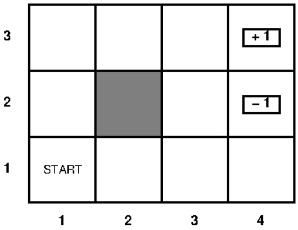
Terminology

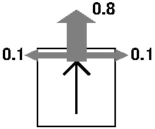




rewards for local utilities, assigned to states - denoted r
values for global long-range utilities, also assigned to states - denoted v

Terminology



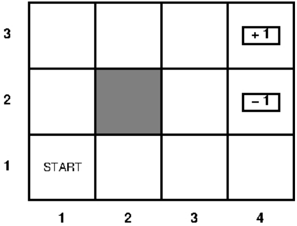


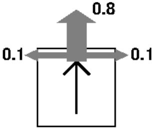
rewards for local utilities, assigned to states - denoted r

values for global long-range utilities, also assigned to states - denoted v

utility and **expected utility** used as general terms applied to actions, states, sequences of states etc. - denoted u

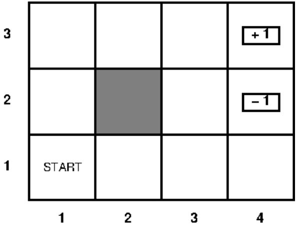
Rewards

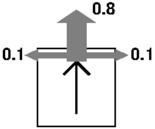




Consider now the following. The reward is:

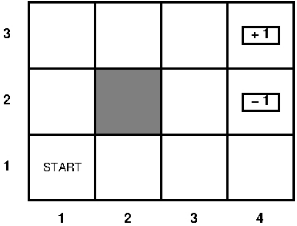
Rewards

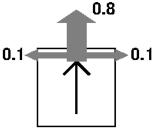




Consider now the following. The reward is:
$+1$ at state $+1$, -1 at -1, -0.04 in all other states.

Rewards

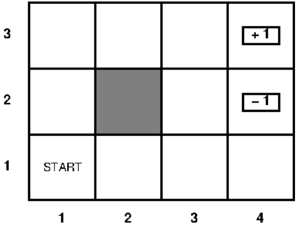


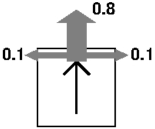


Consider now the following. The reward is:
$+1$ at state $+1$, -1 at -1, -0.04 in all other states.

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

Rewards



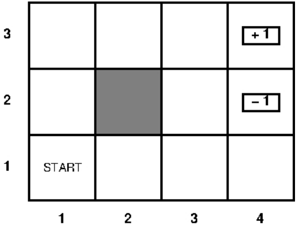


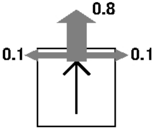
Consider now the following. The reward is:
$+1$ at state $+1$, -1 at -1, -0.04 in all other states.

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 -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

We need to compare **sequences** of states.

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

multi-criteria decision making

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

multi-criteria decision making

Many ways of comparing states:

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

multi-criteria decision making

Many ways of comparing states:

- summing all the rewards

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

multi-criteria decision making

Many ways of comparing states:

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

Utility of state sequences

We need to compare **sequences** of states.
Look at the following:
$u[s_1, s_2, \ldots s_n]$ is the utility of sequence $s_1, s_2, \ldots s_n$.
Does it remind you of anything?

multi-criteria decision making

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:

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

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

- Additive *utility function:*

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

- Additive *utility function:*
  $u([s_0, s_1, s_2, \ldots]) = r(s_0) + r(s_1) + r(s_2) + \cdots$

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

- Additive *utility function:*
  $u([s_0, s_1, s_2, \ldots]) = r(s_0) + r(s_1) + r(s_2) + \cdots$
- Discounted *utility function:*

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

- Additive *utility function:*
  $u([s_0, s_1, s_2, \ldots]) = r(s_0) + r(s_1) + r(s_2) + \cdots$
- Discounted *utility function:*
  $u([s_0, s_1, s_2, \ldots]) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \cdots$

Utility of state sequences

Theorem

There are only two ways to combine rewards over time.

- Additive *utility function:*
  $u([s_0, s_1, s_2, \ldots]) = r(s_0) + r(s_1) + r(s_2) + \cdots$
- 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**

Discount factor

Discount factor

γ is a measure of the agent patience. How much more she values a gain of 5 today than a gain of 5 tomorrow, the day after etc...

Discount factor

γ is a measure of the agent patience. How much more she values a gain of 5 today than a gain of 5 tomorrow, the day after etc...

- Used everywhere in AI, game theory, cognitive psychology

Discount factor

γ is a measure of the agent patience. How much more she values a gain of 5 today than a gain of 5 tomorrow, the day after etc...

- Used everywhere in AI, game theory, cognitive psychology
- A lot of experimental research on it

Discount factor

γ is a measure of the agent patience. How much more she values a gain of 5 today than a gain of 5 tomorrow, the day after etc...

- Used everywhere in AI, game theory, cognitive psychology
- A lot of experimental research on it
- Variants: hyperbolic discounting

Discounting

Discounting

With discounted rewards the utility of an infinite sequence if **finite**

Discounting

With discounted rewards the utility of an infinite sequence if **finite**
In fact, if $\gamma < 1$ and rewards are bounded by $\mathbf{r}$, we have:

Discounting

With discounted rewards the utility of an infinite sequence if **finite**
In fact, if $\gamma < 1$ and rewards are bounded by $\mathbf{r}$, we have:

$$u[s_1, s_2, \ldots] = \sum_{t=0}^{\infty} \gamma^t r(s_t) \leq \sum_{t=0}^{\infty} \gamma^t \mathbf{r} = \frac{\mathbf{r}}{1-\gamma}$$

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:

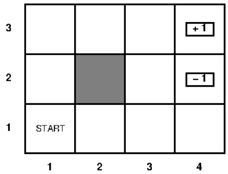
- fully observable environment
- with stochastic actions
- with a Markovian transition model

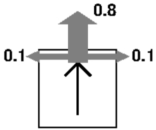
Markov Decision Process

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

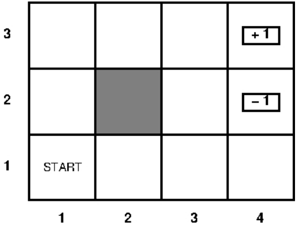
States $s \in S$, actions $a \in A$

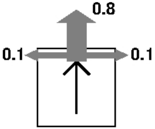
Model $P(s'|s,a)$ = probability that a in s leads to s'

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

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

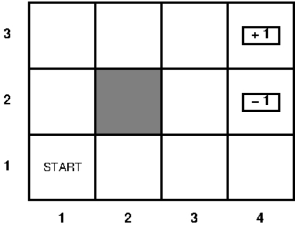
Value of plans

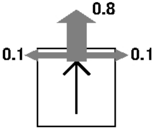




The utility of executing a plan p from state s is given by:

Value of plans

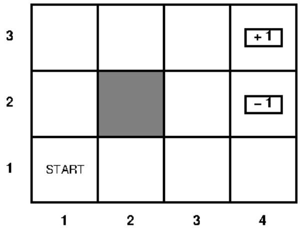


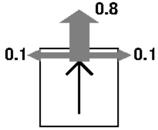


The utility of executing a plan p from state s is given by:

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

Value of plans



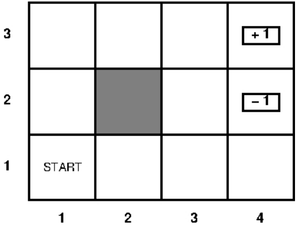


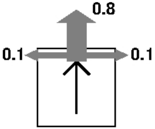
The utility of executing a plan p from state s is given by:

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

Where S_t is a random variable and the expectation is wrt to the probability distribution over state sequences determined by s and p.

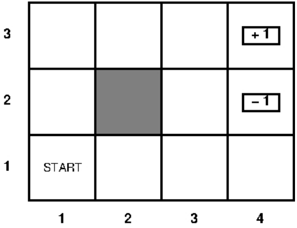
Value of plans

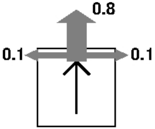




- Calculate the utility of the sequences you can actually perform, times the probability of reaching them.

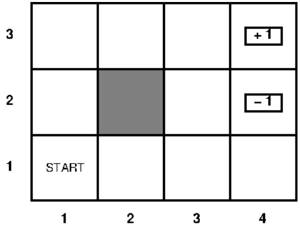
Value of plans

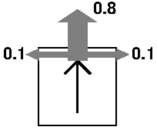




- Calculate the utility of the sequences you can actually perform, times the probability of reaching them.
- Add these numbers

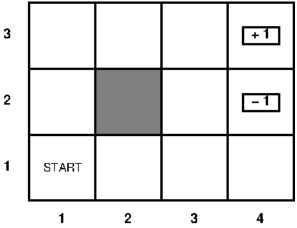
Value of plans

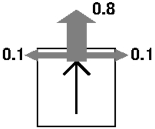




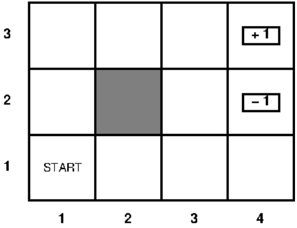
- Calculate the utility of the sequences you can actually perform, times the probability of reaching them.
- Add these numbers
- Forget about the rest

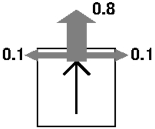
Value of plans





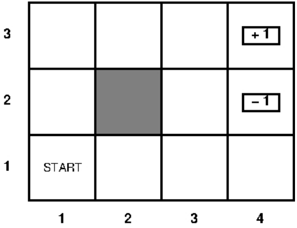
Value of plans

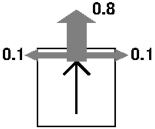




For instance the plan $[Up, Up]$ can generate sequences

Value of plans

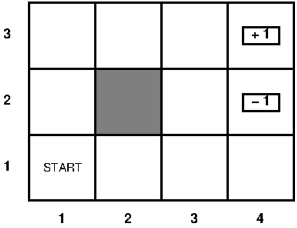


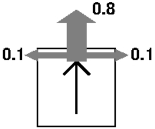


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[2,1],[3,1])$ with probability 0.8^2

Value of plans

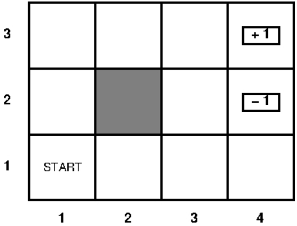


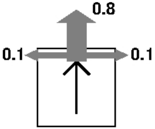


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[2,1],[3,1])$ with probability 0.8^2
- $([1,1],[2,1],[2,1])$ with probability $2 \times 0.8 \times 0.1$ (collisions)

Value of plans

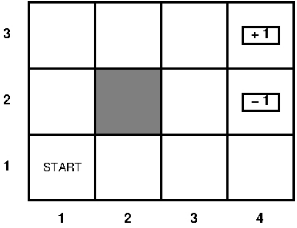


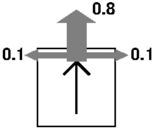


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[2,1],[3,1])$ with probability 0.8^2
- $([1,1],[2,1],[2,1])$ with probability $2 \times 0.8 \times 0.1$ (collisions)
- $([1,1],[1,2],[1,2])$ with probability 0.1×0.8

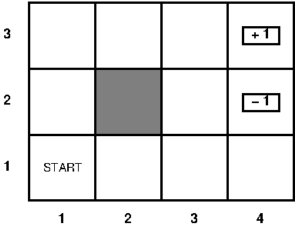
Value of plans

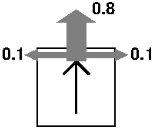




For instance the plan $[Up, Up]$ can generate sequences

Value of plans

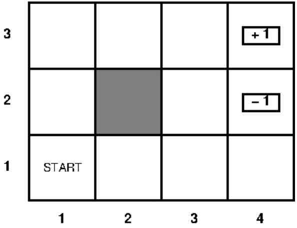


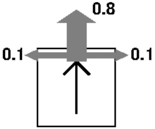


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,1],[1,1])$ with probability 0.1^2

Value of plans

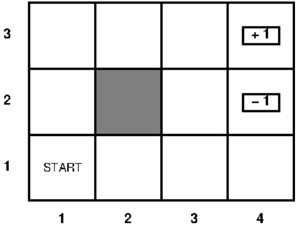


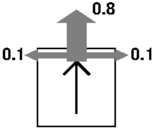


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,1],[1,1])$ with probability 0.1^2
- $([1,1],[1,1],[2,1])$ with probability 0.1×0.8

Value of plans

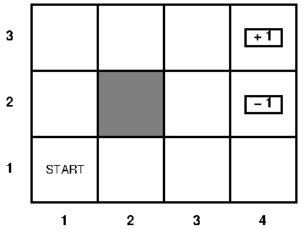


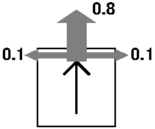


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,1],[1,1])$ with probability 0.1^2
- $([1,1],[1,1],[2,1])$ with probability 0.1×0.8
- $([1,1],[1,1],[1,2])$ with probability 0.1^2

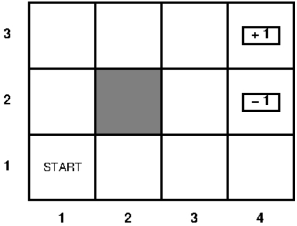
Value of plans

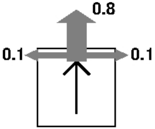




For instance the plan $[Up, Up]$ can generate sequences

Value of plans

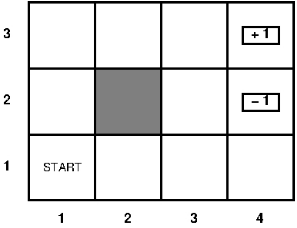


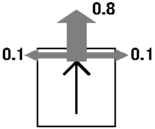


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,2],[1,1])$ with probability 0.1^2

Value of plans

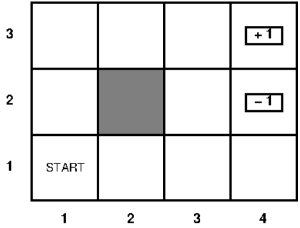


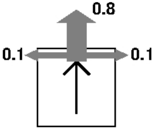


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,2],[1,1])$ with probability 0.1^2
- $([1,1],[1,2],[1,3])$ with probability 0.1^2

Value of plans

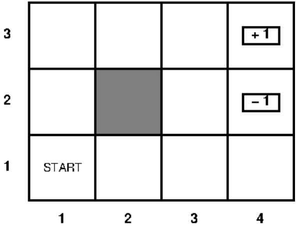


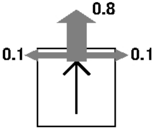


For instance the plan $[Up, Up]$ can generate sequences

- $([1,1],[1,2],[1,1])$ with probability 0.1^2
- $([1,1],[1,2],[1,3])$ with probability 0.1^2
- for a total of nine sequences

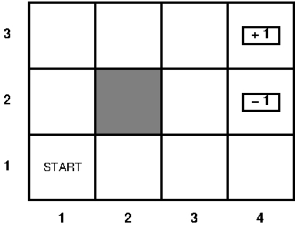
Value of plans

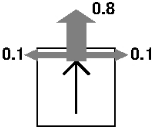




Adding utility and summing up, we have that the expected utility is -0.08

Value of plans





Adding utility and summing up, we have that the expected utility is -0.08

To be expected, because no matter how we proceed, we are making two steps and at each step getting -0.04 of reward.

Plans vs Policies

- We have looked at a finite sequence of actions. But why should the agent stop after, say, five steps, if she can reach the terminal states in a few steps?

Plans vs Policies

- We have looked at a finite sequence of actions. But why should the agent stop after, say, five steps, if she can reach the terminal states in a few steps?
- The intuitively "best" course of action is not getting us there in $\frac{2}{3}$ of the cases, even if we count getting unwanted trajectories.

Plans vs Policies

- We have looked at a finite sequence of actions. But why should the agent stop after, say, five steps, if she can reach the terminal states in a few steps?
- The intuitively "best" course of action is not getting us there in $\frac{2}{3}$ of the cases, even if we count getting unwanted trajectories. Can we do better?

Plans vs Policies

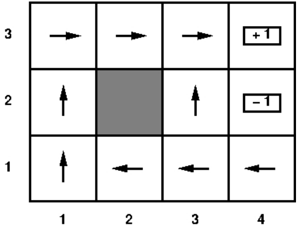
- We have looked at a finite sequence of actions. But why should the agent stop after, say, five steps, if she can reach the terminal states in a few steps?
- The intuitively "best" course of action is not getting us there in $\frac{2}{3}$ of the cases, even if we count getting unwanted trajectories. Can we do better?
- The idea is that we don't only care about specifying a sequence of moves, but we need to think of what to do **in each situation**.

Plans vs Policies

- We have looked at a finite sequence of actions. But why should the agent stop after, say, five steps, if she can reach the terminal states in a few steps?
- The intuitively "best" course of action is not getting us there in $\frac{2}{3}$ of the cases, even if we count getting unwanted trajectories. Can we do better?
- The idea is that we don't only care about specifying a sequence of moves, but we need to think of what to do **in each situation**.

A **policy** is a specification of moves at each decision point

A policy



Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

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

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

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

The probability distribution over the state sequences is induced by the policy π, the initial state t and the transition model for the environment.

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

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

The probability distribution over the state sequences is induced by the policy π, the initial state t and the transition model for the environment.

We want the **optimal** policy:

$$\pi^*_s = \operatorname*{argmax}_{\pi} v^{\pi}(s)$$

A remarkable fact

Theorem

With discounted rewards and infinite horizons
$\pi^*_s = \pi^*_{s'}$, *for each* $s' \in S$

A remarkable fact

Theorem

With discounted rewards and infinite horizons
$\pi^*_s = \pi^*_{s'}$, *for each* $s' \in S$

Idea: Take π^*_a and π^*_b. If they both reach a state c, because they are both optimal, there is no reason why they should disagree. So π^*_c is identical for both. But then they behave the same at all states!

Optimal policies

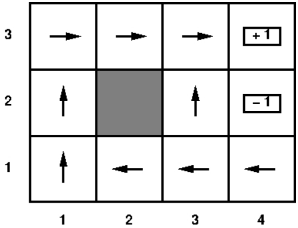
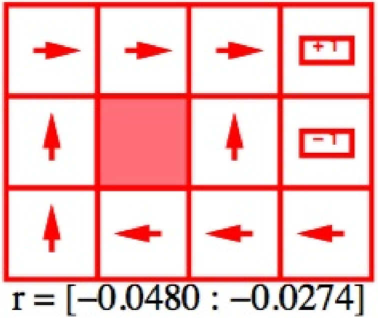
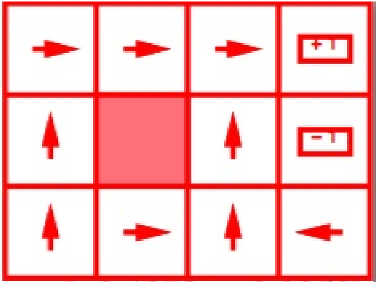


Figure: Optimal policy when state penalty $R(s)$ is –0.04:

Risk and reward

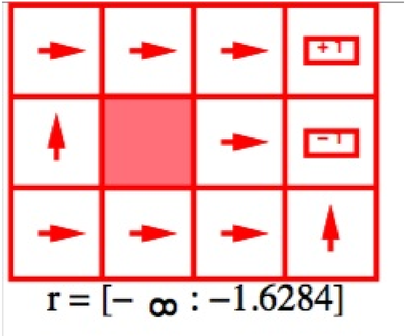


Risk and reward

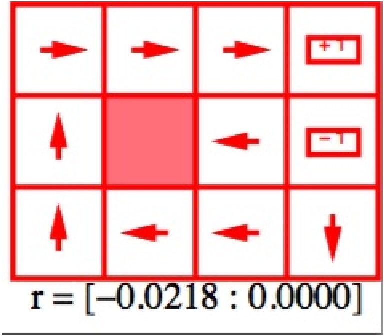


$r = [-0.4278 : -0.0850]$

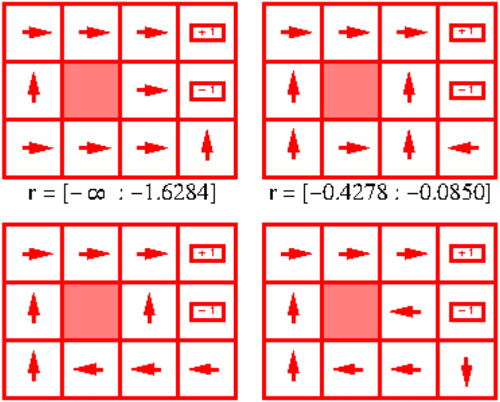
Risk and reward



Risk and reward



Risk and reward



r = [−∞ : −1.6284]   r = [−0.4278 : −0.0850]

r = [−0.0480 : −0.0274]   r = [−0.0218 : 0.0000]

To be continued

- Next Tuesday we are going to finish the slides on MDPs