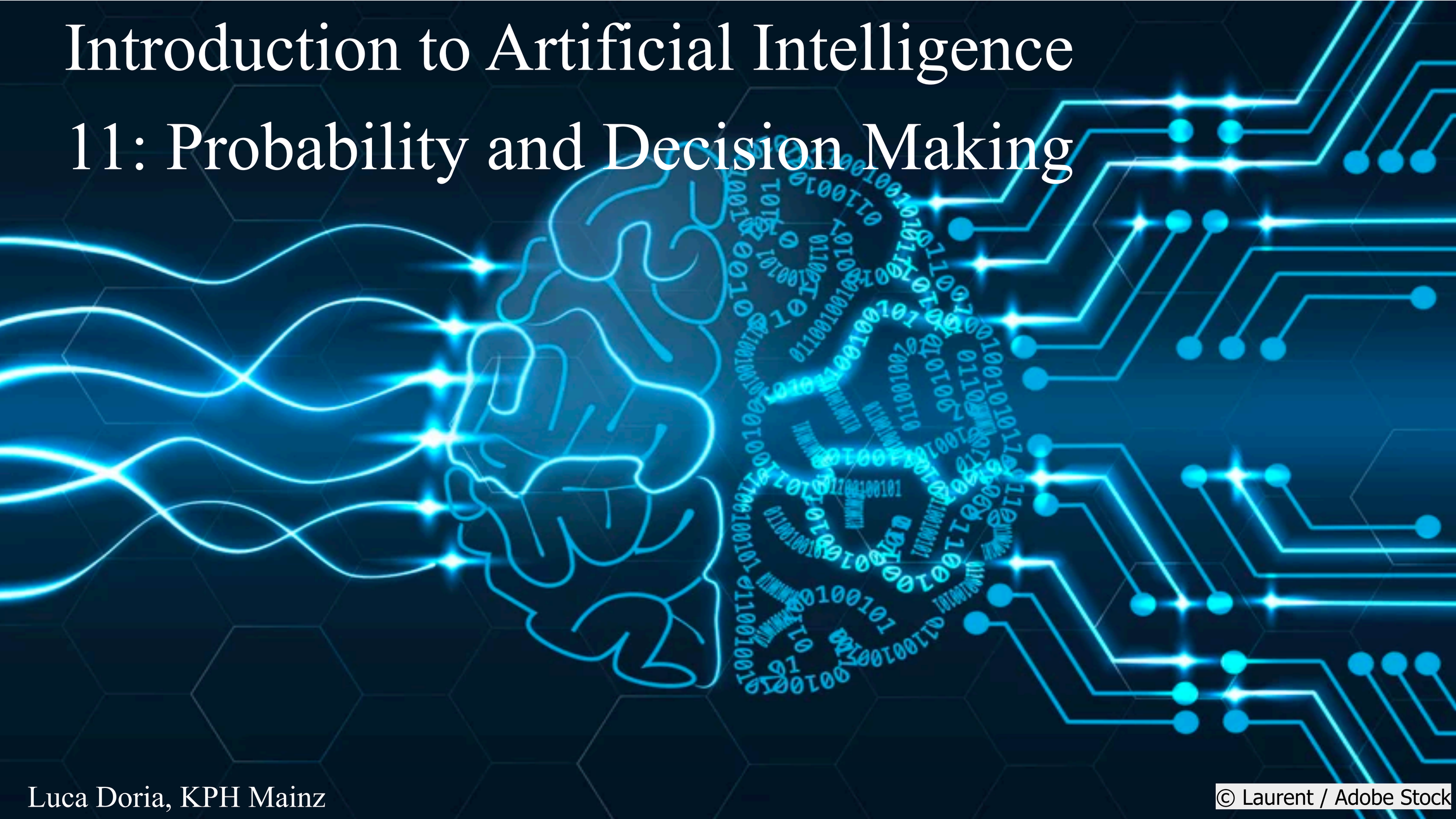
Introduction to Artificial Intelligence

11: Probability and Decision Making



Luca Doria, KPH Mainz

Introduction

- How can an agent decide the next action? A single percept might not be enough.
- Agents can have preferences.
- Preferences lead to the quantitative concept of **utility**.
- How to specify an utility function.
- How to calculate an utility function.

The St. Petersburg Paradox

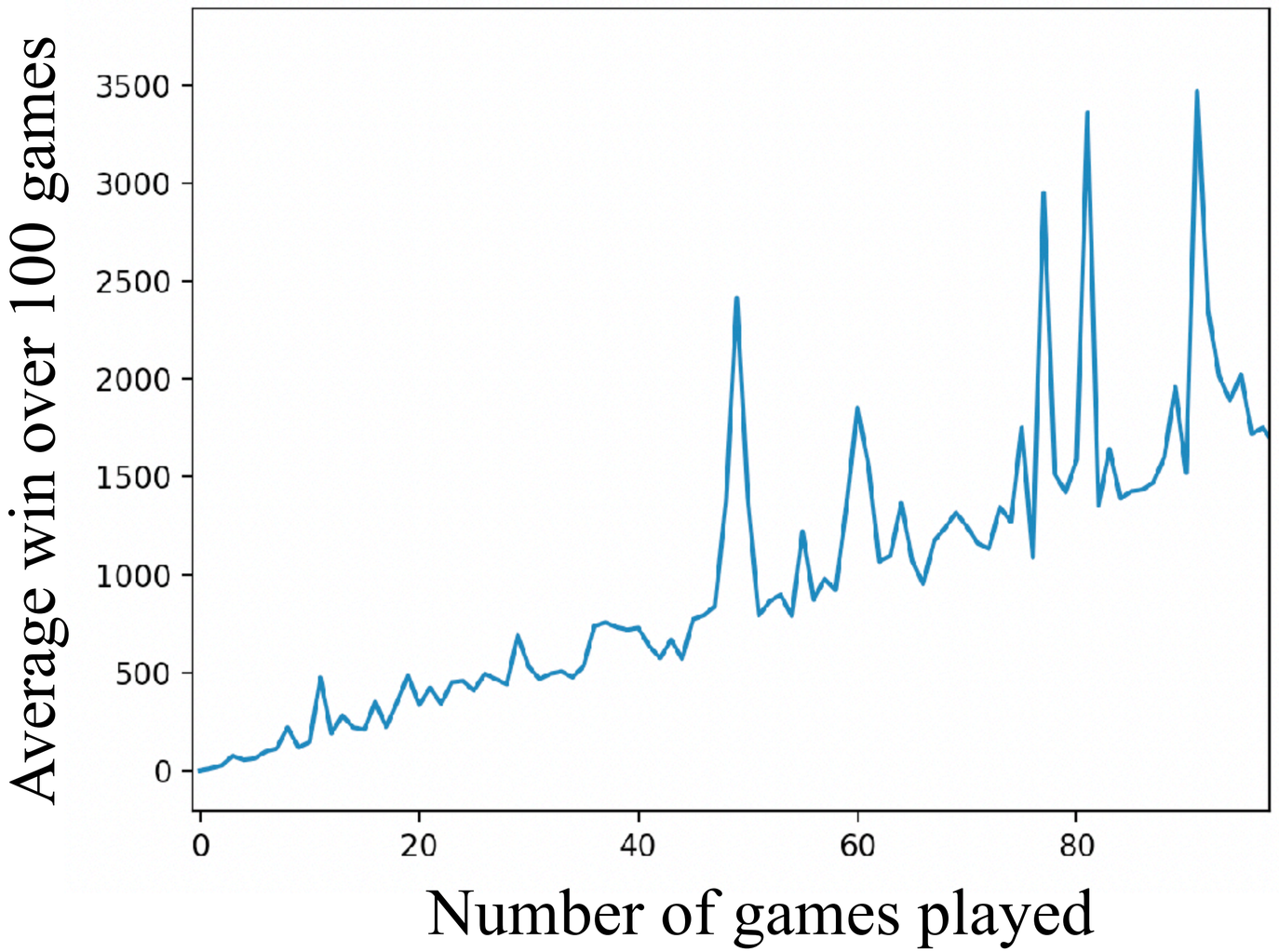
Consider the following gambling game ("St. Petersburg Game"):
A (fair) coin is tossed: if it lands on head, you win 2$, if tail you loose.
If the coin is tossed again and lands on head, you win 2$ and so on…
The expected win is therefore:

$$E = \frac{1}{2} \times 2 + \frac{1}{4} \times 4 + \ldots = 1 + 1 + \ldots = \infty$$

Problem: how much should the casino' ask for playing, given the fact that you could win a potentially infinite amount of money?

Moreover: how much are you willing to pay?

The St. Petersburg Paradox



A Solution?

- From a mathematical point of view there is no problem: the average win is infinite. It makes sense to accept an infinitesimal probability to win an infinite sum!

- In fact, the paradox highlight a decision problem. Some people will be willing to risk a large sum for a potential high gain, other ones will be happy with modest wins.

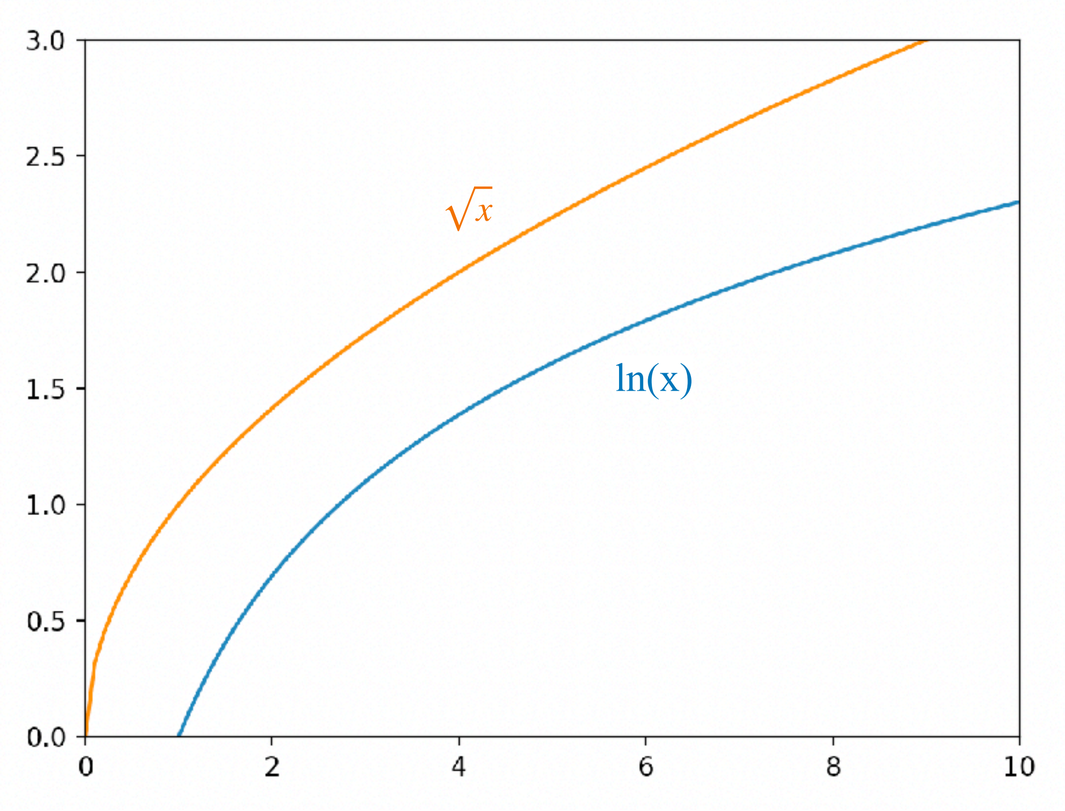
- These considerations brought mathematicians, economists, sociologists,… to develop the idea of (marginal) utility.

- Basic idea: a price or an amount of money is not the same for everybody. 1000$ is a lot for a “poor” person and “nothing” for a very rich person. The price/value is not everything but is related to the utility of the object/money for a certain person.

Developing Utility Functions

The utility should have a “saturation” property: its growth should diminish as function of the value/price of the considered good (for example).

Early examples of utility functions were ln(x) and $\sqrt{x}$.



Possible condition: $\lim_{x \to \infty} \frac{du}{dx} = 0$

Decisions under uncertainty

- Consider an agent that must take a decision and considers an action **a**.
- We assign a probability to each possible current state **s**: **P(s)**.

- The probability that an action **a** makes the agent transition from the state s to s' is

$$P(s' \mid s, a)$$

- We are interested in the outcome of the action: P(Result(a)=s')
- The previous probabilities are related by

$$P(Result(a) = s') = \sum_{s} P(s)P(s' \mid s, a)$$

Maximum Expected Utility (MEU)

The simplest form of decision is the one concerned with immediate outcomes in an episodic environment.

The preferences of an agent are encoded in an **utility function** U(s) which associates a number at each state, expressing how desirable it is.

The **expected utility** EU(a) is the average utility of the possible outcomes:

$$EU(a) = \sum_{s'} P(Result(a) = s')U(s')$$

The **principle of maximum expected utility** expresses the fact that an agent chooses:

$$action = argmax_a EU(a)$$

Meaning of MEU

The MEU is a way to encode a prescription of an “intelligent” agent.

This principle just formalises the concept, but it is not constrictive.

Constructing the probability distribution P(s) over all the states of the world requires:

- Perception
- Learning
- Knowledge representation (e.g. a logic)
- Inference rules

The same is true for U(s’) which is function of the outcomes s’.

Remember the performance measures? U is a form of them.

Rational Preferences

The MEU is just one possibility and we have to better formalise what a preference means.

Notation:

$A \succ B$: A is preferred over B,

$A \sim B$: indifferent between A and B,

$A \succeq B$: A is preferred over B or the agent is indifferent.

A,B,… can be thought as "lotteries" L where each possible outcome S_i can happen with a certain probability p_i such that

$$L = \{p_1,S_1 ; p_2,S_2 ; \ldots ; p_n, S_n\}$$

Axioms (1)

Orderability: the agent must choose thus exactly one of those holds:
$A \succ B$, $A \sim B$, or $A \succeq B$.

Transitivity: $(A \succ B) \wedge (B \succ C) \Rightarrow A \succ C$

Continuity: $A \succ B \succ C \Rightarrow \exists p [p, A; 1 - p, C] \sim B$.
If B is between two choices then there exists a probability for which the agent is indifferent in choosing B or A with prob. p or C with prob. 1-p .

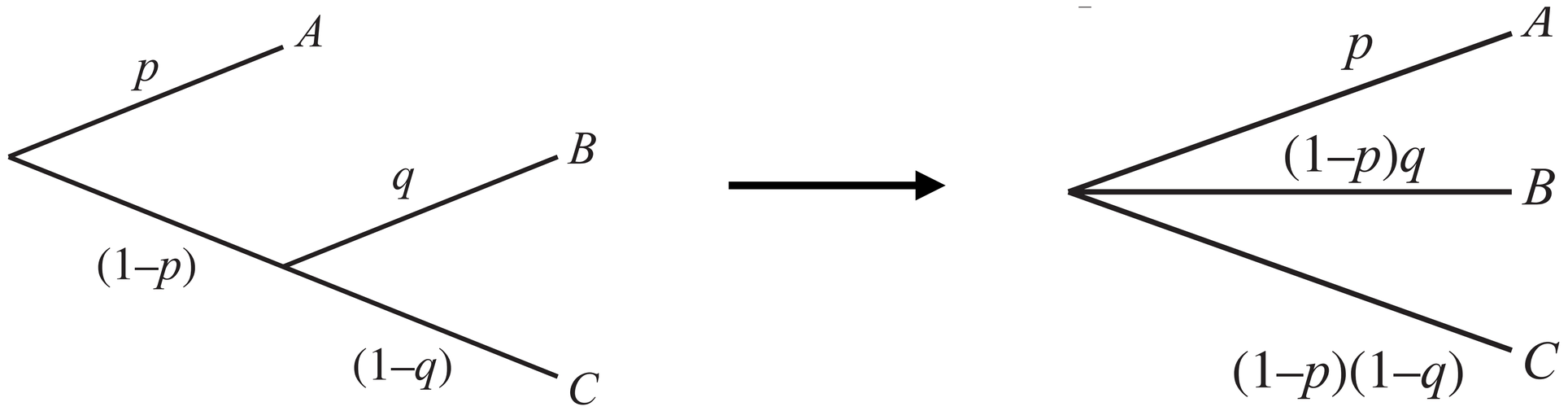
Axioms (2)

Substitutability: $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$ (Holds also for $\succ$).

Monotonicity: $A \succ B \Rightarrow (p > q \Leftrightarrow [p, A; 1-p, B]) \succ [q, A; 1-q, B]$
Two lotteries have two possible outcomes (A or B).
If A is preferred, then the agent must prefer the lottery where A has the highest probability.

Axioms (3)

Decomposability:
$[p, A; 1-p, [q, B; 1-q\,.\,C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$
Compound lotteries can be reduced to simpler ones with the laws of probability.
Two consecutive lotteries are compressed in a single (equivalent) one.



Utility

The previous axioms were about **preferences**.
From them, we can derive the **utility** concept: a map from lotteries to numbers.

Theorem (Existence of an Utility Function)
If the agent's preferences obey the previous axioms, then there exists a function U such that U(A)>U(B) iff A$\succ$B and U(A)=U(B) iff A$\sim$B.

Theorem (Expected utility of a lottery)
The utility of a lottery is the sum of the probability of each outcome times the utility of that outcome:

$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i)$$

Theorem (Unicity)
The utility function is not unique: $U'(S) = aU(S) + b$

Utility: the money case

You won 1000EUR but the game offers you to flip a coin:

- Tail: you loose what you have.
- Head: you get 3000EUR.

In principle the expected win is:
0EURx0.5 + 3000EURx0.5 = 1500EUR > 1000EUR, so you should accept to flip.

What is the “rational” choice you have to make?

Utility: the money case

If S_n is the state where we possess a total of nEUR and S_k is the event where we have a current wealth of kEUR, in terms of expected utility functions we have:

$$EU(\text{Accept to flip}) = 0.5 \times U(S_k) + 0.5 \times U(S_{k+3000})$$
$$EU(\text{No flip}) = U(S_{k+1000})$$

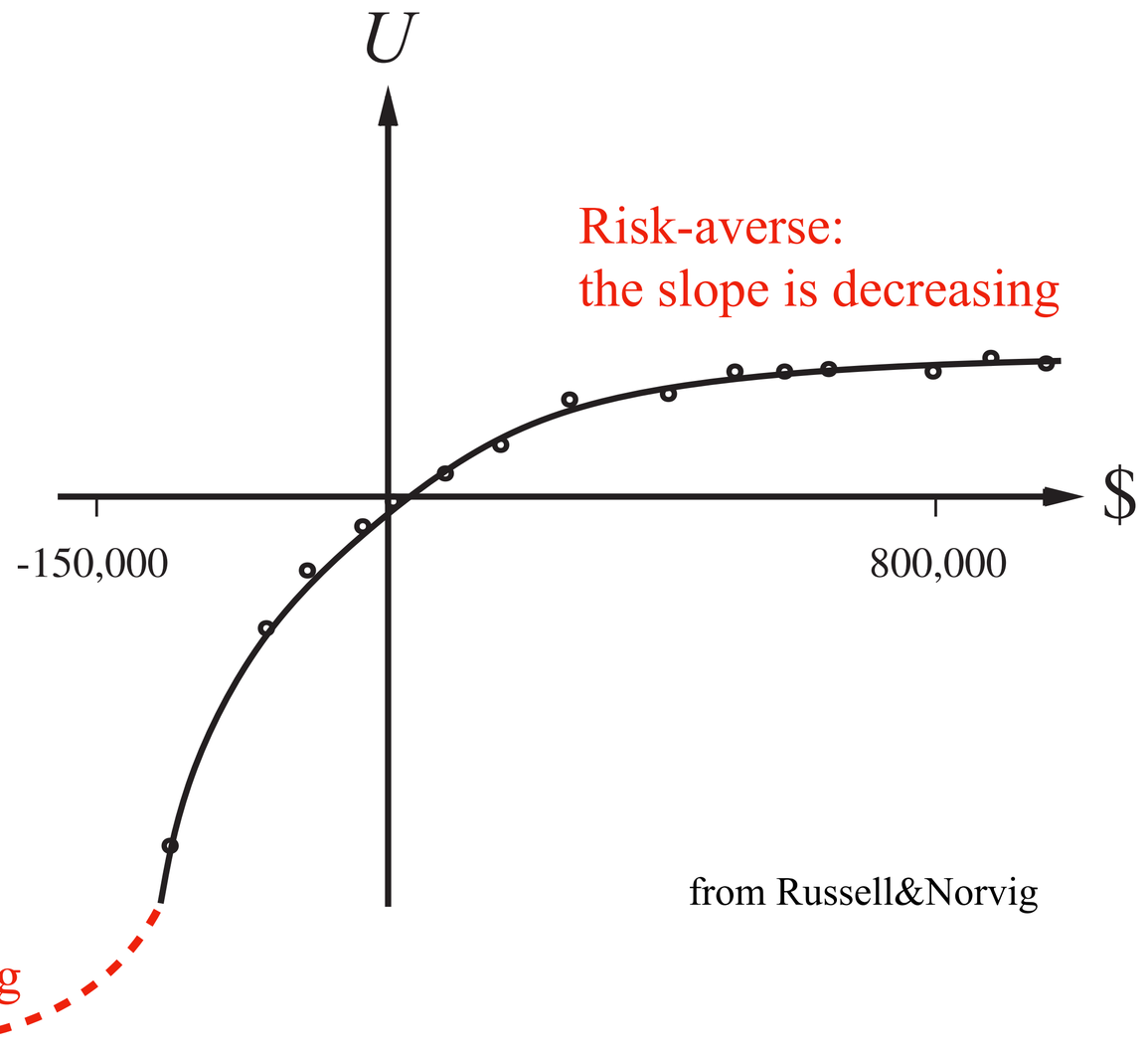
To decide what to do, we have to specify U.
Remember that U is not proportional to the amount of money, since the first sum will look very important to you, while adding more money will be less and less important.
The question can be: are you a risk-averse or a risk-seeking agent?

Utility: the money case

In one of the first empirical studies (in the 1960s), it emerged that for money, the utility function is very close to a logarithm (as the intuition of Bernoulli hundreds of years before).
Example: in an empirical study, the utility function of a specific person was consistent with:

$U(S_{k+n}) = -263.3 + 22.09 \log (n+150.000)$.

with -150.000$<n<800.000$



Relation to Insurance

Empirically, people prefer to gain 400EUR than gamble between 0 and 1000EUR.
The expected monetary value (EMV) of the gamble is 0x0.5+1000x0.5 = 500EUR.

The difference:

EMV - 400EUR = 100EUR

is called **insurance premium**.
The fact that the premium is positive, is the basis of the insurance industry: most people prefers to pay a small price but avoid large losses (car or house insurance…)

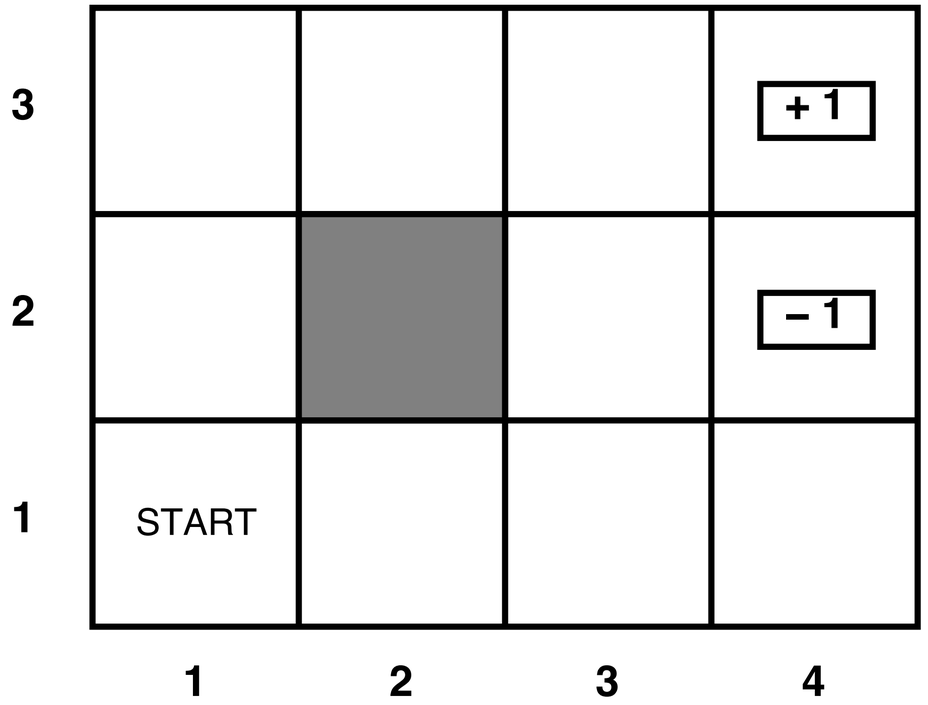
Sequential Decision Problems

Model example:

From the START square, the agent must choose an action (e.g. direction) at each time step.

The process terminates if one of the goal states (with the associated rewards) is reached.

In each square, the available actions are up, down, left, right.



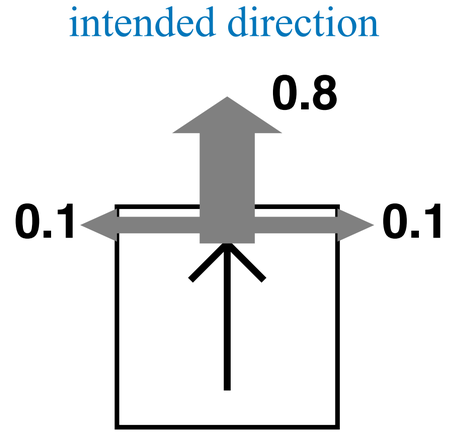
from Russell&Norvig

Sequential Decision Problems

Two versions of the problem:

Deterministic:
the intended direction is always taken, except when directed against a wall.

Stochastic:
the intended direction is taken with 0.8 probability, while with 0.1 probability it takes the one of the two orthogonal directions wrt the intended one.



from Russell&Norvig

Markov Decision Problem

Given a set of states, a Markov Decision Problem is defined by:

- Initial state S_0
- Transition Model P(s,a,s')
- Reward Function R(s)

P is the probability to reach the state s' from the state s if the action a is taken.

Policy π(s): A function mapping states to actions.

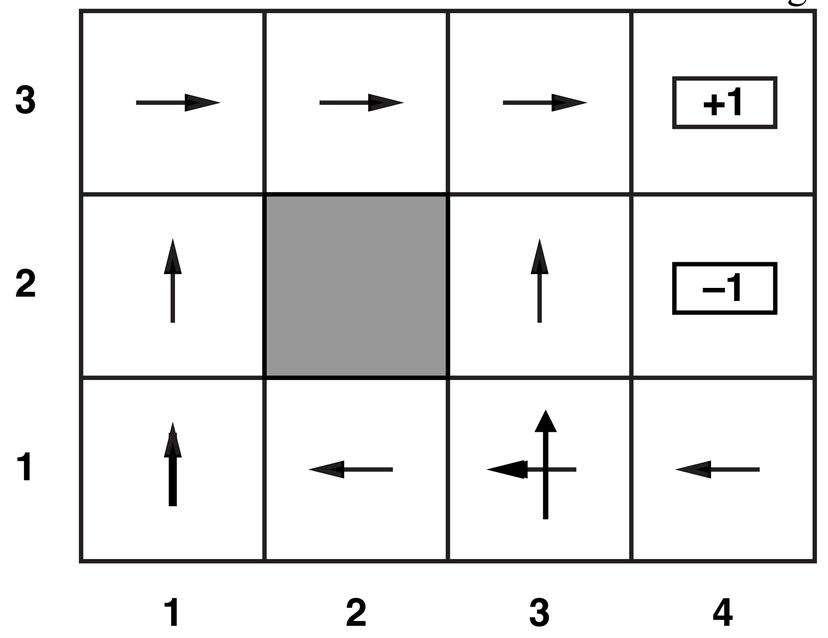
Optimal policy π^*: optimises the future expected reward.

Optimal Policies

- If an agent has an optimal policy, he can use its current percept letting him know its current state s.
- Given the state, he executes the action $\pi^*(s)$
- The last points realise a reflex agent.

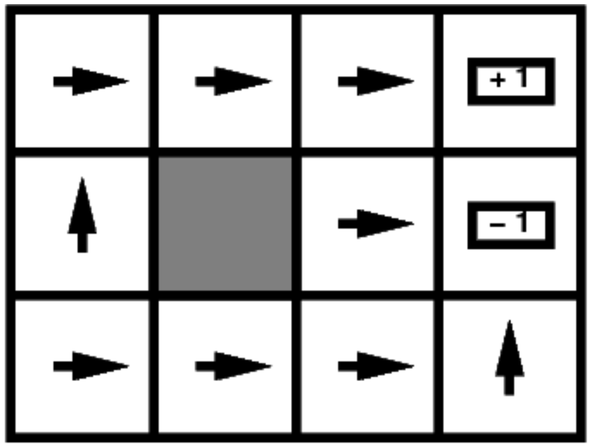
Example: optimal policy for a reward R=-0.04 for each transition.
(Rewards sum up)

from Russell&Norvig

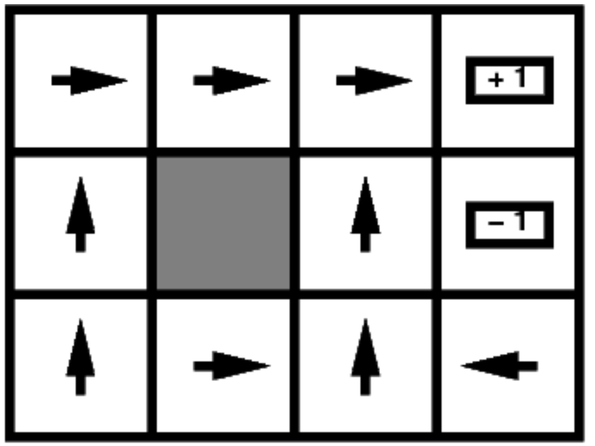


Different Reward Models

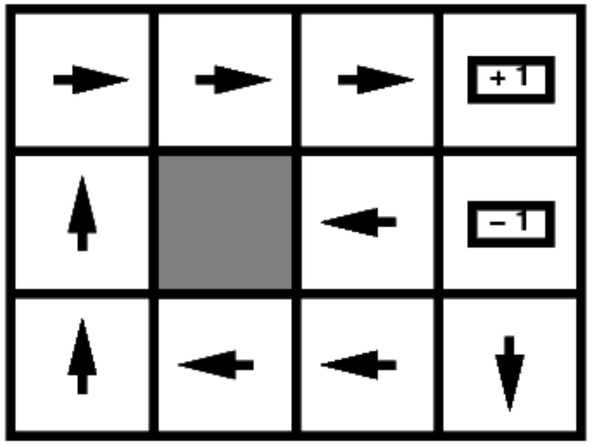
A move is so expensive that the agent head to whatever exit is the closest.



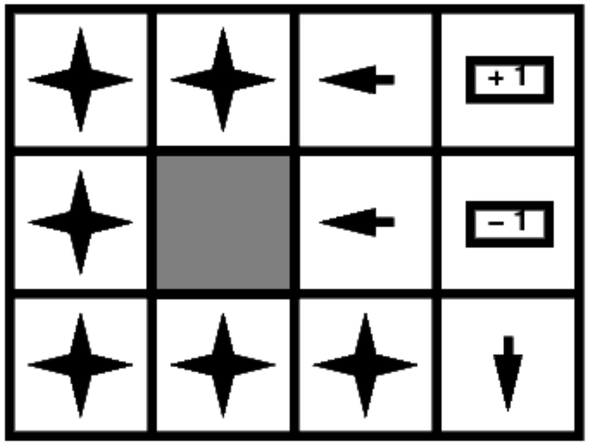
$R(s) < -1.6284$



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



$-0.0221 < R(s) < 0$



$R(s) > 0$

from Russell&Norvig

The reward for each move is so high that the agent prefers to keep moving and avoid exits.

Finite and Infinite Horizons

- The performance of an agent is calculated with the sum of rewards for the states visited.
- For determining an optimal policy we first calculate the utility of each state and then use the single state utilities for deciding the best action.
- The result will depend on whether we have a finite or infinite horizon problem.
- Utility function for state sequences: $U_h([s_0,s_1,\ldots,s_n])$
- Finite horizon: $U_h([s_0,s_1,\ldots,s_{N=k}])=U_h([s_0,s_1,\ldots,s_N])\ \forall k>0$
- Finite horizon problems are also called non-stationary, since there is a time dependence.
- Infinite horizon problems are instead stationary.

Utility of State Sequences

In the case of stationary systems, we can define the utilities with:

- Additive rewards:
$U_h([s_0,s_1,\ldots,]) = R(s_0) + R(s_1) + R(s_2) + \ldots$

- Discounted rewards:
$U_h([s_0,s_1,\ldots,]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

where $\gamma \in [0,1)$ is the discount factor.
With discounted rewards the utility of an infinite sequence becomes finite:

$$U_h([s_0,s_1,\ldots,]) = \sum_{t=0}^{\infty} \gamma^t R \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{\pm R_{max}}{1-\gamma}$$

Discount factors disfavour contributions too far ahead in the future.

Utility of States

- The utility of a state depends on the utility of the states that follow.
- Let $U^{\pi}(s)$ be the utility of a state s with policy π.
- Let s_t be the state after executing π for t steps. The utility is:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \,|\, \pi, s_0 = s\right]$$

- The true utility U(s) of a state s is $U^{\pi^*}(s)$
- R(s) is the short-term reward for being in the state s
- U(s) is the long-term total reward from s, onwards.

Optimal Policy

Once we define the police corresponding to s as starting point, we can define the optimal one:

$$\pi_s^* = argmax_\pi U^\pi(s)$$

Starting from s, there are many possible recommendations (policies) for the next steps and this one is the best.

Note: the discounted utility definition with an infinite horizon makes the optimal policy independent from the starting state, even if the action sequence can be different.

This is because after a while two different starting states will reach a common future state and from that point on, they will follow the same path.

Optimal Policy

Using the principle of maximum expected utility ($action = argmax_a EU(a)$):

$$\pi_s^* = argmax_\pi U^\pi(s) = argmax_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

which means, choose the action that maximises the expected utility of the subsequent state.

The Bellman Equation

We defined the utility of a state as sum of (discounted) rewards onwards.

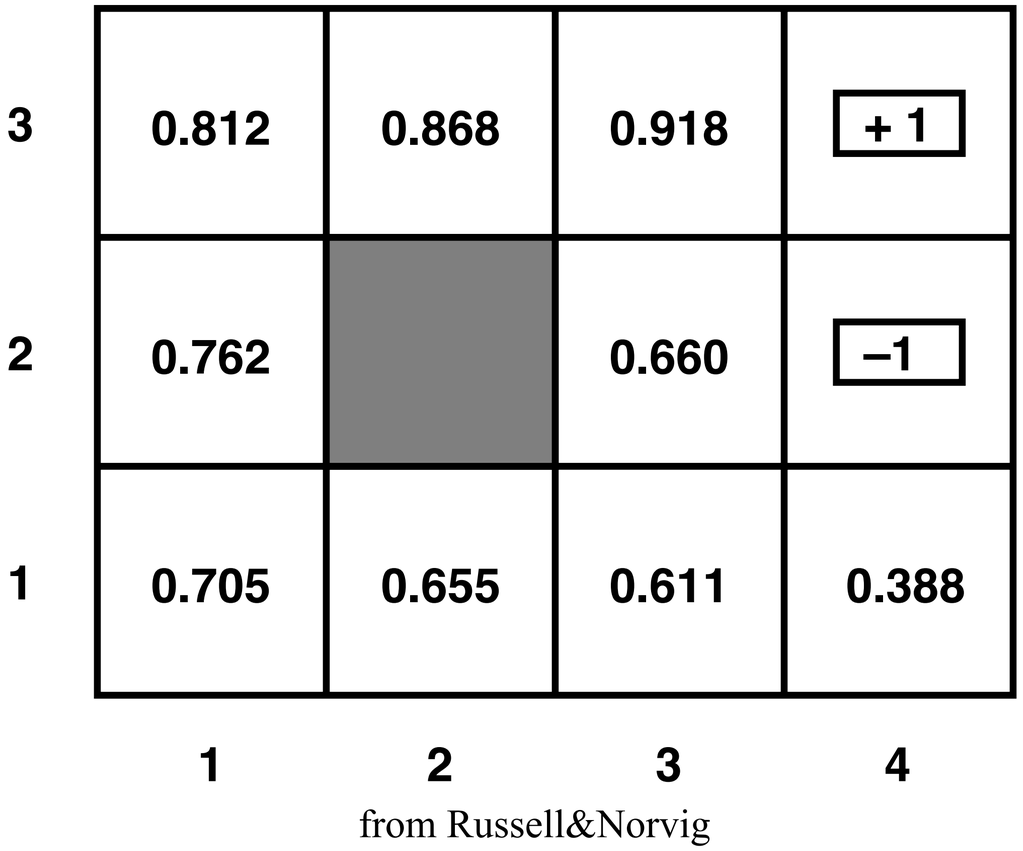
From this definition, we can state the relationship between the utility of a state and the utility of the neighbours:

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

This is the **Bellman Equation** (R. Bellman, 1957): the utility of a state s is the immediate reward R(s) for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action (the “max”).

Example

In the case of this 4x3 state space, $\gamma = 1$ (additive rewards case), R(s)=-0.04 for transitions in non-terminal states:



from Russell&Norvig

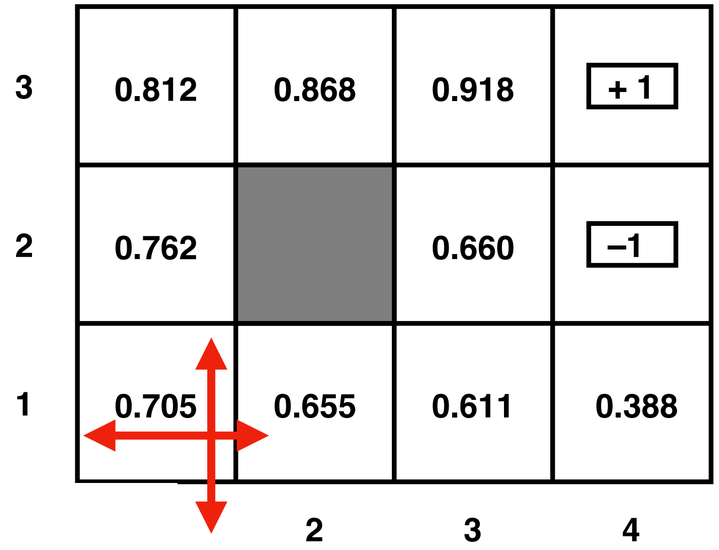
- Note that utilities are higher close to the +1 exit.

- The numbers are expectation values, thus also the 0.8/0.1 probabilities are taken into account.

Example

How to calculate the utility of one state with the BE:

$$U(1,1) = -0.04 + \gamma \max[\ 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1),$$
$$0.9U(1,1) + 0.1U(1,2),$$
$$0.9U(1,1) + 0.1U(2,1),$$
$$0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]$$



$$= -0.04 + \gamma \max\{\ 0.8 \cdot 0.762 + 0.1 \cdot 0.655 + 0.1 \cdot 0.705,$$
$$0.9 \cdot 0.705 + 0.1 \cdot 0.762,$$
$$0.9 \cdot 0.705 + 0.1 \cdot 0.655,$$
$$0.8 \cdot 0.655 + 0.1 \cdot 0.762 + 0.1 \cdot 0.705\}$$

$$= -0.04 + 1.0\,(\ 0.6096 + 0.0655 + 0.0705)$$

How to solve the Bellman Equation?

- The Bellman equation refers to a single state, but in general we have n states.
- This means that there are n Bellman equations to solve.
- The BE is non-linear (max is a non-linear operator) and finding an analytical solution is generically not possible.
- Like in other cases of PDEs or non-linear equations, an iterative approach works.

→ **Value Iteration Algorithm**

Value Iteration Algorithm

1. Start with arbitrary utility values.
2. Calculate $R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$
3. Update all the utilities for each state (all the squares in our example)
4. Repeat until convergence.

Value Iteration Algorithm

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum change in the utility of any state in an iteration

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return U

Convergence of the Algorithm (1)

Definition (Contraction): A function f is a contraction if $\|f(x) - f(y)\| \leq k\|x - y\|$
Example: f(x) = x/2

Theorem: Considering the Bellman equation, the operator

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

is a contraction and $\|BU - BU'\| \leq \gamma \|U - U'\|$ if we consider the norm $\| \cdot \| = \max(\cdot)$

If we consider U and U' as successive iterations of the value iteration algorithm, we can view $\|U - U'\|$ as the error in our estimate of U.
In other words, the error is reduced by a factor γ after each iteration.

Convergence of the Algorithm (2)

Remembering the result: $\sum_{t=0}^{\infty} \gamma^t R \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{\pm R_{max}}{1-\gamma}$ we can use the bound:

$$\|U_0 - U\| \leq \frac{2R_{max}}{1-\gamma}$$

Now, if we iterate N times for reaching an error ϵ, we have:

$$\gamma^N \frac{2R_{max}}{1-\gamma} \leq \epsilon \quad \Rightarrow \quad N = \lceil \frac{\log 2R_{max}/\epsilon(1-\gamma)}{\log(1/\gamma)} \rceil$$

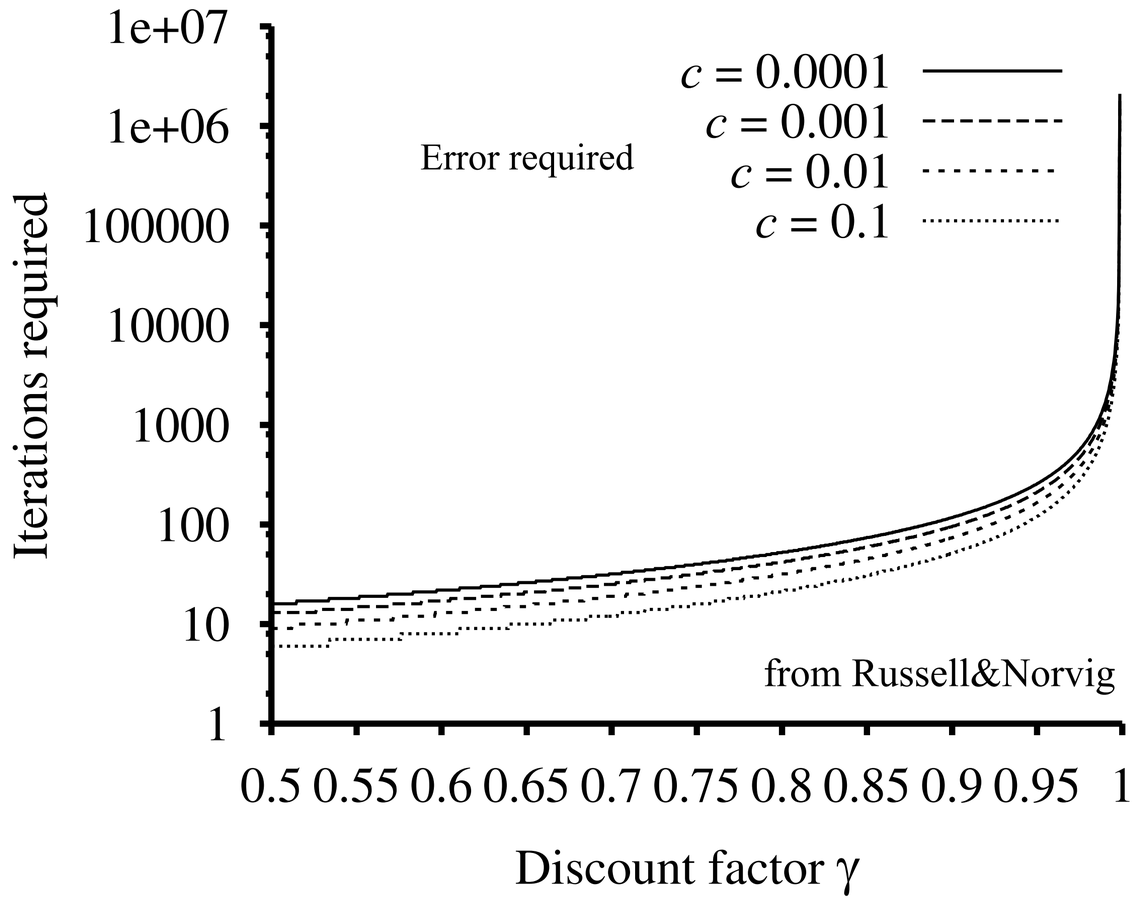
The last formula is an estimate of the iterations needed to reach a certain error.

Convergence of the Algorithm (3)

$$N = \lceil \frac{\log 2R_{max}/\epsilon(1-\gamma)}{\log(1/\gamma)} \rceil$$

Exponentially fast convergence: weak dependence from R_{max}/ϵ .

N grows very fast as $\gamma \to 1$.
A small γ can be used, but this means a small horizon for the agent (the "future"): long-term effects can be missed.



Summary

- Rational agents can be designed using probability theory and utility theory.
- Agents take decisions according to the axioms of unity theory and employ an utility function.
- Sequential problems in uncertain environments (probability!) can be solved through the determination of a policy.
- The Bellman equation provides a way to calculate an optimal policy. It is related to dynamical programming.
- A solution of the BE is given through the value iteration algorithm.