Chapter 1 : An introduction to Markov Decision Processes

Emmanuel Hyon[1]

[1]LIP6, Sorbonne Université, CNRS, France

TSP December 2020

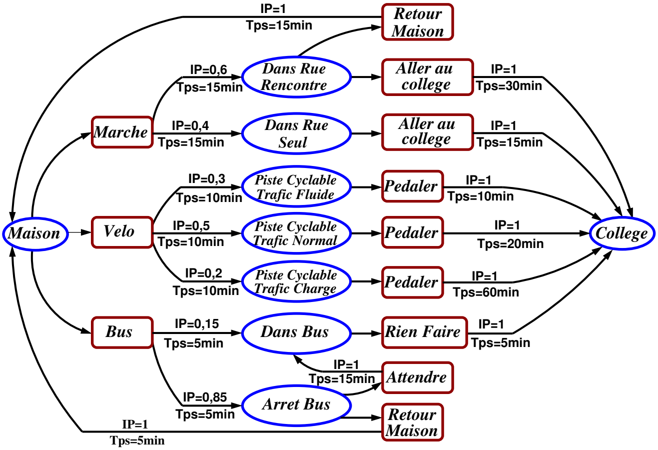
Outline

Outline

1. Introduction
   - Stochastic Optimization : some questions

The Schoolboy commute

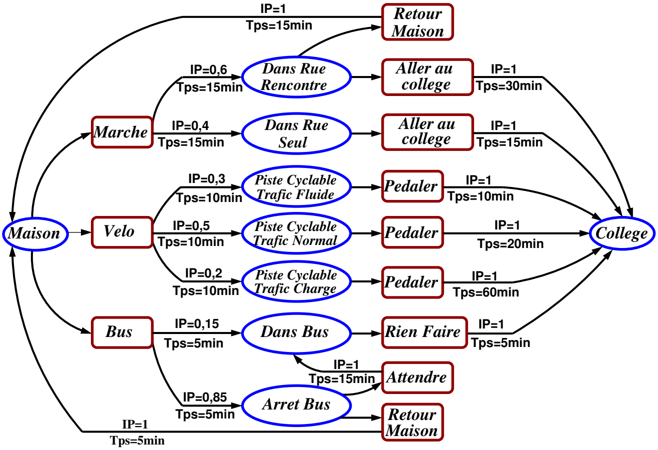
A schoolboy's commute from home to high school.



Question: do you know the real travel time ?

The Schoolboy commute

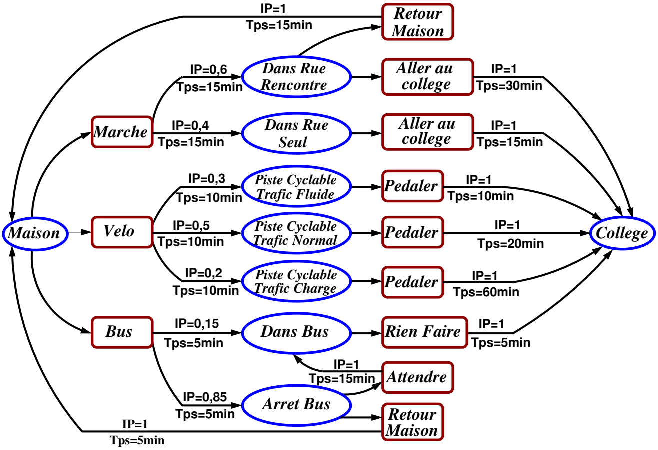
A schoolboy's commute from home to high school.



Answer: When the choice is walking two time possibilities

The Schoolboy commute

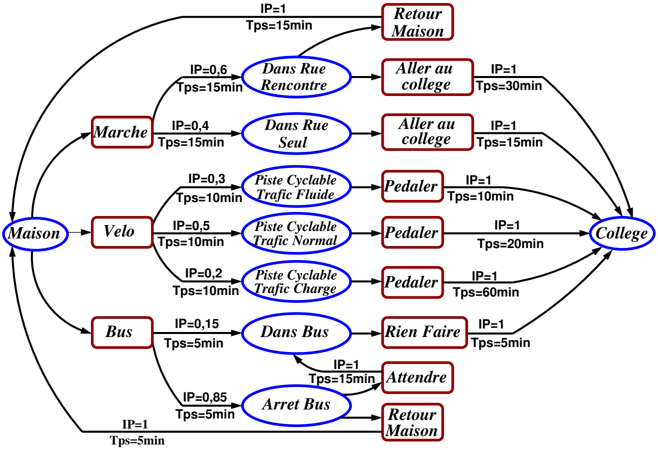
A schoolboy's commute from home to high school.



Requisite: the statistics

The Schoolboy commute

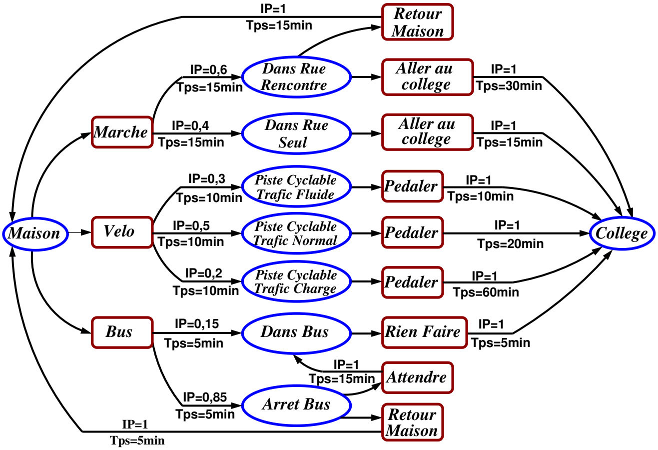
A schoolboy's commute from home to high school.



Question: is taking the bus the shortest ?

The Schoolboy commute

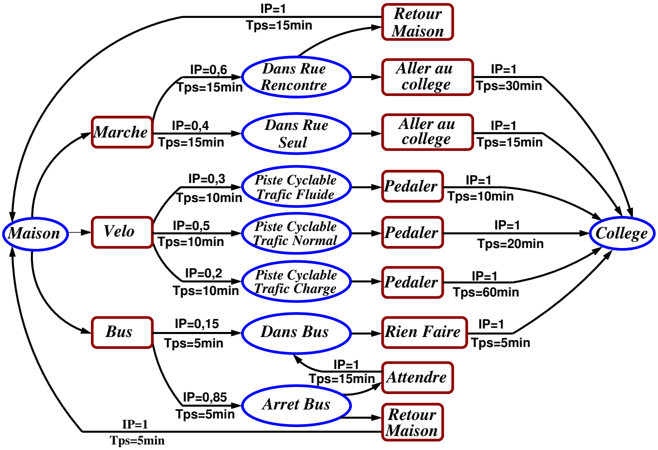
A schoolboy's commute from home to high school.



Answer: It depends on the wait

The Schoolboy commute

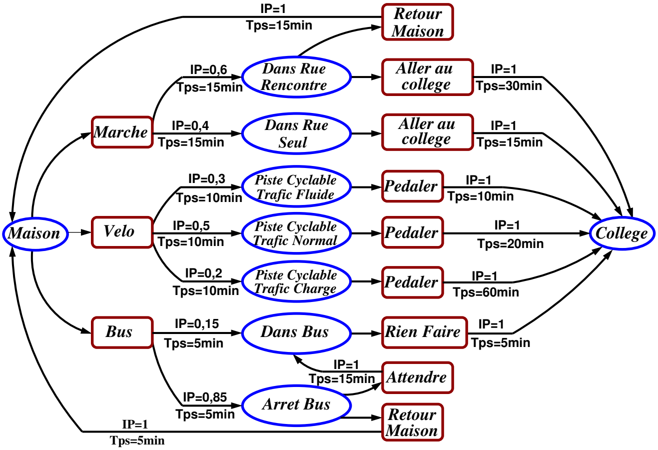
A schoolboy's commute from home to high school.



Requisite: Optimization should take into account the hazard

The Schoolboy commute

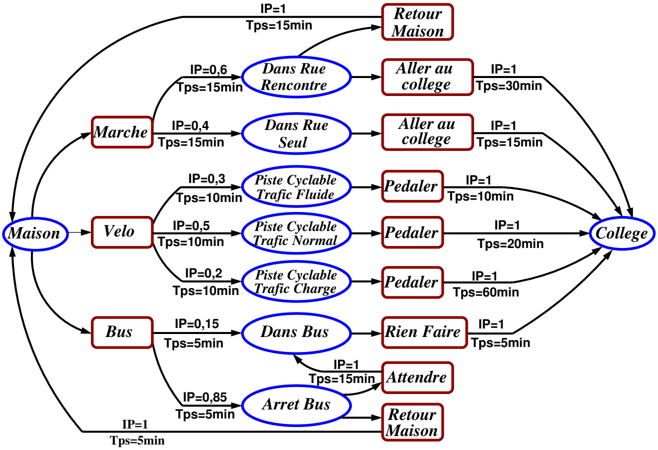
A schoolboy's commute from home to high school.



Question: Is the shortest path criteria relevant ?

The Schoolboy commute

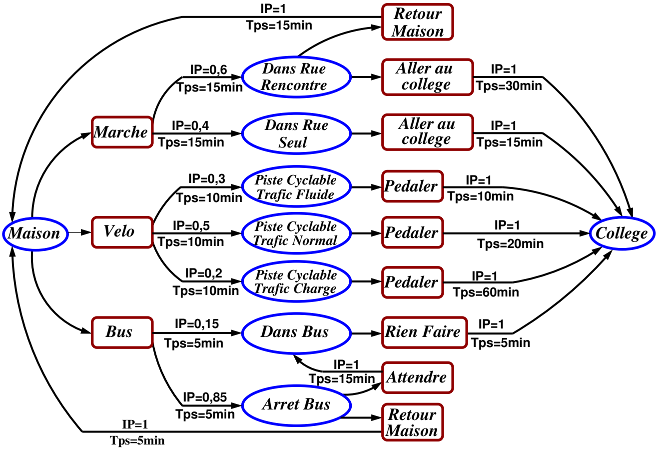
A schoolboy's commute from home to high school.



Answer: No you can experiment a different travel time

The Schoolboy commute

A schoolboy's commute from home to high school.



Requisite: New optimization criteria *Expected Shortest Path*

Controlled stochastic system (ctd)

This is a *Controlled stochastic system*.

Two axis for studies of controlled stochastic system

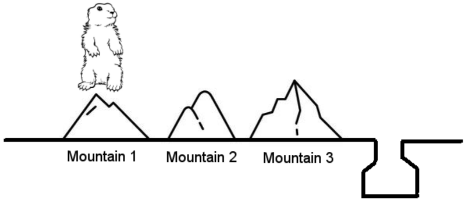
- Keeping some features of the system during its life,
  e.g Let the travel time smaller than a value
  $\Longrightarrow$ mainly in *automatic* field.
- Coupled with the evolution of the system there exist costs,
  e.g. you want to have the "shortest" travel time.
  $\Longrightarrow$ mainly in *Stochastic optimisation* field.

We are interested here by the stochastic optimisation aspects.

Outline

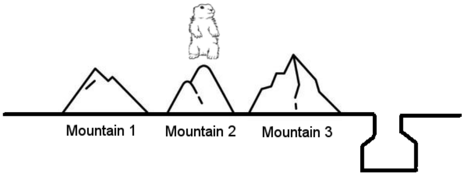
Introductory Example

You are on vacation and every morning you observe a groundhog and you record its position.



Introductory Example

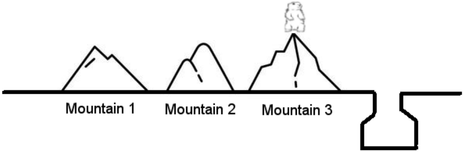
You are on vacation and every morning you observe a groundhog and you record its position.



The groundhog moves during the night.

Introductory Example

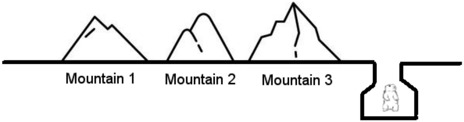
You are on vacation and every morning you observe a groundhog and you record its position.



The mountain on which is the groundhog changes.

Introductory Example

You are on vacation and every morning you observe a groundhog and you record its position.



The behaviour of the groundhog seems you random.

Introductory Example (ctd.) Stochastic Dynamical system

The sequence of the daily records is called a *path* or an *episode*. It can take several values :

- (M1, M2, M3, M3, M2, M3, M2, M1, M1,...)
- (M1, M2, M2, M1, M1, M2, M2, M2, M2,...)
- (H, H, H, H, H, H, H, H, H, H, H,......)
- (H, M3, M2, M3, H, H, H, M3, M2, M1,...)

Dynamical system

The system you observe is a *dynamical system*:

- the "*state*" of the system evolves during time.
  Let x_k be the position of the groundhog at day k.
- the next state can be determined with a *transition function*:

$$x_{k+1} = f_k(x_k, x_{k-1}, \dots, x_1) .$$

Introductory Example (ctd.) Stochastic dynamical system

Stochastic dynamical system

When the transition from a state at day k to a state at day $k+1$ follows a stochastic transition function it is a *stochastic system* :

$$x_{k+1} = f_k(x_k, x_{k-1}, \ldots, x_1, \varepsilon(\omega))$$

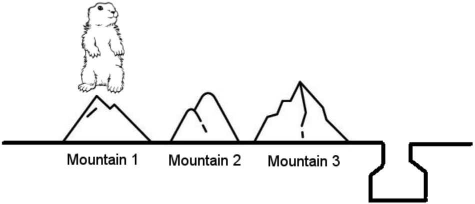
with $\varepsilon(\omega)$ is the hazard.

-The transition can depend on the states :
e.g. if k is a day corresponding of a winter day there few chances that the state will be different of H (home).

-The transition can depend on events that are not considered in the states:
e.g. the hazard is the presence of an eagle in the sky.

Example and Markov properties

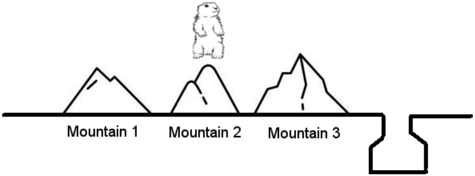
Consider again the example :



a) We assume that we know the transition probabilities (*i.e.* the stochastic transition function is known).

Example and Markov properties

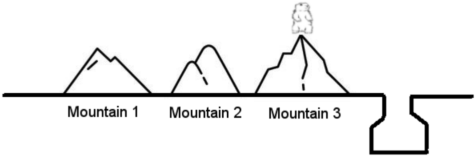
Consider again the example :



b) We assume that the transition probabilities only depend on the current state.

Example and Markov properties

Consider again the example :



c) We also assume that we know the initial place.

Markov Chain

We describe the behaviour of the groundhog by a Markov Chain that is a memoryless random process with Markov property.

- Let x_k be the position of the groundhog at day k.
- The stochastic transition function depends only on the hazard and on the current state :

$$x_{k+1} = f\left(x_k, \varepsilon_{k+1}(\omega)\right)$$

Definition

A (finite) *Markov Chain* is a tuple $\mathcal{X}$, $\mathcal{P}$:

- $\mathcal{X}$ is the state space : a (finite) set of states.
- $\mathcal{P}$ is the state to state transition probability (given by a transition matrix):
  $\mathcal{P}_{x,x'} = \mathbb{P}\left[x_{k+1} = x' \mid x_k = x\right]$.

Example (ctd.)

Assume transition probabilities on the Example 1 such that:

- If the groundhog stays on the first mountain, then it can
  - remain on the first mountain with probability 0.25;
  - leave to go on the second mountain with probability 0.5;
  - leave to go on the third mountain with probability 0.25.
- If the groundhog stays on the second mountain, then it can
  - remain on the second mountain with probability 0.2;
  - leave to go on the first mountain with probability 0.4;
  - leave to go on the third mountain with probability 0.4.
- If the groundhog stays on the third mountain, then it can
  - remain on the third mountain with probability 0.3;
  - leave to go on the first mountain with probability 0.4
  - leave to go on the second mountain with probability 0.3.

Example (ctd.): Model

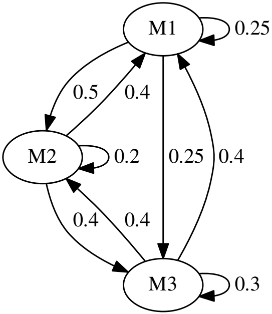
The *state space* is $\{M1, M2, M3\}$, where Mi is the mountain i.

The transition matrix is

$$\mathcal{P} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}$$

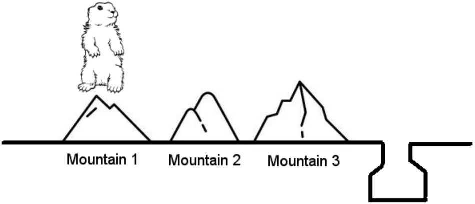
$$\begin{aligned} \mathbb{P}(X_{n+1} = j) &= \mathbb{P}(X_n = i) \times \mathcal{P} \\ &= \mathbb{P}(X_0 = i) \times \mathcal{P}^n \end{aligned}$$

The transition graph is



Example modified adding rewards

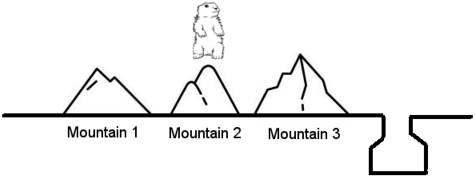
Consider again the second example and *add satisfactions*:



In state $M1$ we are *very happy*: we receive a satisfaction of 10,

Example modified adding rewards

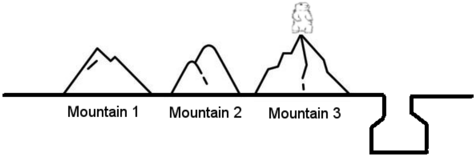
Consider again the second example and *add satisfactions*:



In state $M2$ we are *happy*: we receive a satisfaction of 1,

Example modified adding rewards

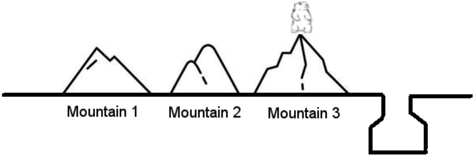
Consider again the second example and *add satisfactions*:



In state $M3$ we are *a little bit happy*: we receive 0.1,

Example modified adding rewards

Consider again the second example and *add satisfactions*:



Satisfaction is the reward of a state.

Markov Reward Process (MRP)

- Let r_k be the reward collected at day k.
- The reward is given by a function $r_k = \mathbb{E}\left(\mathcal{R}(x_{k+1}, x_k)\right)$

Definition

A *Markov Reward process* is a tuple $\mathcal{X}$, $\mathcal{P}$, $\mathcal{R}$:

- $\mathcal{X}$ is the state space : a set of states.
- $\mathcal{P}$ is the state to state transition probability: $\mathcal{P}_{x,x'} = \mathbb{P}\left[x_{k+1} = x' \mid x_k = x\right]$.
- $\mathcal{R}_k$ is a reward function $\mathcal{R}_k\left(x_{k+1}, x_k\right)$.

$$r_k(x) = \sum_{x' \in \mathcal{X}} \mathcal{P}_{x,x'} \times \mathcal{R}_k\left(x', x\right)$$

Collected rewards

We are interested by the return of a MRP: the sum of the rewards collected along a path :

Define $V_N(x)$ the sum of collected rewards from state x at day k :

$$V_N(x) = \sum_{k=0}^{k=N} r_k \, .$$

Collected rewards

We are interested by the return of a MRP: the sum of the rewards collected along a path :

Define $V_N(x)$ the sum of collected rewards from state x at day k :

$$V_N(x) = \sum_{k=0}^{k=N} r_k \, .$$

Path:	Return: ($V_1(M1)$ with $N = 6$)
(M1, M2, M3, M3, M3, M3)	$10 + 1 + 0.1 + 0.1 + 0.1 + 0.1 = 11.4$
(M1, M2, M3, M2, M1, M2)	$10 + 1 + 0.1 + 1 + 10 + 1 = 23.1$
(M1, M2, M1, M1, M2, M1)	$10 + 1 + 10 + 10 + 1 + 10 = 42$

Expected collected rewards

In stochastic problems it is more relevant to use the expectation of the sum of the rewards.

$$V_N(x) = \mathbb{E}\Big(\sum_{k=0}^{k=N} r_k(x_k)\Big)\,.$$

This takes into account the fact that a path occurs with a given probability.

Expected collected rewards

In stochastic problems it is more relevant to use the expectation of the sum of the rewards.

$$V_N(x) = \mathbb{E}\Big(\sum_{k=0}^{k=N} r_k(x_k)\Big)\,.$$

This takes into account the fact that a path occurs with a given probability.

Expected reward represents the average of the rewards among all the possible paths weighted by their probabilities.

$$V_N(x) = \sum_{k=0}^{k=N} \sum_{i\in\mathcal{X}} \mathbb{P}(X_k = i) r_k(i)\,.$$

Expected collected rewards

In stochastic problems it is more relevant to use the expectation of the sum of the rewards.

$$V_N(x) = \mathbb{E}\Big(\sum_{k=0}^{k=N} r_k(x_k)\Big)\,.$$

This takes into account the fact that a path occurs with a given probability.

Expected reward represents the average rewards if you repeat many times the experiment.

$$V_N(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{n} \Big(\sum_{k=0}^{k=N} r_k(x_k)\Big)$$

Expected collected rewards

We can also approximate $V_N(x)$ with the scenarios and $\hat{V}_N(x)$:

$$\hat{V}_N(x) = \sum \left((\sum_{k=0}^{k=N} \mathcal{R}_k\left(x_{k+1}, x_k\right)) \times \prod_{k=0}^{k=N} \mathbb{P}(X_{k+1} = j, X_k = i) \right) .$$

Path:	Return:
(M1, M2, M3, M3, M3, M3)	11.4 with probability 0.0054 $(0.5 \times 0.4 \times 0.3 \times 0.3 \times 0.3) = 0.0054$
(M1, M2, M3, M2, M1, M2)	23.1 with probability 0.012 $(0.5 \times 0.4 \times 0.3 \times 0.4 \times 0.5) = 0.012$
(M1, M2, M1, M1, M2, M1)	42 with probability 0.01 $(0.5 \times 0.4 \times 0.3 \times 0.3 \times 0.3) = 0.01$

Expected reward: $22, 66$
(on the paths) 22.18 (computed)

Objectives

But actually it exists more realistic objectives:

Discounted reward

Let $\theta \in [0, 1)$ be the discount factor. The *finite discounted reward* is:

$$V_N^\theta(x) = \mathbb{E}\left[\sum_{k=0}^{k=N} \theta^k r_k | x_0 = x\right]$$

Economic relevance: interest rate can be taken into account.
Practical relevance: future rewards have different weights according to θ.
Mathematical relevance: convergence of the sum.

Objectives

But actually it exists more realistic objectives:

Average reward

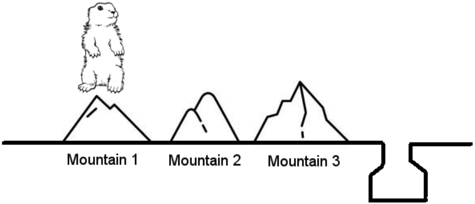
In an infinite horizon problem, the average reward is defined by:

$$\rho(x) = \lim_{N \to \infty} \frac{1}{N} \mathbb{E} \left[\sum_{k=0}^{k=N} r_k | x_0 = x \right]$$

Practical relevance: same cost in average for each day.
Mathematical relevance: convergence.

Example 2 with rewards and controls

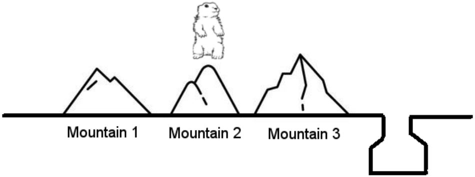
Consider *again* the first example and add control:



You can put food on one mountain each day.

Example 2 with rewards and controls

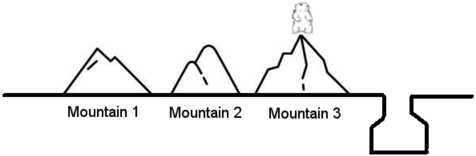
Consider *again* the first example and add control:



The food modifies the behaviour of the groundhog.

Example 2 with rewards and controls

Consider *again* the first example and add control:



But putting food on a mountain has a cost:
$M1$ costs 5, $M2$ costs 1 and $M3$ costs 0.5.

Example 1 with with rewards and controls :

a *dynamic control* problem

- You have a set of possible actions in each state.
- When you perform an action the stochastic transition function is modified (also the evolution of the system).
- When you perform an action the reward is modified.

We denote by $\mathcal{R}_k(x_k, a_k)$ the reward at day k when action a_k is triggered in state x_k.

The transition function depends on state, action and hazard:

$$x_{k+1} = f\left(x_k, a_k, \varepsilon_{k+1}\right).$$

The transition probability is now expressed with respect to the state and action:

$$\mathbb{P}\left(x_{k+1} = y | x_k = x, a_k = a\right) = \mathrm{p}(y|x, a)$$

Markov Decision Process

Definition

A *Markov Decision Process* is a tuple $\mathcal{X}$, $\mathcal{A}$, $(\mathcal{P})_a$, $\mathcal{R}$:

- $\mathcal{X}$ is the state space : a set of states.
- $\mathcal{A}$ is the action space : a set of actions.
- $(\mathcal{P})_a$ is a collection of transition probabilities :
  $$(\mathcal{P}_a)_{x,x'} = \mathbb{P}\left[x_{k+1} = x' \mid x_k = x, a_k = a\right].$$
- $\mathcal{R}_k$ is a reward function $\mathcal{R}_k\left(x_k, a_k\right)$,
  $$r_k(x,a) = \sum_{x' \in \mathcal{X}} \mathcal{R}_k\left(x', x, a\right) \times \mathrm{p}(x'|x,a).$$

The Markov Decision Process (MDP) is a tool to solve :

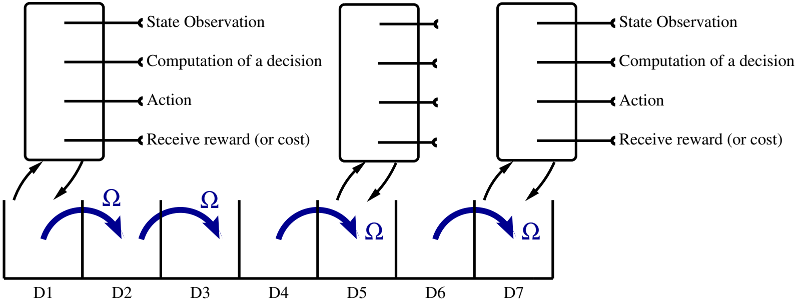
Dynamic control problems
aka *Optimal Control* problems
aka *Stochastic multistage* problems.

Dynamic control Applications fields

- Inventory Control
  1. *State* the stock
  2. *Action* the command
  3. *Hazard* the demand
- Marketing (or pricing)
  1. *State* The number of customers
  2. *Action* Prices of offers
  3. *Hazard* The number of people that become customers
- Networking (Admission Control)
  1. *State* packets in the network
  2. *Action* admission of a packet
  3. *Hazard* the number of arriving packets in the network

Multistage problem

A problem with: several stages, an optimisation problem, stochastic behaviour.



Example 1: Importance of the temporal modeling

Temporal behaviour

It is important to express with precision the temporal decomposition of events and their effects in a time slot.

Hence, it is important to detail carefully when rewards are received and when action is triggered.

Temporal decomposition of a day (method 1)

1. Breakfast
2. Decide where to place the food
3. Put the food on the chosen mountain
4. Go to sleep (*cost due to the action*)
5. Wake up
6. Observe the mountain and receive satisfaction (*cost due to nature*)

Example 1: Importance of the temporal modeling

Temporal behaviour

It is important to express with precision the temporal decomposition of events and their effects in a time slot.

Temporal decomposition of a day (method 2)

1. Wake up
2. Observe the mountain and receive satisfaction (*cost due to nature*)
3. Breakfast
4. Decide where to place the food
5. Put the food on the chosen mountain
6. Go to sleep (*cost due to the action*)

If we choose to let begin the slot at step 5 (wake), then satisfaction is not (directly) related with the action taken during the day.

Example 1 expressed with MDP (ctd.)

State space : $\{1, 2, 3\}$, where i means mountain Mi.

Action space : $\{0, 1, 2, 3\}$, where i means put the food on mountain Mi (and 0 means that nothing is done)

Transition probabilities:

$$\mathcal{P}_0 = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}, \mathcal{P}_1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix};$$

$$\mathcal{P}_2 = \begin{bmatrix} 0.25 & 0.55 & 0.2 \\ 0.2 & 0.4 & 0.4 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}, \mathcal{P}_3 = \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}.$$

Example 1 expressed with MDP (rewards)

Express the rewards now for state 1:

$$\begin{array}{lll} r(1|1,0) = 10\ , & r(2|1,0) = 1\ , & r(3|1,0) = 0.1 \\ r(1|1,1) = 5\ , & r(2|1,1) = -4\ , & r(3|1,1) = -4.9 \\ r(1|1,2) = 9\ , & r(2|1,2) = 0\ , & r(3|1,2) = -0.9 \\ r(1|1,3) = 9.5\ , & r(2|1,3) = 0.5\ , & r(3|1,3) = -0.4 \end{array}$$

The expected reward $r(x,a)$ is

$$r(x,a) = \sum_{y \in \mathcal{X}} R(y|x,a) \cdot \mathrm{p}(y|x,a)$$

and then we have :

$$\begin{aligned} r(1,0) &= 0.25 \cdot r(1|1,0) + 0.5 \cdot r(2|1,0) + 0.25 \cdot r(3|1,0) = 3.025 \\ r(1,1) &= 0.5 \cdot r(1|1,1) + 0.25 \cdot r(2|1,1) + 0.25 \cdot r(3|1,1) = 0.275 \\ r(1,2) &= 0.25 \cdot r(1|1,2) + 0.55 \cdot r(2|1,2) + 0.2 \cdot r(3|1,2) = 2.07 \\ r(1,3) &= 0.1 \cdot r(1|1,3) + 0.2 \cdot r(2|1,3) + 0.7 \cdot r(3|1,3) = 0.77 \end{aligned}$$

Example 1 expressed with MDP (rewards ctd.)

We can count the rewards for the other states in the same way, we get :

$$r(2,0) = 10 \cdot 0.4 + 1 \cdot 0.2 + 0.1 \cdot 0.4 = 4.24$$
$$r(2,1) = 5 \cdot 0.6 - 4 \cdot 0.2 - 4.9 \cdot 0.2 = 1.22$$
$$r(2,2) = 9 \cdot 0.2 + 0 \cdot 0.4 - 0.9 \cdot 0.4 = 1.44$$
$$r(2,3) = 9.5 \cdot 0.0 + 0.5 \cdot 0.1 - 0.4 \cdot 0.9 = -0.31$$

$$r(3,0) = 10 \cdot 0.4 + 1 \cdot 0.3 + 0.1 \cdot 0.3 = 4.33$$
$$r(3,1) = 5 \cdot 0.5 - 4 \cdot 0.4 - 4.9 \cdot 0.1 = 0.41$$
$$r(3,2) = 9 \cdot 0.3 + 0 \cdot 0.4 - 0.9 \cdot 0.3 = 2.43$$
$$r(3,3) = 9.5 \cdot 0 + 0.5 \cdot 0 - 0.4 \cdot 1 = -0.4$$

Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^{\theta} = \mathbb{E}\left[\sum_{k=0}^{k=7} \theta^{k} r(\mathbf{x_k}, \mathbf{a_k})\right] \tag{1}$$

How optimise our problem ?

Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^{\theta} = \mathbb{E}\left[\sum_{k=0}^{k=7} \theta^{k} r(\mathbf{x_k}, \mathbf{a_k})\right] \tag{1}$$

How optimise our problem ?

What can I do ?

Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^{\theta} = \mathbb{E}\left[\sum_{k=0}^{k=7} \theta^{k} r(\mathbf{x_k}, \mathbf{a_k})\right] \tag{1}$$

How optimise our problem ?

What can I do ?

$\Rightarrow$ Choose the action to perform each day.

Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^{\theta} = \mathbb{E}\left[\sum_{k=0}^{k=7} \theta^{k} r(\mathbf{x_k}, \mathbf{a_k})\right] \tag{1}$$

How optimise our problem ?

What can I do ?

$\Rightarrow$ Choose the action to perform each day.

What should I do ?

Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^{\theta} = \mathbb{E}\left[\sum_{k=0}^{k=7} \theta^{k} r(\mathbf{x_k}, \mathbf{a_k})\right] \tag{1}$$

How optimise our problem ?

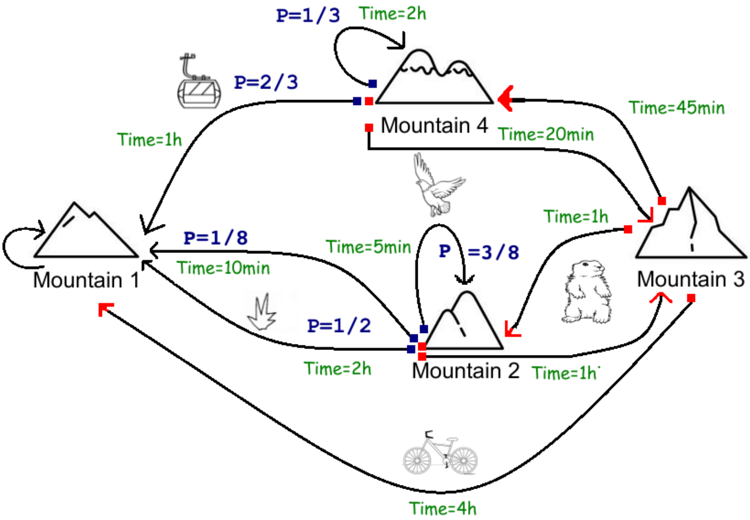
What can I do ?

$\Rightarrow$ Choose the action to perform each day.

What should I do ?

$\Rightarrow$ Determine the most profitable action (for the objective)

Example 3 : A stochastic shortest path problem



Decision tree

A dynamic control problem can be represented using a *decision tree*.

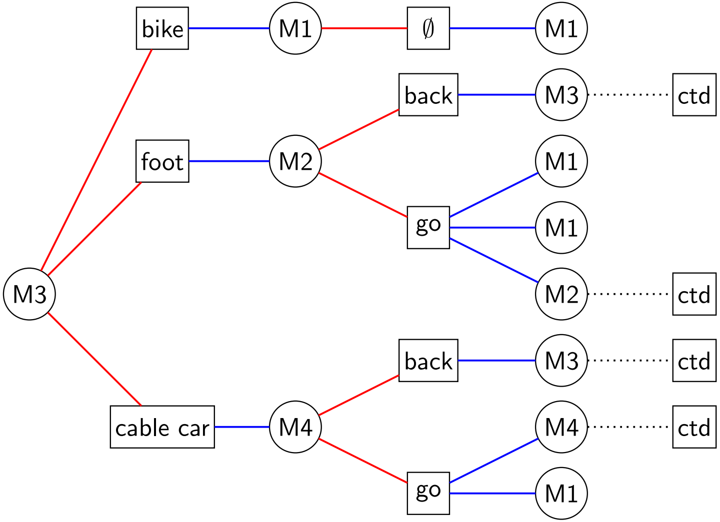
- A *decision tree* represents the set of possible paths including decisions and their possible consequences.
- A branch in the tree is a path in the system
- The costs of a trajectory in the system are the costs of a tree path.

Example 3

starting from $M3$

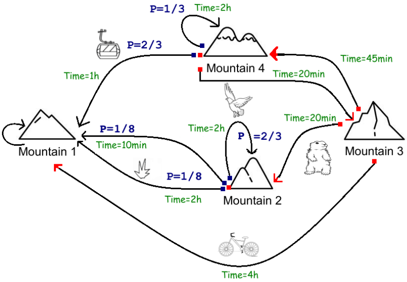
- red edges represent actions and their *triggered transitions*.
- blue edges represent system behaviour and their *natural transitions*.

Decision tree associated with Example 3



Computation of optimal action
Difficulties for computing optimal actions

The slighty modified SSP problem is now:



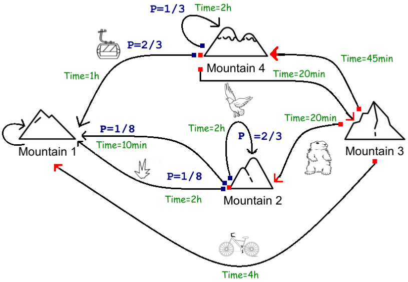
An action has a long term effect.

Ex. 1 Once you are in state $M1$ you remain in it.

Ex. 2 Once you played *foot* in $M3$ then no possibility to go in $M4$ or $M3$.

Computation of optimal action
Difficulties for computing optimal actions

The slighty modified SSP problem is now:

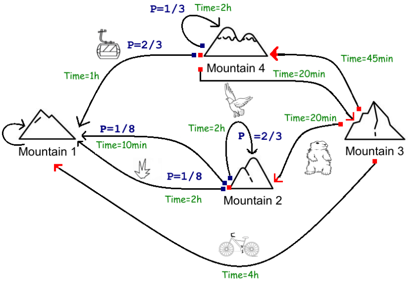


The benefit of an action also depends on the following rewards.

Ex. 3 Action *foot* in $M3$ gives a good immediate time (20min). But total times to reach M1 can be large ($30min$ with $\mathbb{P} = 1/8$, $2h20min$ with $\mathbb{P} = 1/8$, $\geq 2h$ with $\mathbb{P} = 2/3$)

Computation of optimal action
Difficulties for computing optimal actions

The slighty modified SSP problem is now:



The benefit of an action also depends on the following rewards.

Ex. 4 Action *cable*4 is bad for reaching $M4$ ($45min$).
But total times to reach M1 can be better ($1h45min$ with $\mathbb{P} = 2/3$).

Computation of optimal action (ctd.)

So the question is how evaluate branches ?

How evaluate sub branches ?

- Monte Carlo Simulations ?
  But how do we choose the action ?
  - Random choice ?
  - The action that gives the best immediate reward ?
  - We explore a part of the branch ?

The *Markov Decision Process*

can represent a dynamic control problem.

is a tool to solve the computation of optimal action using dynamic programming.

Outline

3. Formal elements of a Markov Decision Process
   - Mathematical model
   - Objective function
   - Decision Rule and Policy
   - Value function
   - Bellman Equation

Mathematical model of MDP

Definition

A *Markov Decision Process* is a tuple $\mathcal{X}$, $\mathcal{A}$, $(\mathcal{P})_a$, $\mathcal{R}$:

- $\mathcal{X}$ is the state space : a set of states.
- $\mathcal{A}$ is the action space : a set of actions.
- $(\mathcal{P}_k)_a$ is a collection of transition probabilities : $(\mathcal{P}_k)_a(x, x') = \mathbb{P}_k \left[x_{k+1} = x' \mid x_k = x, a_k = a\right]$.
- $\mathcal{R}_k$ is a reward function $r\left(x_k, a_k\right)$

When

1. transition probabilities $(\mathcal{P}_k)_a$ are the same for any k
2. the rewards are the same for any k

then the MDP is *stationary* (or homogeneous).

We only study here stationary MDP.

State Space

The state space is the (countable) set of states it represents the set of values in which our system evolves.
It can be

- the *physical state* of the system.
  e.g. the number of customers in a queue.
  e.g. the number of objects that are connected to me.
- the *physical state* of the system and an additional information.
  e.g. the mountain where the groundhog is *and* the presence of an eagle in the sky.
  e.g. the number of customers in a queue *and* an information about the transmission rate.

Powell [2] proposes a "definition" :
State represents the "physical" state and all additional information required for the decision.

Action Space

The Action Space represents the set of actions available at the controller in a given state.

However this set can be different depending on the state. Hence you can have two different sets $\mathcal{A}_x$ and $\mathcal{A}_{x'}$ for different states x and x'.

Usually you represent the action space by a set $\mathcal{A}$ that contains all possible actions ($\mathcal{A} = \cup_x \mathcal{A}_x$) considering that there are impossible actions in given states.

To model impossible action, two possibilities

1. You consider only the actions of $\mathcal{A}_x$ in x.
2. You consider all the actions in each state but you penalise impossible actions

Transition function

Your transition function "has" the Markov properties.

All the past is resumed in the present state.

$$x_{k+1} = f(x_k, a_k, \varepsilon_{k+1}) .$$

Thus

$$\mathbb{P}\left(x_{k+1} = y \,\middle|\, x_k = x, a_k = a, x_{k-1}, a_{k-1}, \ldots, x_0, a_0\right) = \mathbb{P}\left(x_{k+1} = y \,\middle|\, x_k = x, a_k = a\right)$$

also

$$\mathbb{P}\left(x_{k+1} = y \,\middle|\, x_k = x, a_k = a\right) = \mathrm{p}(y|x, a) .$$

Rewards

We consider only rewards which depend on the state and the action : $r(x, a)$.

Reward can be bounded or unbounded.

Computer Representation

Matrices

How: Storing a set of transition probabilities matrices in an array.

When: a complete description of the system is storable and available.

Advantages: Efficient especially coupled with sparse matrices

Disadvantages Large MDP can not be represented.

Hash tables

How: Storing transition probabilities in an hash table. Key is the tuple (x_k, x_{k+1}, a_k).

When: Simple system in which a complete description is available.

Advantages: Fast to program. Efficient.

Disadvantages Large and complex MDP can not be represented.

List and function

How: Probability transitions are described by a function and not all the states are studied.

When: A compact form is desired.

Advantages Adapted for very large systems.

Disadvantages Require many computations and recalculations.

Objectives

The different objectives that can be solved are :

Finite Horizon

The gain on a finite horizon N is the expected sum of the reward on a path of length N:

$$V_N = \mathbb{E}\left(\sum_{i=0}^{N-1} r(x_i, a_i)\right)$$

where $r(x_i, a_i)$ is the reward at step i.

Objectives II

The different objectives that can be solved are :

discounted finite Horizon

The gain on a discounted finite horizon N is the expected sum of the weighted rewards on a path of length N:

$$V_N^\theta = \mathbb{E}\left(\sum_{i=0}^{N-1} \theta^i r(x_i, a_i)\right)$$

where $r(x_i, a_i)$ is the reward at step i,
where $\theta \in [0, 1)$ is the discount factor.

Objectives III

The different objectives that can be solved are :

discounted infinite Horizon

The gain on a discounted infinite horizon is the expected sum of the rewards on an infinite path :

$$V^\theta = \mathbb{E}\left(\sum_{i=0}^{\infty} \theta^i r(x_i, a_i)\right)$$

where $r(x_i, a_i)$ is the reward at step i,
where $\theta \in [0, 1)$ is the discount factor.

Objectives IV

The different objectives that can be solved are :

total reward

The gain on a total reward is the expected sum of the rewards on an infinite path :

$$V = \mathbb{E}\left(\sum_{i=0}^{\infty} r(x_i, a_i)\right)$$

where $r(x_i, a_i)$ is the reward at step i.

The sum of the total rewards may not converge.
This kind of models is often applied with a set of absorbing states.

Objectives V

The different objectives that can be solved are :

average

The gain on an average cost model is the limit of the Cesaro mean of the expected sum of the rewards on a path :

$$\rho = \lim_{N \to \infty} \frac{1}{N} \mathbb{E} \left(\sum_{i=0}^{N-1} r(x_i, a_i) \right)$$

where $r(x_i, a_i)$ is the reward at step i.

This kind of objective has a gain which depends on the intrinsic Markov Chain.

Decision rule

Definition

We define a decision rule as the mapping that allows to select an action to perform.

Its classification depends on the information set that is used.

We denote by π_k the decision rule that is applied at step k.
A decision π_k can be :

History dependent	Depends on the complete past (states, actions)
State dependent (or *Markov*)	Depends on the current state

Thus a Decision rule is a mapping π

from the history to an action $\pi : \mathcal{H} \mapsto A$

from the state to an action $\pi : S \mapsto A$

Decision rule (ctd.)

Another classification is made according to the manner the decision is returned

Deterministic	The rule defines a single action
Random	The rule defines a probability on an action.

Thus a decision rule can define

- a *single action* $\pi : \mathcal{H} \mapsto a$
- a *probability* with which the action is chosen $\pi : \mathcal{H} \mapsto \mathbb{P}(a)$

Decision rule (ctd.)

Thus we have four classes of decision rules.

1. Random History dependent,
2. Deterministic History dependent,
3. Random Markov,
4. Deterministic Markov,

Random history dependent is the more general.

Deterministic Markov is the more specific and is included in Random History dependent.

Policy

Usually the terms *policy* and *decision rule* are considered as synonym.
Let us make a distinction between them for the sake of clarity

Definition

A Policy Π is a sequence of decision rules

$$\Pi = (\pi_1, \pi_2, \ldots, \pi_n, \ldots).$$

Definition

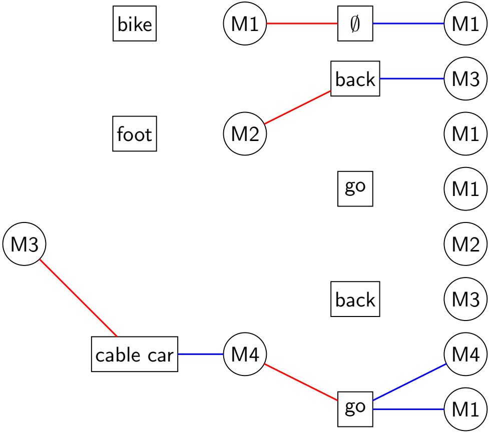
A decision rule can be *stationary* : the same rule is applied in each slot

$$\Pi = (\pi, \pi, \ldots, \pi).$$

Back to example 3

Decision tree associated with a decision rule

We assume that $\pi(M_1) = \emptyset$, $\pi(M2) = back$, $\pi(M3) =$ cable car, $\pi(M4) = go$,
then the decision tree becomes



Value function

How to evaluate the reward of a policy ?

For this we define value function.

Definition

A value function is defined for:

- a fixed policy Π,
- a fixed initial state $x \in \mathcal{X}$

and returns the total gain (for the criteria considered) collected following the policy Π from the initial state.

For a fixed Π we have a function (or a vector) such that

$$V^{\Pi} : \mathcal{X} \mapsto \mathbb{R}.$$

Value function (ctd.)

There exists a value function expression for each of the objective.

Value Function for finite horizon problem

Let N be the horizon. The finite horizon value function is denoted by $V_N^\Pi(x)$ with

$$V_N^\Pi(x) = \mathbb{E}^\Pi\left(\sum_{k=0}^{N-1} r\left(x_k^\Pi, \pi_k(x_k^\Pi)\right) | x_0 = x\right).$$

Value Function for infinite discounted cost

The discounted cost value function is V_θ^Π such that $\forall x \in \mathcal{X}$:

$$V_\theta^\Pi(x) = \mathbb{E}^\Pi\left(\sum_{k=0}^{\infty} \theta^k r\left(x_k^\Pi, \pi(x_k^\Pi)\right) | x_0 = x\right).$$

Back to Example 1 and the groundhog

Markov deterministic stationary policy

I will define the policy such that the decision rule is

$$\pi(1) = 0$$
$$\pi(2) = 1$$
$$\pi(3) = 2$$

Back to Example 1 and the groundhog

Value function

The value function should be computed for each of the state $M1$, $M2$, $M3$.
For Example 1 and a finite horizon objective of three days I have to compute:

$$V_3^\pi(1)$$
$$V_3^\pi(2)$$
$$V_3^\pi(3)$$

Back to Example 1 and the groundhog

Computation

I can compute the value function for this policy (the MDP is now a Markov Reward Process) either

- *formally* (with *Power Method*)
- by *Monte Carlo Simulation*
- by *temporal learning*.

Back to Example 1 Compute the value function

Building the matrix

Transition matrix

$$\mathcal{P}_0 = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}, \mathcal{P}_1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix};$$

$$\mathcal{P}_2 = \begin{bmatrix} 0.25 & 0.55 & 0.2 \\ 0.2 & 0.4 & 0.4 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}, \mathcal{P}_3 = \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}$$

and then, transition matrix associated with policy Π is

$$\mathcal{P}_\Pi = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}$$

Back to Example 1 Compute the value function (ctd.)

Building rewards

Rewards

$$\begin{aligned} r(1,0) &= 3.025\ , & r(2,0) &= 4.24\ , & r(3,0) &= 4.33 \\ r(1,1) &= 0.225\ , & r(2,1) &= 1.22\ , & r(3,1) &= 0.41 \\ r(1,2) &= 2.07\ , & r(2,2) &= 1.44\ , & r(3,2) &= 2.43 \\ r(1,3) &= 0.77\ , & r(2,3) &= -0.31\ , & r(3,3) &= -0.4 \end{aligned}$$

and then

$$\mathbf{r}_\Pi = \begin{bmatrix} r(1,0) \\ r(2,1) \\ r(3,2) \end{bmatrix} = \begin{bmatrix} 3.025 \\ 1.22 \\ 2.43 \end{bmatrix}$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^\pi = \mathcal{P}_\Pi \mathbf{r}_\Pi + \mathcal{P}_\Pi^2 \mathbf{r}_\Pi + \mathcal{P}_\Pi^3 \mathbf{r}_\Pi$$

and then

$$\mathbf{V}^\pi = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{bmatrix} \begin{bmatrix} 3.025 \\ 1.22 \\ 2.43 \end{bmatrix} + \mathcal{P}_\Pi^2 \mathbf{r}_\Pi + \mathcal{P}_\Pi^3 \mathbf{r}_\Pi,$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^{\pi} = \mathcal{P}_{\Pi}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{2}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi}$$

and then

$$\mathbf{V}^{\pi} = \begin{bmatrix} 1.9737 \\ 2.545 \\ 2.1245 \end{bmatrix} + \begin{bmatrix} 0.4375 & 0.325 & 0.2375 \\ 0.3300 & 0.420 & 0.2500 \\ 0.4050 & 0.350 & 0.2450 \end{bmatrix} \begin{bmatrix} 3.025 \\ 1.22 \\ 2.43 \end{bmatrix} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi},$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^{\pi} = \mathcal{P}_{\Pi}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{2}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi}$$

and then

$$\mathbf{V}^{\pi} = \begin{bmatrix} 1.9737 \\ 2.545 \\ 2.1245 \end{bmatrix} + \begin{bmatrix} 2.2971 \\ 2.1181 \\ 2.2475 \end{bmatrix} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi},$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^{\pi} = \mathcal{P}_{\Pi}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{2}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi}$$

and then

$$\mathbf{V}^{\pi} = \begin{bmatrix} 4.2708 \\ 4.6631 \\ 4.3720 \end{bmatrix} + \begin{bmatrix} 0.37563 & 0.37875 & 0.24563 \\ 0.40950 & 0.34900 & 0.24150 \\ 0.38475 & 0.37050 & 0.24475 \end{bmatrix} \begin{bmatrix} 3.025 \\ 1.22 \\ 2.43 \end{bmatrix}.$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^{\pi} = \mathcal{P}_{\Pi}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{2}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi}$$

and then

I get

$$V_3^{\pi}(M1) = 6.4660$$
$$V_3^{\pi}(M2) = 6.9145$$
$$V_3^{\pi}(M3) = 6.5826$$

Back to Example 1 and the groundhog

Compute the value function (ctd.)

Power method

$$\mathbf{V}^{\pi} = \mathcal{P}_{\Pi}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{2}\mathbf{r}_{\Pi} + \mathcal{P}_{\Pi}^{3}\mathbf{r}_{\Pi}$$

and then

I get

$$V_3^{\pi}(M1) = 6.4660$$
$$V_3^{\pi}(M2) = 6.9145$$
$$V_3^{\pi}(M3) = 6.5826$$

But to find the best policy, I have to compare all different reward processes !!!

Optimal policy

Partial Order on value function

We denote by $\mathcal{V}$ the space of functions from $\mathcal{X} \to \mathbb{R}$ (related with the vector space $\mathbb{R}^{|\mathcal{X}|}$)
The space $\mathcal{V}$ has a partial order :

$$\forall\ U, V \in \mathcal{V} \quad U \leq V \Leftrightarrow\ U(x) \leq V(x)\,,\ \forall x \in \mathcal{X}\,.$$

Optimal policy

The goal of a Markov Decision Process is to characterise, as well as to search and compute (if they exist): the optimal policy $\Pi^* \in \Pi^{HR}$ (Π^{HR} is the set of history dependent random policies) such that

$$\forall\ \Pi \in \Pi^{HR}\ ,\ \forall x \in \mathcal{X} : V^{\Pi}(x) \leq V^{\Pi^*}(x)\,,$$

(equivalently said $\Pi^* \in \arg\max_{\Pi \in \Pi^{HR}} V^{\Pi}$) and the optimal value V^{Π^*}.

Equivalence of policy

If we should consider all the history dependent random policies, then the computation will suffer from curse of dimensionality.

Fortunately, they are results that claim we can search in the *Markov Deterministic policy* set.

Theorem

Let $\Pi \in \Pi^{HR}$ be an history dependent random policy. Then for each initial state x it exists a Markov Random policy Π' such that

- $V_N^{\Pi}(x) = V_N^{\Pi'}(x)$ *Finite horizon*
- $V_\theta^{\Pi}(x) = V_\theta^{\Pi'}(x)$ *Discounted cost*
- $V^{\Pi}(x) = V^{\Pi'}(x)$ *Total reward*
- $\rho^{\Pi}(x) = \rho^{\Pi'}(x)$ *average reward*

For each objective, the two value functions have the same value.

Equivalence of policy

If we should consider all the history dependent random policies, then the computation will suffer from curse of dimensionality.

Fortunately, they are results that claim we can search in the *Markov Deterministic policy* set.

Then the passing of the equivalence from Markov Random policy to Markov Deterministic policy is a case by case study (for each objective).

Proofs and validity conditions can be found in [1].

Dynamic Programming and framework of proof

Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

Dynamic Programming and framework of proof

Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

Then to obtain the shortest path from A to B:

1. you consider all the successors of A
2. for each successor (say y)
   you compute the length of the path from A to B by adding:
   -the length from A to y
   -the shortest path from y to B.
3. you take the successor y such that the path from A to B is the smallest.

Dynamic Programming and framework of proof

Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

Framework of proof

1. Express the *Bellman Equation*.
   This means decompose your cost in two parts:
   the immediate cost due to action and the future cost due to the effects of your action.
2. Prove its validity:
   This means to show that using the Bellman Equation will lead to the optimal policy.
3. Solve the equation numerically:
   This means apply well known algorithms.

Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$
\begin{aligned}
V_N^*(x) &= \max_{a_1,\dots,a_N \in \mathcal{A}^N} \mathbb{E}\Big(\sum_{k=0}^{N-1} r(x_k, a_k) | x_0 = x \Big) , \\
&= \max_{(a)^N \in \mathcal{A}^N} \left(\mathbb{E}\Big(r(x,a) \big| \, x_0 = x \Big) + \mathbb{E}\Big(\sum_{k=1}^{N-1} r(x_k, a_k) \Big| \, x_0 = x \Big) \right), \\
&= \max_{(a)^N \in \mathcal{A}^N} \Bigg(r(x,a) \\
&\qquad\qquad + \mathbb{E}\Big(\sum_{y \in \mathcal{X}} \sum_{k=1}^{N-1} r(x_k, a_k) \mathbb{1}_{\{x_1 = y \,, a_0 = a\}} \Big| \, x_0 = x \Big) \Bigg) ,
\end{aligned}
$$

Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$
\begin{aligned}
V_N^*(x) = \max_{a,\ldots,a \in \mathcal{A}^N} \Bigg(& r(x,a) \\
& + \sum_{y \in \mathcal{X}} \mathbb{E}\left(\sum_{k=1}^{N-1} r(x_k, a_k) \middle| x_1 = y, a_0 = 0, x_0 = x \right) \times \\
& \mathbb{P}\left(x_1 = y \middle| x_0 = x, a_0 = a \right) \Bigg) \\
= \max_{(a)^N \in \mathcal{A}^N} \Bigg(& r(x,a) + \sum_{y \in \mathcal{X}} \mathrm{p}(y|x,a) \mathbb{E}\left(\sum_{k=1}^{N-1} r(x_k, a_k) \middle| x_1 = y \right) \Bigg)
\end{aligned}
$$

Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$
\begin{aligned}
V_N^*(x) &= \max_{a\in\mathcal{A}}\Bigg(r(x,a) + \sum_{y\in\mathcal{X}} \mathrm{p}(y|x,a)\times \\
&\qquad \max_{a_1,\dots,a_N\in\mathcal{A}^{N-1}}\left(\mathbb{E}\Big(\sum_{k=0}^{N-2} r(x_k,a_k)\Big|\, x_0 = y\Big)\right)\Bigg) \\
&= \max_{a\in\mathcal{A}}\left(r(x,a) + \sum_{y\in\mathcal{X}} \mathrm{p}(y|x,a) V_{N-1}^*(y)\right).
\end{aligned}
$$

Bellman Equation (ctd.)

The Bellman equation in finite horizon case is :

$$V_N^*(x) = \max_{a \in \mathcal{A}} \Big(r(x,a) + \sum_{y \in \mathcal{X}} V_{N-1}^*(y) \mathrm{p}(y|x,a) \Big) . \qquad (2)$$

Sketch of proof of the method validity:
To prove the validity of the use of BE we have several things to show.

1. Show that the value function V_N that results from Eq. (2) is the optimal (i.e. V_N^*).
2. Show that the value function V_{N-1} used is the optimal (i.e. V_{N-1}^*)

Bellman Equation (proof of Bellman function)

Proof of Bellman Function in other fields

The checking of the validity of the Bellman Equation is not restricted to MDP !
e.g. a graph course that presents the Bellman-Ford algorithm will check the conditions of convergence.

Many works have already done the job

- In a large number of MDP models the validity are proved and the assumptions that should be verified are detailed.
- The Puterman's book presents some of them with detailed proofs.
- When you work with MDP you have to use the assumptions related to your model.
  e.g. bounded rewards or unbounded rewards models require different sets of assumptions

Q value function

The Bellman Equation in infinite discounted case is

$$V_\theta^*(x) = \max_{a \in \mathcal{A}} \Big(r(x,a) + \theta \sum_{y \in \mathcal{X}} V_\theta^*(y) \mathrm{p}(y|x,a) \Big) .$$

Q value function

From Bellman Equation we define:

Definition (Q value function)

For a fixed policy Π with value function V_θ^Π, one associates the function Q^Π such that

$$\forall\ x \,\forall\ a\ ,\ Q^\Pi(x,a) = r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|x,a) V_\theta^\Pi(y)$$

Q value function

From Bellman Equation we define:

Definition (Q value function)

For a fixed policy Π with value function V_θ^Π, one associates the function Q^Π such that

$$\forall\, x \,\forall\, a\ ,\ Q^\Pi(x,a) = r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|x,a) V_\theta^\Pi(y)$$

It is the expected value

1. starting from x,
2. applying a,
3. and then following policy Π.

Q value function

From Bellman Equation we define:

Definition (Q value function)

For a fixed policy Π with value function V_θ^Π, one associates the function Q^Π such that

$$\forall\ x \forall\ a\ ,\ Q^\Pi(x,a) = r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|x,a) V_\theta^\Pi(y)$$

Property

The optimal value function V^* satisfies

$$\forall\ x\ ,\ V^*(x) = \max_a Q^\Pi(x,a)\,.$$

Outline

4. Numerical solving
    - Objectives
    - Value Iteration
    - Policy Iteration

Three main uses of Bellman Equation (BE)

- Prediction:
  Computing the value of a policy
- Planification:
  Computing both
  - Optimal policy
  - Optimal value function

Resolution algorithms

It exists three main ways to compute both optimal value function and optimal policy (both are computed together).

1. Value Iteration (VI)
2. Policy Iteration (PI)
3. Linear Programming

Note that the computation of the decision rule is computed off-line and then applied by the controller.

You can compute the decision rule *on-line* but in this case you are in the field of *on-line MDP*.

Value Iteration Principle

Value iteration is based on main principles of *dynamic programming*

Finite Horizon case

Equivalent of the DP deterministic cases.

Infinite Horizon case

First notice that *Bellman Equation* is a *fixed point Equation*:

$$V^*(y) = \max_{a \in \mathcal{A}} \left(r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|x,a) V^*(y) \right),$$

$$V^* = TV^*$$

It “suffices” to apply fix point equation methods and to check their validity (Bellman operator T should be contracting).

Value Iteration for the finite horizon

Backward Algorithm: resolution of finite horizon criterion

Initialise $V_0 = r_N$
for $k = N - 1$ to 0 do
  for $x \in \mathcal{X}$ do
$$V_k^*(s) = \max_{a \in \mathcal{A}} \Big[r_k(x, a) + \sum_{y \in \mathcal{X}} \mathrm{p}_k(y \mid x, a) V_{k_1}^*(y) \Big]$$
$$\pi_k(x) \in \arg\max_{a \in \mathcal{A}} \Big[r_k(x, a) + \sum_{y \in \mathcal{X}} \mathrm{p}_k(y \mid x, a) V_{k-1}^*(y) \Big]$$
  end for
end for
return the two sequences V_k^* and π_k^*

Exercice Computation of the optimal policy (ctd)

Recalling the datas

Transition matrices

$$\mathcal{P}_0 = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}, \mathcal{P}_1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix};$$

$$\mathcal{P}_2 = \begin{bmatrix} 0.25 & 0.55 & 0.2 \\ 0.2 & 0.4 & 0.4 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}, \mathcal{P}_3 = \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}$$

Rewards

$$\begin{array}{lll} r(1,0) = 3.025 \,, & r(2,0) = 4.24 \,, & r(3,0) = 4.33 \\ r(1,1) = 0.225 \,, & r(2,1) = 1.22 \,, & r(3,1) = 0.41 \\ r(1,2) = 2.07 \,, & r(2,2) = 1.44 \,, & r(3,2) = 2.43 \\ r(1,3) = 0.77 \,, & r(2,3) = -0.31 \,, & r(3,3) = -0.4 \end{array}$$

Exercice Computation of the optimal policy (ctd)

Apply value function algorithm for Example 1 with horizon $H = 3$

We define V_0 as the terminal value function (with horizon 0).
We have

$$\begin{aligned} V_0(1) &= 0 \ , \ \pi_0^*(1) = 0 \\ V_0(2) &= 0 \ , \ \pi_0^*(2) = 0 \\ V_0(3) &= 0 \ , \ \pi_0^*(3) = 0 \, . \end{aligned}$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 1

$$\begin{aligned} V_1(1)^* &= \max_{a\in\{0,\dots,3\}} \Big(r(1,a) + \sum_{y\in\{1,\dots,3\}} \mathrm{p}(y|1,a) V_0^*(y) \Big) \\ &= \max\Big(3.025, 0.225, 2.07, 0.77\Big) \\ &= 3.025\,. \end{aligned}$$

and

$$\pi_1^*(1) = \arg\max_{a\in\{0,\dots,3\}} \Big(r(1,a) \Big) = 0;$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 1

$$
\begin{aligned}
V_1(2)^* &= \max_{a\in\{0,\dots,3\}} \Big(r(2,a) + \sum_{y\in\{1,\dots,3\}} \mathrm{p}(y|2,a) V_0^*(y) \Big) \\
&= \max\Big(4.24, 1.22, 1.44, -0.31\Big) \\
&= 4.24\,.
\end{aligned}
$$

and

$$
\pi_1^*(2) = \arg\max_{a\in\{0,\dots,3\}} \Big(r(2,a) \Big) = 0;
$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 1

$$
\begin{aligned}
V_1(3)^* &= \max_{a\in\{0,\dots,3\}} \Big(r(3,a) + \sum_{y\in\{1,\dots,3\}} \mathrm{p}(y|3,a) V_0^*(y) \Big) \\
&= \max\Big(4.33, 0.41, 2.43, -0.4\Big) \\
&= 4.33\,.
\end{aligned}
$$

and

$$
\pi_1^*(3) = \arg\max_{a\in\{0,\dots,3\}} \Big(r(3,a) \Big) = 0;
$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$

We define V_0 as the terminal value function (with horizon 0). We have

$$
\begin{aligned}
&V_0^*(1) = 0\,,\ \pi_0^*(1) = 0\,, \quad V_1^*(1) = 3.025\,,\ \pi_1^*(1) = 0\\
&V_0^*(2) = 0\,,\ \pi_0^*(2) = 0\,, \quad V_1^*(2) = 4.24\,,\ \ \pi_1^*(2) = 0\\
&V_0^*(3) = 0\,,\ \pi_0^*(3) = 0\,, \quad V_1^*(3) = 4.33\,,\ \ \pi_1^*(3) = 0\,.
\end{aligned}
$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

$$
\begin{aligned}
V_2(1)^* &= \max_{a\in\{0,\dots,3\}} \Big(r(1,a) + \sum_{y\in\{1,\dots,3\}} \mathrm{p}(y|1,a) V_1^*(y) \Big) \\
&= \max\Big(r(1,0) + \mathrm{p}(1|1,0) V_1^*(1) + \mathrm{p}(2|1,0) V_1^*(2) \\
&\qquad + \mathrm{p}(3|1,0) V_1^*(3), \dots \Big) \\
&= \max\Big(3.025 + 3.025 \cdot 0.25 + 4.24 \cdot 0.5 + 4.33 \cdot 0.25, \dots \Big) \\
&= \max\Big(6.98375, r(1,1) + \mathrm{p}(1|1,1) V_1^*(1) \\
&\qquad + \mathrm{p}(2|1,1) V_1^*(2) + \mathrm{p}(3|1,1) V_1^*(3), \dots \Big)
\end{aligned}
$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

$$
\begin{aligned}
V_2(1)^* &= \max_{a\in\{0,\dots,3\}} \Big(r(1,a) + \sum_{y\in\{1,\dots,3\}} \mathrm{p}(y|1,a) V_1^*(y) \Big) \\
&= \max\Big(6.98375, \\
&\qquad 0.225 + 3.025 \cdot 0.5 + 4.24 \cdot 0.25 + 4.33 \cdot 0.25, \dots \Big) \\
&= \max\Big(6.98375, 3.88, r(1,2) + \mathrm{p}(1|1,2) V_1^*(1) \\
&\qquad + \mathrm{p}(2|1,2) V_1^*(2) + \mathrm{p}(3|1,2) V_1^*(3), \dots \Big) \\
&= \max\Big(6.98375, 3.88, \\
&\qquad 2.07 + 3.025 \cdot 0.2 + 4.24 \cdot 0.4 + 4.33 \cdot 0.4, \dots \Big)
\end{aligned}
$$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

$$
\begin{aligned}
V_2(1)^* &= \max_{a \in \{0,\dots,3\}} \left(r(1,a) + \sum_{y \in \{1,\dots,3\}} \mathrm{p}(y|1,a) V_1^*(y) \right) \\
&= \max\Big(6.98375, 3.88, 6.103, \\
&\qquad r(1,3) + \mathrm{p}(1|1,3) V_1^*(1) + \mathrm{p}(2|1,2) V_1^*(2) + \mathrm{p}(3|1,2) V_1^*(3) \Big) \\
&= \max\Big(6.98375, 3.88, 6.103, \\
&\qquad 0.77 + 3.025 \cdot 0.1 + 4.24 \cdot 0.2 + 4.33 \cdot 0.7 \Big) \\
&= \max\big(6.98375, 3.88, 6.103, 4.67925 \big) = 6.98375
\end{aligned}
$$

and $\pi_2^*(1) = 0$.

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

$$V_2(2)^* = \max_{a \in \{0,\ldots,3\}} \left(r(2,a) + \sum_{y \in \{1,\ldots,3\}} \mathrm{p}(y|2,a) V_1^*(y) \right)$$
$$= 7.895$$

and $\pi_2^*(2) = 0$

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

$$
\begin{aligned}
V_2(3)^* &= \max_{a\in\{0,\ldots,3\}} \left(r(3,a) + \sum_{y\in\{1,\ldots,3\}} \mathrm{p}(y|3,a) V_1^*(y) \right) \\
&= 8.12
\end{aligned}
$$

and $\pi_2^*(3) = 0$.

Exercice Computation of the optimal policy (ctd)

Value function algorithm for Example 1 with horizon $H = 3$ step 2

We have

$$
\begin{aligned}
V_0^*(1) &= 0\,,\ \pi_0^*(1) = 0\,, \quad V_1^*(1) = 3.025\,,\ \pi_1^*(1) = 0\\
V_0^*(2) &= 0\,,\ \pi_0^*(2) = 0\,, \quad V_1^*(2) = 4.24\,,\ \pi_1^*(2) = 0\\
V_0^*(3) &= 0\,,\ \pi_0^*(3) = 0\,, \quad V_1^*(3) = 4.33\,,\ \pi_1^*(3) = 0\,.
\end{aligned}
$$

We have

$$
\begin{aligned}
V_2^*(1) &= 6.983\,,\ \pi_2^*(1) = 0\,, \quad V_1^*(1) = 10.75\,,\ \pi_1^*(1) = 0\\
V_2^*(2) &= 7.895\,,\ \pi_2^*(2) = 0\,, \quad V_1^*(2) = 11.74\,,\ \pi_1^*(2) = 0\\
V_2^*(3) &= 8.12\,,\ \pi_2^*(3) = 0\,, \quad V_1^*(3) = 11.95\,,\ \pi_1^*(3) = 0\,.
\end{aligned}
$$

Value Iteration for discounted infinite horizon

value iteration algorithm: θ-discounted cost criterion
Initialise V_0
$k \leftarrow 0$
repeat
for $x \in \mathcal{X}$ do
$V_{k+1}(x) = \max_{a \in \mathcal{A}} \Big(r(x,a) + \theta \sum_{y \in S} \mathbb{P}(y \mid x,a) V_k(y) \Big)$
$k \leftarrow k+1$
until $\lVert V_{k+1} - V_k \rVert \leq \epsilon$
for $x \in \mathcal{X}$ do
$\pi(x) \in \arg\max_{a \in \mathcal{A}} \{ r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y \mid x,a) V_{k+1}(y) \}$
return V_{k+1}, π

Value Iteration for total reward policy

value iteration algorithm: for total reward cost criterion
Initialise V_0
$k \leftarrow 0$
repeat
for $x \in \mathcal{X}$ do
$V_{k+1}(x) = \max_{a \in \mathcal{A}} \Big(r(x,a) + \sum_{y \in S} \mathbb{P}(y \mid x, a) V_k(y) \Big)$
$k \leftarrow k+1$
until $\lVert V_{k+1} - V_k \rVert \leq \epsilon$
for $x \in \mathcal{X}$ do
$\pi(x) \in \arg\max_{a \in \mathcal{A}} \{ r(x,a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y \mid x, a) V_{k+1}(y) \}$
return V_{k+1}, π

Exercice

Iterate value function algorithm for Example 3

Policy Iteration principles

Two steps :

1. *Policy improvement*
2. *Policy evaluation*

Policy improvement

Aims at improving the policy

Policy evaluation

Aims at evaluating the policy

Policy Iteration for infinite horizon discounted cost

Policy iteration algorithm: θ-discounted cost criterion

Initialise $\pi_0 \in \Pi^{MD}$
$k \leftarrow 0$
repeat
Computation phase): solve
$$V_k(x) = r(x, \pi_k(x)) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|(x, \pi_k)) V_k(y), \ \forall x \in \mathcal{X}$$
improvement phase): improve
for $x \in \mathcal{X}$ do
$$\pi_{k+1}(x) \in \arg\max_{a \in \mathcal{A}} \{r(x, a) + \theta \sum_{y \in \mathcal{X}} \mathrm{p}(y|(s, a)) V_k(y)\}$$
$k \leftarrow k + 1$
until $\pi_k = \pi_{k+1}$
return V_k, π_{k+1}

Outline

Bibliography

References that present the main concepts of MDP

F. Garcia.
Processus Décisionnels de Markov
Chapter 1 in *Processus Décisionnels de Markov en Intelligence Artificielle*
Hermes, 2008.

W.B. Powell
Introduction to Markov Decision Processes
Chapter 3 in Approximate Dynamic Programming, Solving the Curses of Dimensionality
Wiley, 2011.

E.L. Porteus
Foundations of stochastic inventory theory
Stanford University Press, 2002.

Bibliography

Complete book (with detailed proofs)

M. Puterman.
Markov Decision Processes Discrete Stochastic Dynamic Programming.
Wiley, 2005.

D. Bertsekas
Dynamic Programming and Optimal Control, Vol. II.
Athena Scientific, (4th edition) 2012.