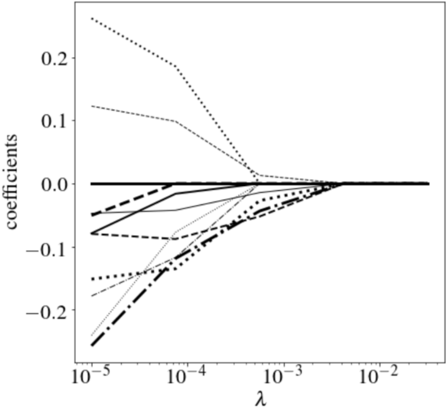
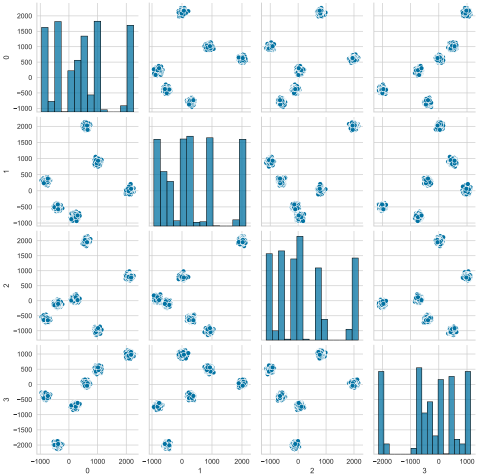
Fondamentaux théoriques du machine learning

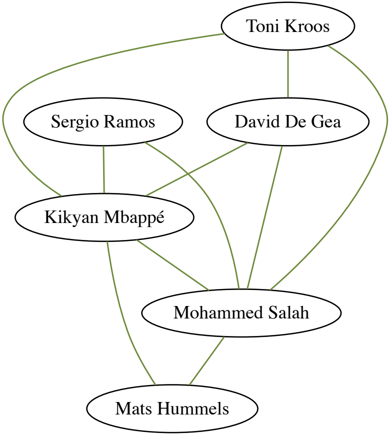


Overview of lecture 8

Some data are numerical.



But some are non-numerical or not only numerical.



threshold 10

Working with non-numerical data

In ML, it is always necessary to have a **metric**, or a way to compare data points. When the data are non-numerical, we have already seen that **one-hot encoding** can be used to convert them to numerical data. Then, a proper metric can be defined in $\mathbb{R}^d$ for some $d \in \mathbb{N}$.

Working with non-numerical data

However in some cases it might be not possible or relevant to one-hot encode the data. A proper **distance** might be not possible to define.
We recall that a distance is a mapping $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ that is symmetric, separates the values and respects the triangular inequality.

Working with non-numerical data

When it is not possible to define a distance, we can introduce a **dissimilarity** (or equivalently, a **similarity**).

Similarities

- When working with distances, two points that "look the same" should be separated by a **small distance**.
- When working with a similarity, two points that "look the same" should have a **high similarity**.

The similarity is a more general notion than that of ditance. It does not need to be symmetrical.

Similarities

The similarity can take values in $\mathbb{R}$. Sometimes, the relevant similarity is the **binary similarity**. $S_{ij} = 0$ or $S_{ij} = 1$. It can be interpreted as a representation of the **compatibility** between samples i and j.
In a graph, the relationship of adjacency can be seen as a binary similarity.

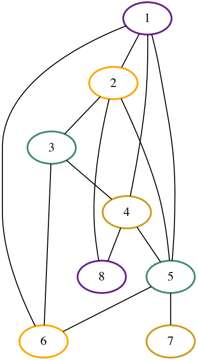
Graph problems

Some famous graph problems can model an optimization problem defined by a **compatibility** relationship.

Coloring problem

We have a map with different countries and need to assign a color to each country, so that two countries that have a common border are filled with a different color. We assume that we would like to use a small number of colors (the smaller, the better).
Exercice 1 : How would you formalize this problem with a graph ?

The coloring problem

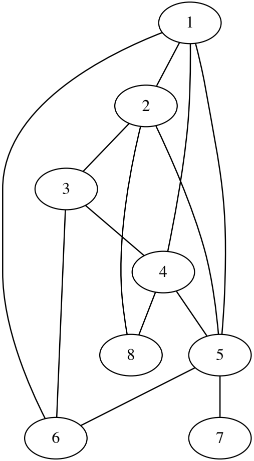


We want to find the smallest number of **fully disconnected subgraphs** in a graph.

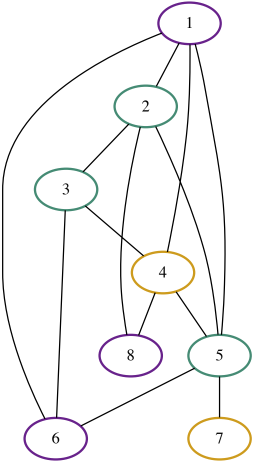
The coloring problem

We want to find the smallest number of **fully disconnected subgraph** in a graph.
Each subgraph will be associated with a color.

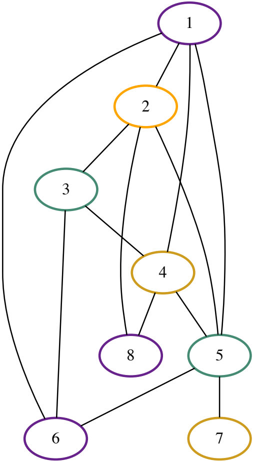
Coloring



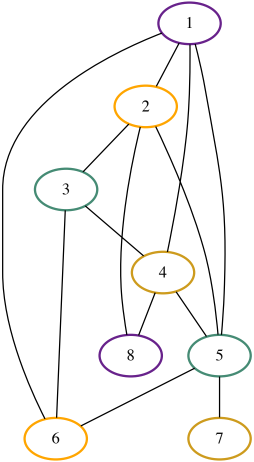
Is this a coloring ?



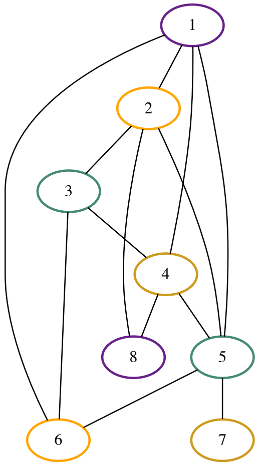
Is this a coloring ?



Is this a coloring ? yes



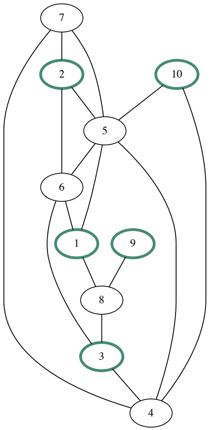
Could we have used only 3 colors ?



Independent set

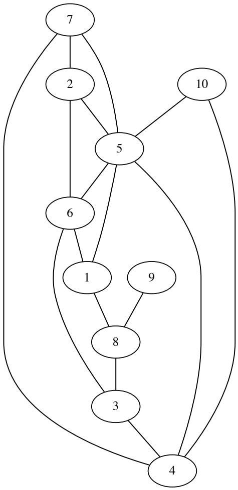
We have a dataset of persons. Some people cannot work with each other. We want to build to largest possible team of people.
Exercice 2 : How could we formalize this problem with a graph ?

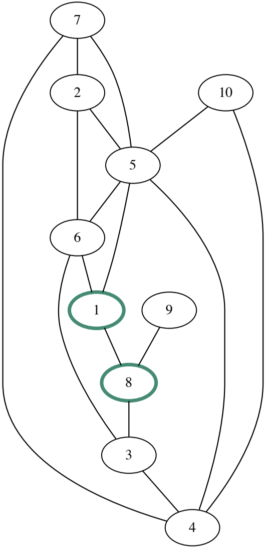
Independent Set

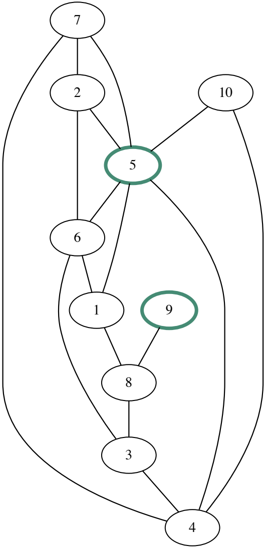


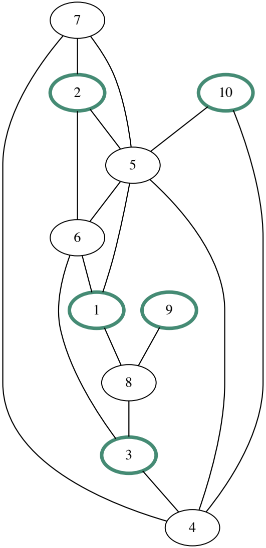
Assuming that an edge represents the fact that two persons cannot work with each other, we want to find **the largest disconnected subgraph**.

Independent set : what is a trivial independent set ?

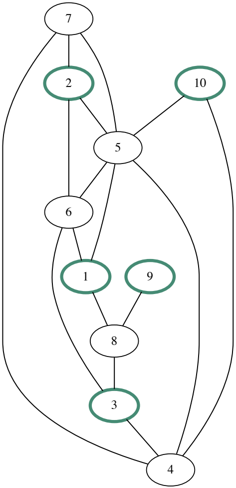



7
2
10
5
6
1
9
8
3
4

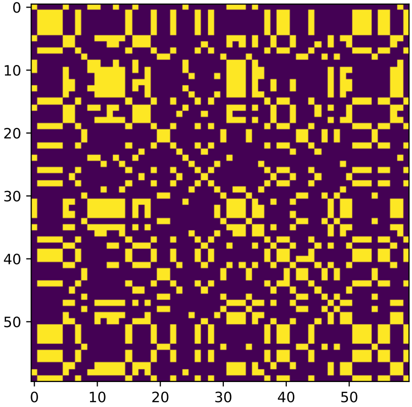

7
2
10
5
6
1
9
8
3
4


7
2
10
5
6
1
9
8
3
4

Maximal vs maximum independent set

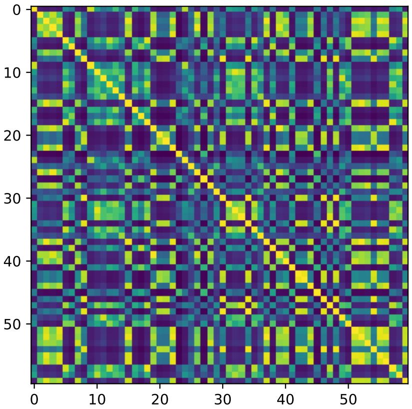


Adjacency matrix



Similarity matrix

The similarity can also be a general value in $\mathbb{R}$ (not necessary binary).



Similarities

- A similarity S is not always symmetrical.
- Indeed, in a **directed graph**, having a directed edge between i and j does not mean that we have an edge between j and i.

Similarities

Differences between similarities and distances :

- A similarity S is not always symmetrical.
- Indeed, in a **directed graph**, having a directed edge between i and j does not mean that we have an edge between j and i.
- $S_{ij} = 0$ does not mean that $i = j$, it is rather the contrary.

Similarities

- A similarity is a more general notion than a distance. Given a similarity between two points, we can deduce a similarity.
- For instance this way, if d_{ij} is the distance betwwen i and j :

$$S_{ij} = \exp(-d_{ij}) \tag{1}$$

Spectral Clustering

- A clustering method that works with similarities
- It performs a low dimensional embedding of the similarity matrix, followed by a Kmeans

Spectral clustering

Some drawbacks of the method :

- Need to provide the number of clusters.
- Not adapted to a large number of clusters.
- kmeans step : so depends on a random initialization.

Heuristic

We would like a critetion in order to justify the number of clusters used.

- We previously used the inertia as a "clustering score", e.g. in the case of the kmeans. We also experimented with the Silhouette score.
- However, in Spectral clustering, there is not necessary a notion of distance, and there is no prototype (not a vector quantization method).

Normalized cut : a measure of the quality of a clustering

- The **cut of a cluster** is the number of outside connections (connections with other clusters).
- The **degree** of a node is its number of adjacent edges
- The **degree of a cluster** is the sum of the degrees of its nodes.
- The **normalized cut** of a clustering is :

$$NCut(\mathcal{C}) = \sum_{k=1}^{K} \frac{Cut(C_k, V \backslash C_k)}{d_{C_k}} \tag{2}$$

Exercice 3 : Why is the normalization relevant in order to define a meaningful scoring ?

Example

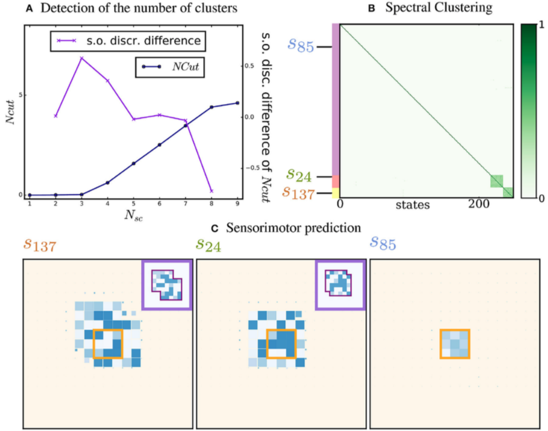


Figure – In a), the elbow method is used to choose the number of clusters.[Le Hir et al., 2018]

Clustering

There are several other approaches to the clustering problem, such as hierarchical clustering.

`https://scikit-learn.org/stable/modules/clustering.html`

Example

- If $d >> n$ and we want to learn a linear model $x \mapsto \langle \theta, x \rangle$, we have seen that this raises statistical issues (high variance, overfitting).
- However, if we know in advance that θ only has $s < d$ non-zero coordinates (sparse θ), we can reformulate to an easier problem.
- But most of the time this is not the case, s is not known, so we need to test several subsets of non-zero coordinates.

Example

We could write the following regularized optimization problem

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \left(||Y - X\theta|| + \lambda ||\theta||_0 \right) \tag{3}$$

- $y \in \mathbb{R}^n$ (labels)
- $X \in \mathbb{R}^{n,d}$ (design matrix)
- $||\theta||_0$: number of non-zero components of θ

Example

We could write the following regularized optimization problem

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^d}{\arg\min} \Big(||Y - X\theta|| + \lambda ||\theta||_0 \Big) \tag{4}$$

- $y \in \mathbb{R}^n$ (labels)
- $X \in \mathbb{R}^{n,d}$ (design matrix)
- $||\theta||_0$: number of non-zero components of θ

However,

- optimization issue (not convex)
- computationally prohibitive to test all subsets of $[1, d]$.
  Exercice 4 : How many subsets does $[1, d]$ contain ?

Lasso

Le Lasso replaces $||\theta||_0$ by $||\theta||_1$.

$$||\theta_1|| = \sum_{i=1}^{d} |\theta_i| \tag{5}$$

Lasso estimator :

$$\tilde{\theta}_\lambda \in \underset{\theta \in \mathbb{R}^d}{\arg\min} \{||Y - X\theta||^2 + \lambda||\theta||_1\} \tag{6}$$

For technically involved reasons, the optimization with the lasso leads to sparser solutions in some situations.

Lasso

Lasso estimator :

$$\tilde{\theta}_\lambda \in \underset{\theta \in \mathbb{R}^d}{\arg\min} \{||Y - X\theta||^2 + \lambda ||\theta||_1\} \quad (7)$$

For technically involved reasons, the optimization with the lasso leads to sparser solutions. Frequently used optimization algorithm :

- coordinate descent (algorithm used in scikit)
- Fista
- LARS

https://en.wikipedia.org/wiki/Coordinate_descent

Example in 1D

Exercice 5 : What is the solution to the following optimization problem ?

$$\min_{\theta} F(\theta) = \frac{1}{2}(y - \theta)^2 + \lambda|\theta| \tag{8}$$

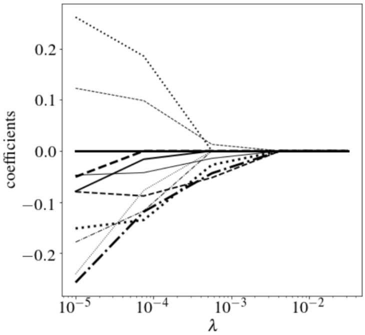


Figure – Regularization path with a Lasso optimization of a problem with $d = 12$. Each line represents the evolution of a θ_i when λ increases. Image from [Azencott, 2022].

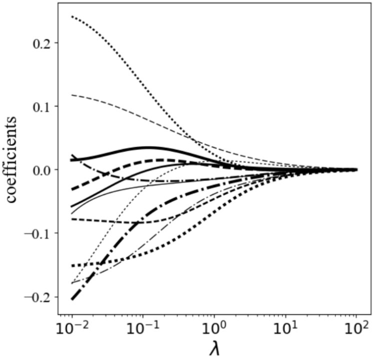


Figure – Regularization path with a Ridge optimization of a problem with $d = 12$. Each line represents the evolution of a θ_i when λ increases. Image from [Azencott, 2022].

Elastic net

Combination of $L1$ and $L2$ regularization.
Elastic-net estimator :

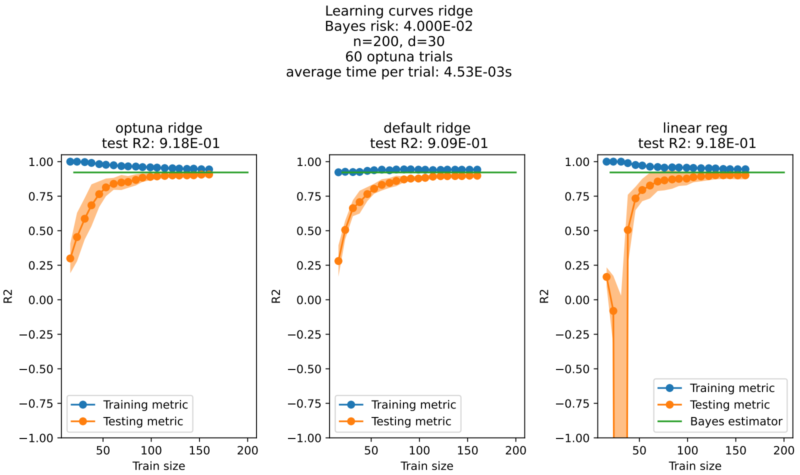
$$\tilde{\theta}_\lambda \in \arg\min_{\theta \in \mathbb{R}^d} \{||Y - X\theta||^2 + \lambda_1 ||\theta||_1 + \lambda_2 ||\theta||_2\} \tag{9}$$

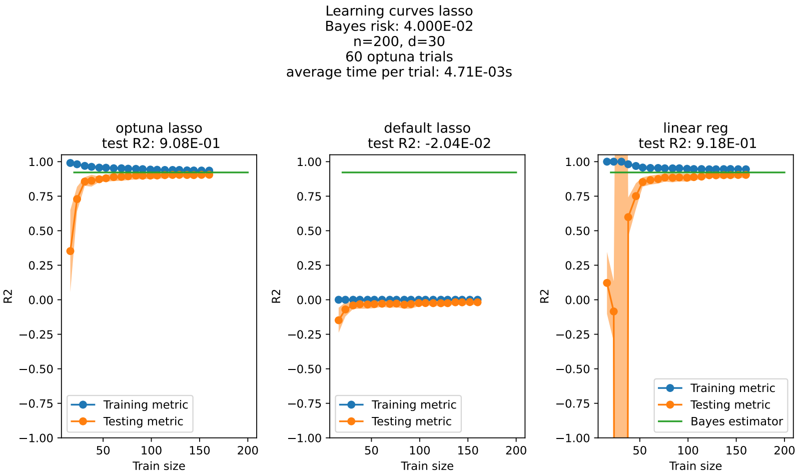
To choose λ_1 and λ_2 : cross validation.

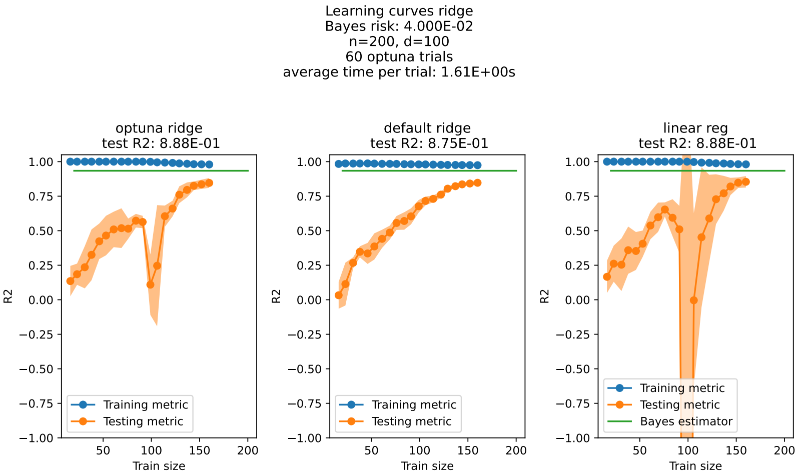
In the following examples, the data are linear, with a sparse θ^*.

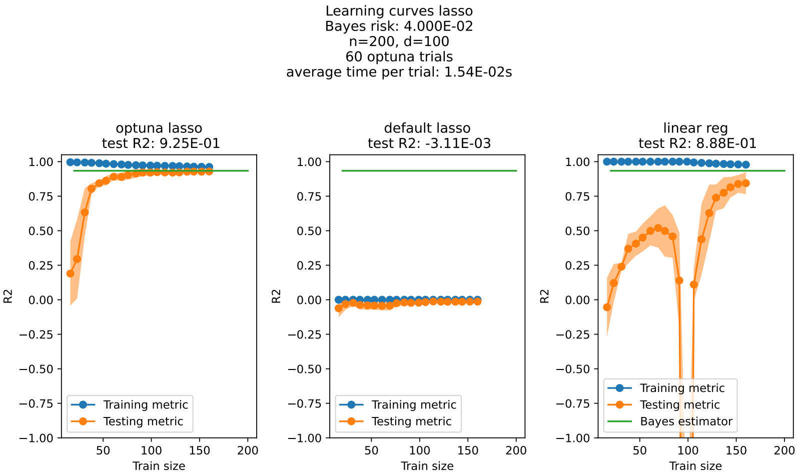
$$y_i = x^T\theta^* + \epsilon_i \tag{10}$$

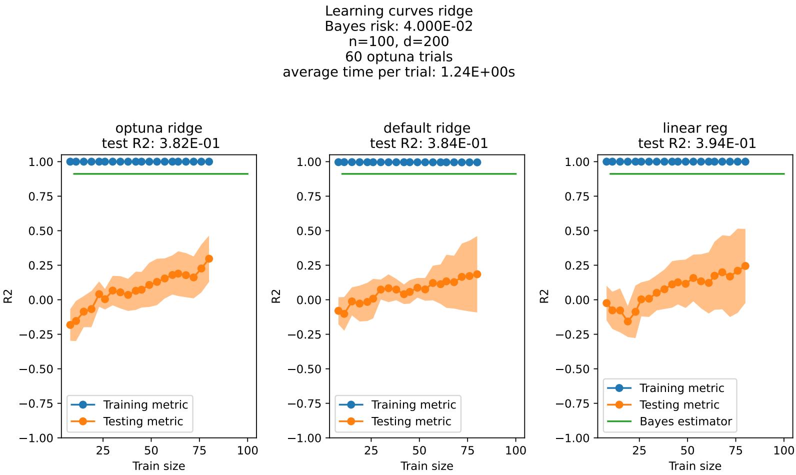
We compare Ridge and Lasso, for several dimensions (n, d).

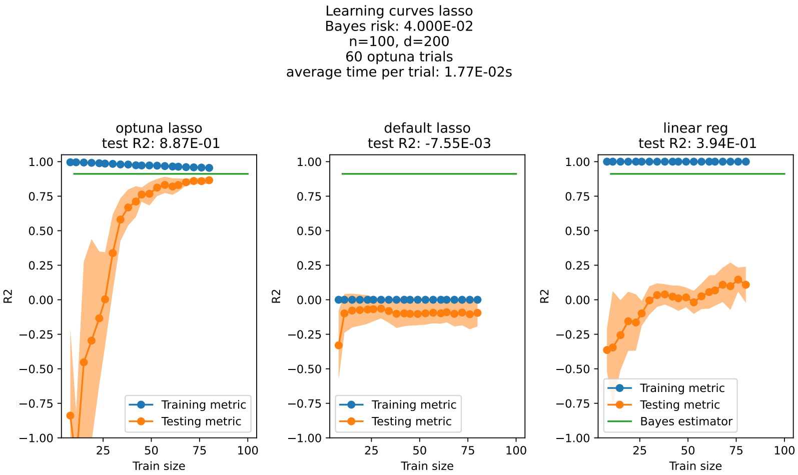

Learning curves ridge
Bayes risk: 4.000E-02
n=200, d=30
60 optuna trials
average time per trial: 4.53E-03s
optuna ridge
test R2: 9.18E-01
default ridge
test R2: 9.09E-01
linear reg
test R2: 9.18E-01
R2
Train size
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
50
100
150
200
Training metric
Testing metric
Bayes estimator

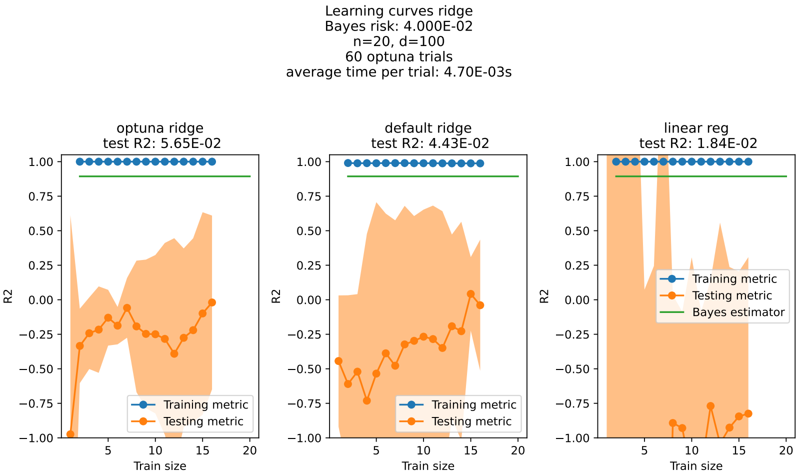

Learning curves lasso
Bayes risk: 4.000E-02
n=200, d=30
60 optuna trials
average time per trial: 4.71E-03s
optuna lasso
test R2: 9.08E-01
default lasso
test R2: -2.04E-02
linear reg
test R2: 9.18E-01
R2
Train size
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
50
100
150
200
Training metric
Testing metric
Bayes estimator

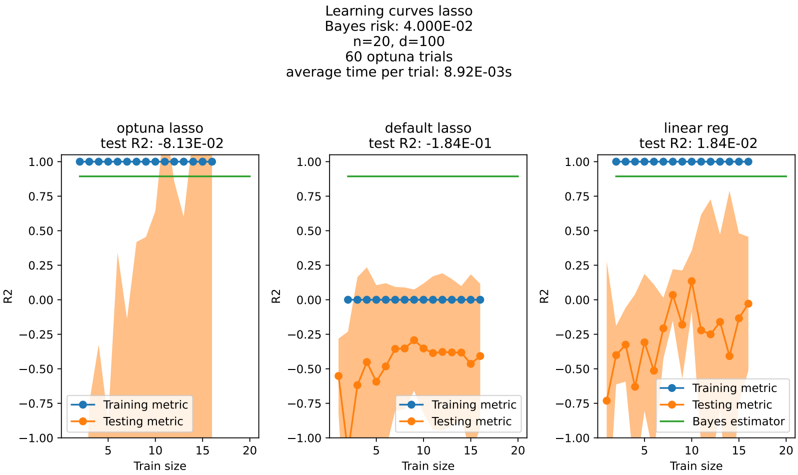

Learning curves ridge
Bayes risk: 4.000E-02
n=200, d=100
60 optuna trials
average time per trial: 1.61E+00s
optuna ridge
test R2: 8.88E-01
default ridge
test R2: 8.75E-01
linear reg
test R2: 8.88E-01
R2
Train size
Training metric
Testing metric
Bayes estimator
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
50
100
150
200


Learning curves lasso
Bayes risk: 4.000E-02
n=200, d=100
60 optuna trials
average time per trial: 1.54E-02s
optuna lasso
test R2: 9.25E-01
default lasso
test R2: -3.11E-03
linear reg
test R2: 8.88E-01
R2
Train size
Training metric
Testing metric
Bayes estimator


Learning curves ridge
Bayes risk: 4.000E-02
n=100, d=200
60 optuna trials
average time per trial: 1.24E+00s
optuna ridge
test R2: 3.82E-01
default ridge
test R2: 3.84E-01
linear reg
test R2: 3.94E-01
R2
Train size
Training metric
Testing metric
Bayes estimator
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
25
50
75
100


Learning curves lasso
Bayes risk: 4.000E-02
n=100, d=200
60 optuna trials
average time per trial: 1.77E-02s
optuna lasso
test R2: 8.87E-01
default lasso
test R2: -7.55E-03
linear reg
test R2: 3.94E-01
R2
Train size
Training metric
Testing metric
Bayes estimator


Learning curves ridge
Bayes risk: 4.000E-02
n=20, d=100
60 optuna trials
average time per trial: 4.70E-03s
optuna ridge
test R2: 5.65E-02
default ridge
test R2: 4.43E-02
linear reg
test R2: 1.84E-02
R2
Train size
Training metric
Testing metric
Bayes estimator


Learning curves lasso
Bayes risk: 4.000E-02
n=20, d=100
60 optuna trials
average time per trial: 8.92E-03s
optuna lasso
test R2: -8.13E-02
default lasso
test R2: -1.84E-01
linear reg
test R2: 1.84E-02
R2
Train size
Training metric
Testing metric
Bayes estimator
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
5
10
15
20

Multiple objective optimization

A estimator might be considered good for several reasons :

- quality of the predictions
- short(er) optimization time
- small(er) computational ressources

Multiple-objective optimization

In optuna for instance, it is possible to do **mutliple objective** optimization.

https://en.wikipedia.org/wiki/Pareto_front

https://optuna.readthedocs.io/en/stable/tutorial/20_recipes/002_multi_objective.html

https://optuna.readthedocs.io/en/stable/reference/visualization/generated/optuna.visualization.plot_pareto_front.html

No free lunch theorems

$\mathcal{A}$: learning rule. Takes the dataset D_n as input and outputs an estimator f_n (for instance based on empirical risk minimization, local averaging, etc).
There are several no free lunch theorems.

No free lunch theorems

Theorem

No free lunch - fixed n

We consider a binary classification task with "0-1"-loss, and $\mathcal{X}$ infinite.

We note $\mathcal{P}$ the set of all probability distributions on $\mathcal{X} \times \{0, 1\}$.

For any $n > 0$ and any learning rule $\mathcal{A}$

$$\sup_{dp \in \mathcal{P}} E\Big[R_{dp}\big(\mathcal{A}(D_n(dp))\big)\Big] - R^*_{dp} \geq \frac{1}{2} \tag{11}$$

We write $D_n(dp)$ in order to emphasize that the dataset is sampled randomly from the distribution dp.

No free lunch

- For any learning rule, there exists a distribution for which this learning rule performs badly.
- No method is universal and can have a good convergence rate on all problems.

However, considering **all** problems is probably not relevant for machine learning.

Adaptivity

If the learning rule improves (faster convergence rate) when we add a property on the problem (for instance, regularity of the target function), we say that we have adaptivity to this property.
For instance : gradient descent is adaptive to the strong convexity of the target function, since with a proper choice of the learning rate γ, the convergence rate is exponential, with a rate that involves the strong convexity constant μ.
There are several forms of adaptivity.

Most general case

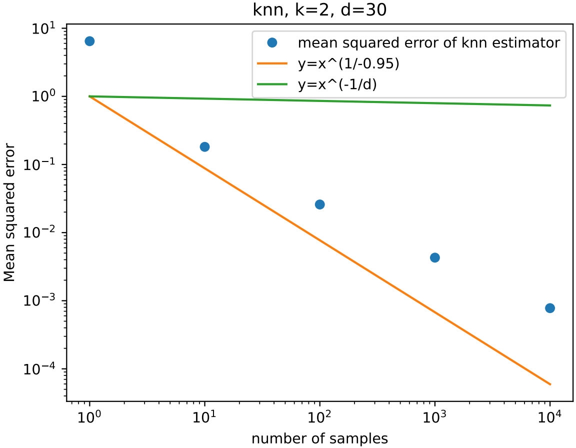
The target is just Lipshitz-continuous, no extra-hypothesis. In this case the optimal rate is of the form $\mathcal{O}(n^{-\frac{1}{d}})$ (curse of dimensionality) for all learning rules.

Adaptivity to the input space

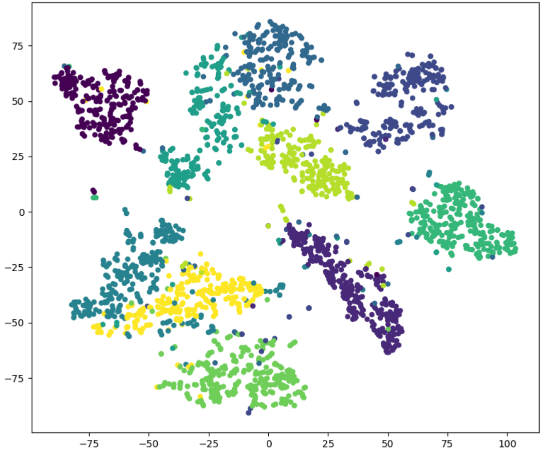
If the input data lie on a submanifold (e.g. a subspace) of $\mathbb{R}^d$ of lower dimension than d, most methods adapt to this property.

Adaptivity to a lower dimensional support

We saw an example of this behavior during a practical seccion.



Lower dimensional manifolds



Adaptivity to the regularity of the target function

If the target is smoother (meaning that all derivatives up to order m are bounded), kernel methods (here, positive-definite kernels) and neural network adapt, if well optimized and regularized. The rate can become $\mathcal{O}(n^{-\frac{m}{d}})$.

Adaptivity to latent variables

If the target function depends only on a k dimensional linear projection of the data, neural networks adapt, if well optimized. The rate can become $O(n^{-\frac{m}{k}})$.
https://francisbach.com/quest-for-adaptivity/

- RL has many applications and is quite a hot topic.
- **Deep Reinforcement Learning** has received a lot of attention recently.

- Atari games

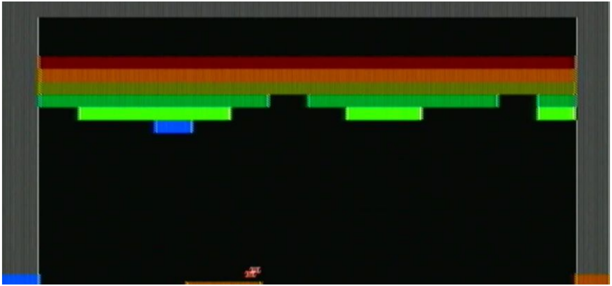


Figure – [Mnih et al., 2013]

- AlphaGo

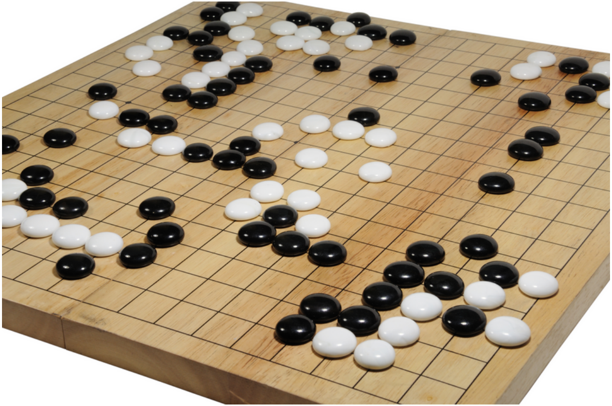


Figure – Go game, beaten by AlphaGo in 2017 [Silver et al., 2016]

- Reinforcement Learning is also being used in the community of **Computationnal neuroscience.**

Supervised learning and Correction

- In **supervised learning**, the supervisor indicates the **expected answer** the model should answer.
- The feedback does not depend on the action performed by the model (for instance the prediction from the model)
- We say that the model receives an **instructive feedback**.
- The model must then **correct its model** based on this answer.

Cost sensitive learning

- In **Cost sensitive learning**, the situation is different.
- The agent receives an **evaluative feedback**. The feedack depends on the action performed by the agent.

Cost sensitive learning

- In **Cost sensitive learning**, the situation is different.
- The agent receives an **evaluative feedback**. The feedack depends on the action performed by the agent.
- **Examples :**
  - AI playing a game and receiving "victory" or "defeat" as a feedback.
  - Child playing with toys.

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.
- In reinforcement learning, the feedback is a **real number**.

Reinforcement learning

- **Reinforcement learning** is a particular case of cost-sensitive learning.
- In reinforcement learning, the feedback is a **real number**.
- **Example :** amount of coins won after a poker turn.

Reinforcement learning

- First, the agent does not know if a reward is good or bad *per se*.
- A reward of -10 can be good or bad depending on the other rewards that are possible to obtain !

Reinforcement learning

- First, the agent does not know if a reward is good or bad per se.
- A reward of -10 good be good or bad depending on the other rewards that are possible to obtain.
- Most of the time, the objective of the agent will be to optimize the **agregation of rewards**.

Reinforcement learning

- The agent lives in a world E, and can be in several states s. The agent performs **actions** a and receives rewards r.

Reinforcement learning

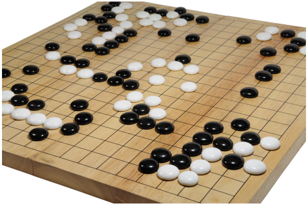
- The agent lives in a world E, and can be in several states s. The agent performs **actions** a and receives rewards r.
- **Examples :**
  - world $= \mathbb{R}^2$
  - state $=$ position
  - actions $=$ moving somewhere
  - reward $=$ amount of food found

Formalization

- There are many aspects of the problem that we need to formalize. Several formalizations are possible depending on the situation.
- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**
- Continuous spaces are also available for RL. In those cases the objects are slightly different, and the optimization procedures also differ. For an introductory course, discrete spaces are more suitable.

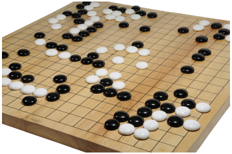
Question

- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**
- Are these hypotheses valid in the case of AlphaGo ?



Question

- We will consider **discrete spaces** :
  - the time will be discrete
  - the number of possible states will be **finite**
  - the number of possible actions will be **finite**
- Are these hypothesis valid in the case of AlphaGo ?



- Yes ! This shows that discrete spaces can still describe very complex problems.

Formalization

- we will write :
  - s_t : state at time t
  - a_t : action performed at time t
  - r_t : reward received at time t
- the actions are chosen according to a **policy** π

Policies

- The policy π is a function of the current state.
- It can be **deterministic :** the action chosen is chosen with probability 1.

Policies

- The policy π is a function of the current state.
- It can be **deterministic :** the action chosen is chosen with probability 1.
- Or **stochastic :** the action performed in a given state is drawn from a **distribution**.

Two levels of randomness

- The policy can be deterministic or stochastic.
- But the result of an action could also be stochastic ! This is called a **stochastic transition function**.

Two levels of randomness

- The policy can be deterministic or stochastic.
- But the result of an action could also be stochastic ! This is called a **stochastic transition function**.

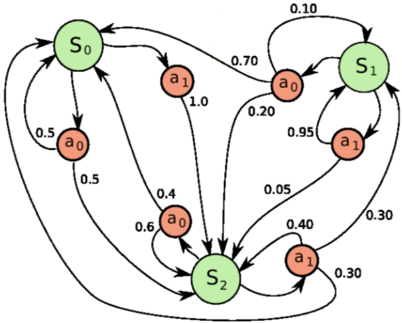


Figure – A stochastic policy with a stochastic transition function.

Exercice 6 :

- What is the probability of staying in state S_0 when performing an action from S_0 ? and from S_1 and S_2 ?

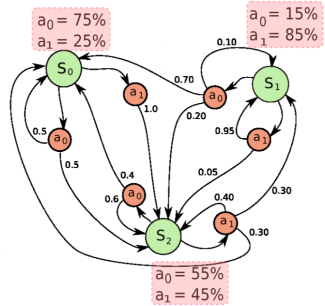


Figure – A stochastic policy with a stochastic transition function.

Agregation of rewards

- The agent want to optimize the **agregation of the rewards.**
- There are several ways to agregate the rewards.

Value function : episodic case

- If the *horizon* is finite (number of steps in the simulation), we can compute the value function of policy π, (assuming the actions are always taken according to policy π)

$$V^{\pi}(s_0) = r_0 + \cdots + r_N \tag{12}$$

- s_0 to the state and V to the *value*.

Value function : episodic case

- If the *horizon* is finite, we can take the sum

$$V^{\pi}(s_0) = r_0 + \cdots + r_N \tag{13}$$

- We could also average a window. For instance a window of size 3 :

$$V^{\pi}(s_0) = \frac{r_0 + r_1 + r_2}{3} \tag{14}$$

Value function : general case

- if the horizon is infinite, the **discount factor** $\gamma \in [0, 1[$ weights the rewards r_k

$$V^\pi(s_0) = \sum_{t=t_0}^{+\infty} \gamma^{t-t_0} r_t \tag{15}$$

More considerations

- The Markov hypothesis
- Exploitation exploration compromise

ϵ-greedy policy

A way to tackle the exploitation-exploration compromise.

- with probability $1 - \epsilon$: go to the best known reward (exploitation).
- with probability ϵ : perform a random action (exploration).

Art

"RL is a science, but dealing with the exploration-exploitation compromise is an art" (Sutton)

Dynamic programming

- in **dynamic** programming, we keep track of the **values** of all the states and assume a known **model** of the world.
- Deterministic transition function.

World

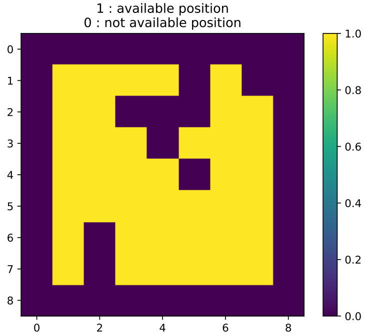


Figure – 2 dimensional world.

Reward

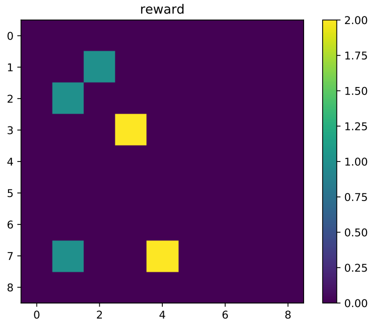


Figure – Reward function.

2D world

- Our agent can move in the 4 directions, one step at a time.
- We will progressively build an agent that learns to evaluate the states and then learns how to go to the best state.

Optimal policy

We look for the value of the **optimal policy** π^* .

$$V^*(s_0) = \max_{(a_t)_{t \in [0, +\infty]}} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \tag{16}$$

$R(s_t, a_t)$ is the reward of doing action a_t in state s_t.

Bellamn optimality equation

Exercice 7 :

- For each state s_0,

$$V^*(s_0) = \max_{(a_t)_{t\in[0,+\infty]}} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \tag{17}$$

- Can you express $V(s_0)$ as a function of $V(s_1)$?

Bellman optimality equation

$$V^*(s_0) = \max_a \left[r_1 + \gamma V^*(s_1(a)) \right] \tag{18}$$

with $s_1(a)$ being the state reached when choosing the action a in state s_0.
This is one of the many forms of **Bellman equations**.
Sutton book :
`http://incompleteideas.net/book/the-book-2nd.html`

Value Iteration

Value iteration belongs to dynamic programming methods. They differ from RL in that a perfect model of the environment is assumed.
These methods are building blocks for RL.

Value Iteration

- First, the initial Value function for all the states is 0.

Value Iteration

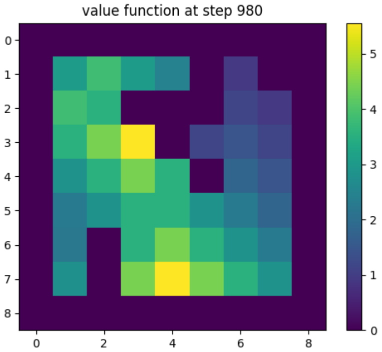
- First, the initial Value function for all the states is 0.
- Then we propagate the information about the rewards between the states, in order to **update the value function**
- We can find an optimal policy in the following way :

$$\forall s \in V(s_t) \leftarrow \max_{a_t} \left(r_{s_t} + \gamma V(s_{t+1}) \right) \tag{19}$$

(s_{t+1} depends on a_t).

Value iteration

- After learning, we will obtain a value function



Optimal policy

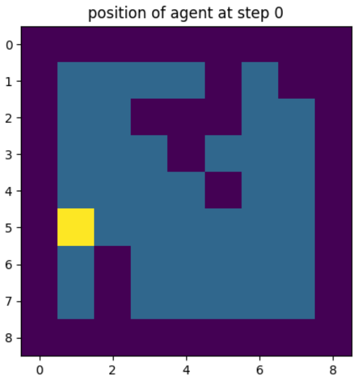


Figure – After learning hte optimal policy, the agent can go to the reward.

Optimal policy

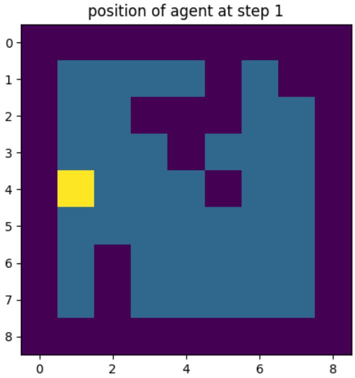


Figure – After learning, the agent can go to the reward.

Optimal policy

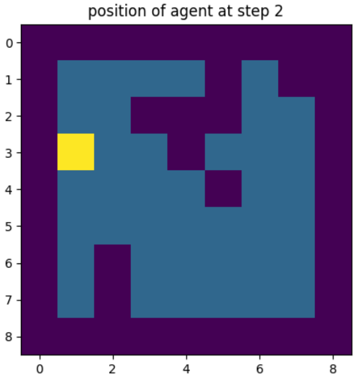


Figure – After learning, the agent can go to the reward.

Optimal policy

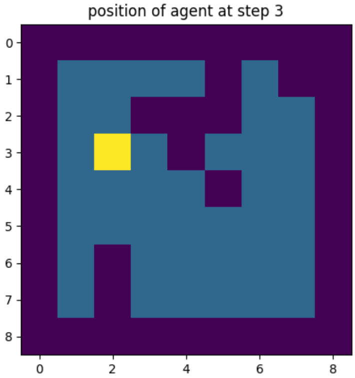


Figure – After learning, the agent can go to the reward.

Optimal policy

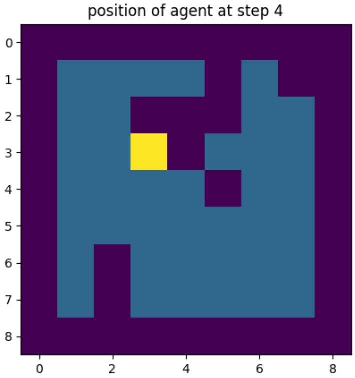


Figure – After learning, the agent can go to the reward.

Optimal policy

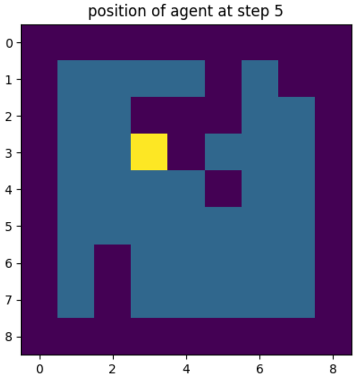


Figure – After learning, the agent can go to the reward.

Multiple paradigms

- Reinforcement learning has many variants.
- In the ones we studied, a model of the effect of our actions were known.
- This is not always de case.

Temporal difference learning

- In temporal difference learning, the agent does not know a **model** of its world.
- But it can still learn the value function with the **TD updates**

Temporal difference learning

- In temporal difference learning, the agent does not know a **model** of its world.
- But it can still learn the value function with the **TD updates**

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{20}$$

Monte Carlo methods

Monte Carlo methods can be used in Reinforcement Learning to estimate the expected values of some random variables (such as the expected reward in a given state).

Actor critic methods

- Sometimes you can use **two** policies
  - the **behavior policy** provides actions and guarantees exploration
  - the **target polivy** is the optimal policy learned in parallel by the agent, that would be used in exploitation mode.

Tabular case and continous case

- We studied **finite** (and thus discrete situations).
- However, RL can also be applied to continuous state / discrete action spaces (DQN).

Tabular case and continous case

- We studied **finite** (and thus discrete situations).
- However, RL can also be applied to continuous state / discrete action spaces (DQN) -> notions of approximation and generalization.
- And even to continous state / continous action spaces (DDPG).

References I

Azencott, C.-A. (2022).
Introduction au Machine Learning - 2e éd.

Le Hir, N., Sigaud, O., and Laflaquière, A. (2018).
Identification of Invariant Sensorimotor Structures as a Prerequisite for the Discovery of Objects.
Frontiers in Robotics and AI, 5(June) :1–14.

Mnih, V., Silver, D., and Riedmiller, M. (2013).
Deep Q Network (Google).
Arxiv.

References II

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016).
Mastering the Game of Go with Deep Neural Networks and Tree Search.
Nature, 529(7587) :484–489.