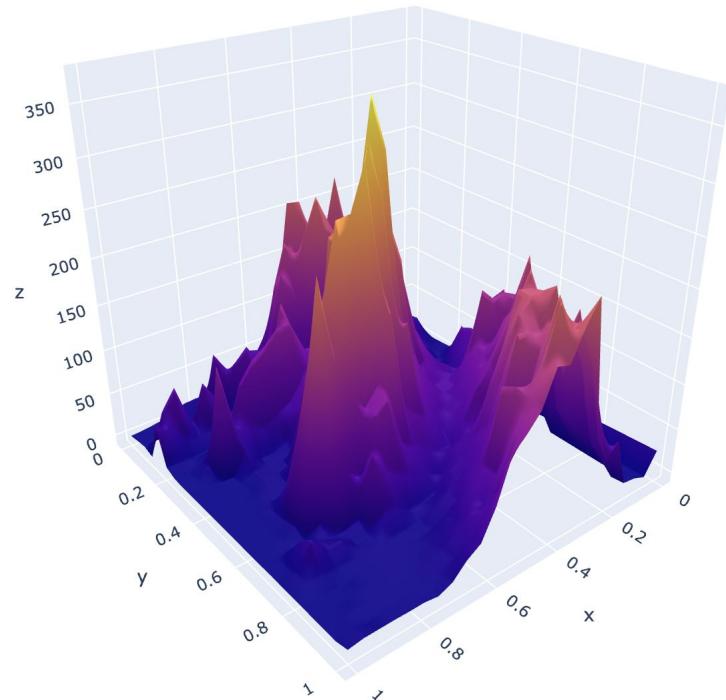


A painting depicting a group of figures in a dark, opulent setting, possibly a library or a study. In the center, a large circular opening in the ceiling allows bright light to illuminate a vast, detailed scene of a city or landscape. Several figures are gathered around this light source, their backs to the viewer, looking intently at the projection. The room is filled with bookshelves and architectural details, creating a sense of historical depth.

Alexandre Stenger
Laurent Eschbach

INTELLIGENCE ARTIFICIELLE

Chapitre I. Avant-propos



I.0 Introduction

A. IA : Un mot valise

3 concepts, du général au particulier

Intelligence Artificielle : Faire effectuer une tâche humaine à un ordinateur

I.0 Introduction

A. IA : Un mot valise

3 concepts, du général au particulier

Intelligence Artificielle : Faire effectuer une tâche humaine à un ordinateur



Machine Learning (Apprentissage Machine) : Faire **apprendre** une tâche humaine à un ordinateur

I.0 Introduction

A. IA : Un mot valise

3 concepts, du général au particulier

Intelligence Artificielle : Faire effectuer une tâche humaine à un ordinateur



Machine Learning (Apprentissage Machine) : Faire **apprendre** une tâche humaine à un ordinateur

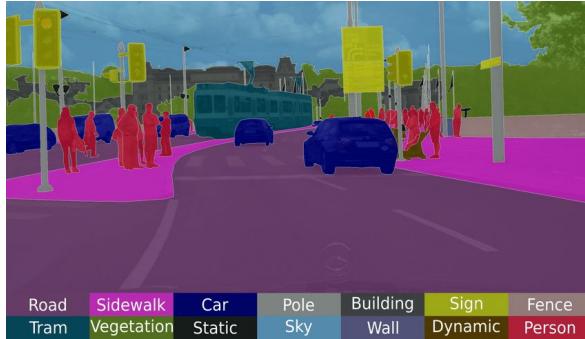


Deep Learning (Apprentissage Profond) : Faire **apprendre** une tâche humaine à l'ordinateur à l'aides des **réseaux de neurones profonds** (Deep Neural Networks)

I.0 Introduction

A. IA : Un mot valise

Modèles prédictifs



ballplayer 50.61%



anemone_fish 97.65%



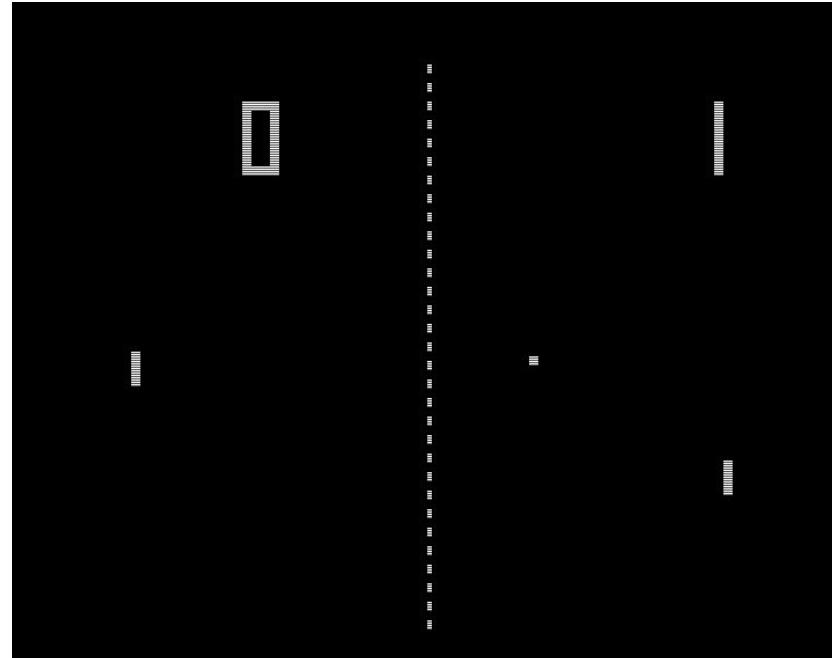
African_elephant 52.34%

Modèles génératifs



I.0 Introduction

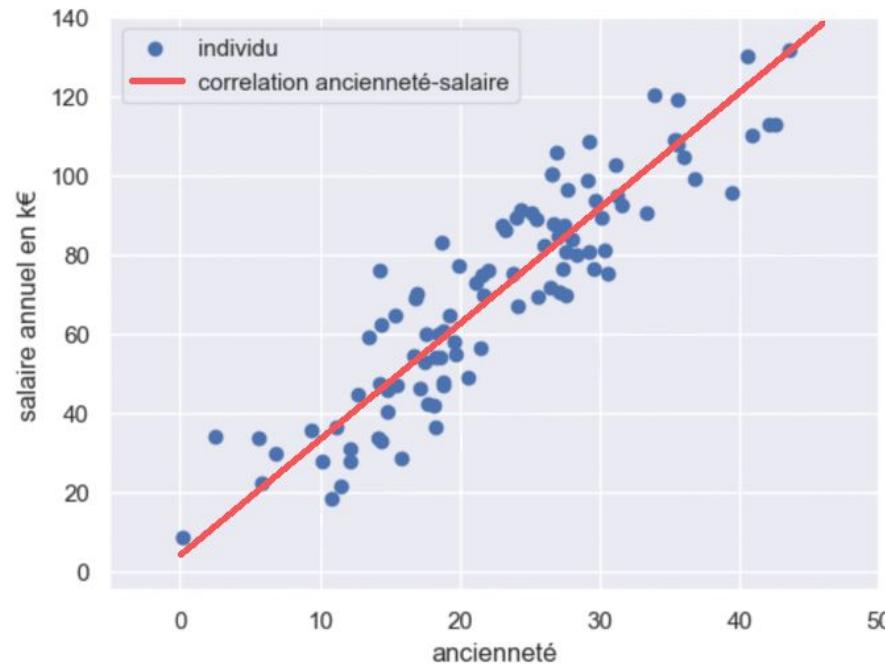
B. Quelques exemples d'IA



Le jeu Pong

I.0 Introduction

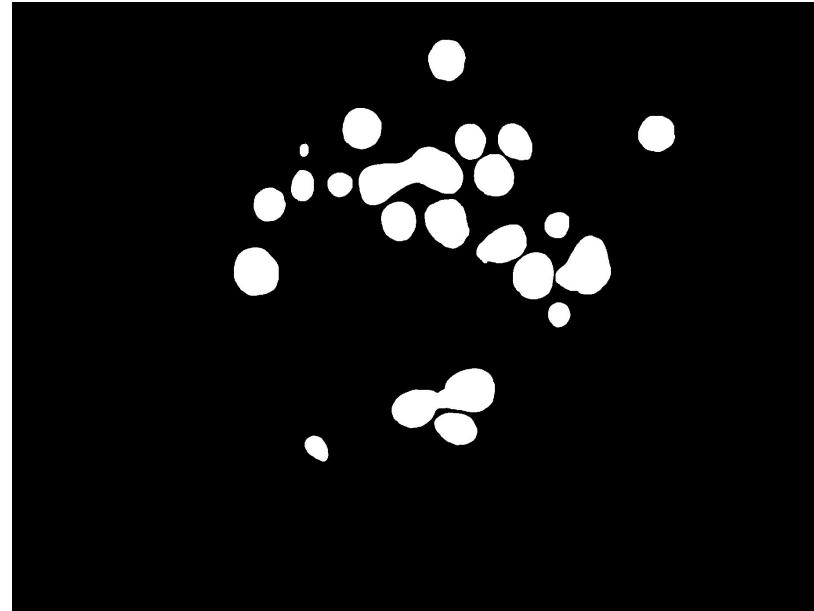
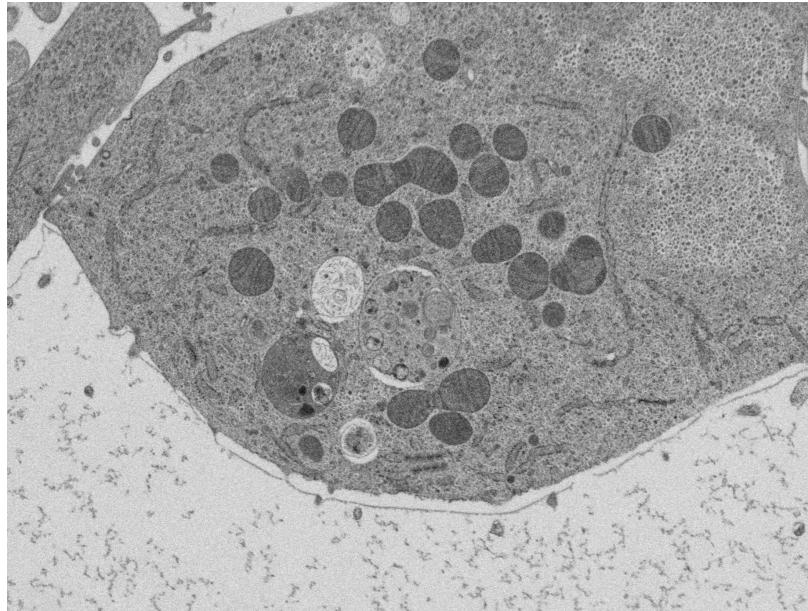
B. Quelques exemples d'IA



Une régression linéaire

I.0 Introduction

B. Quelques exemples d'IA

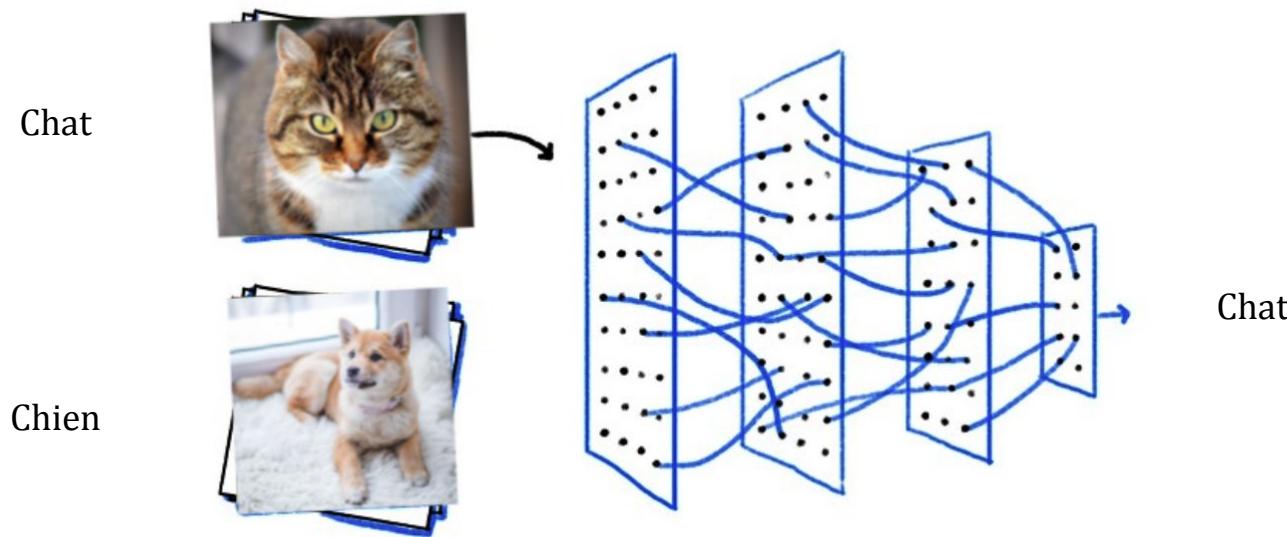


De la segmentation d'images médicales
(ici, segmentation de mitochondries).

Images IGMBC

I.0 Introduction

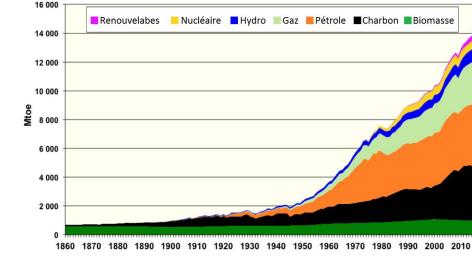
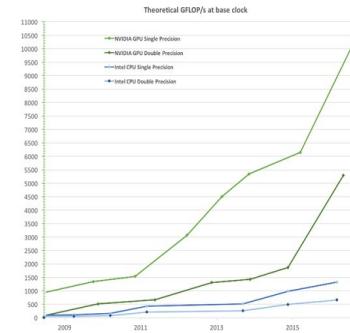
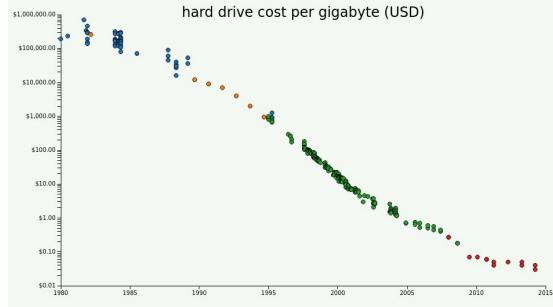
B. Quelques exemples d'IA



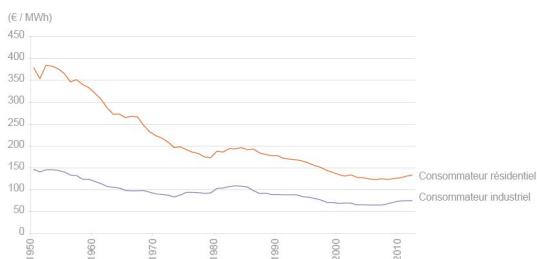
De la classification d'images

I. Introduction

D. Pourquoi ce regain d'intérêt ?



Évolution du prix de l'électricité TTC entre 1950 et 2012 en € constants par MWh



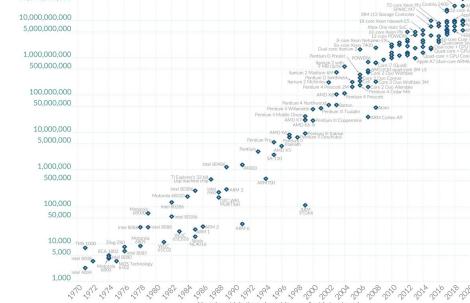
Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.

This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World in Data

Transistor count



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

OurWorldInData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

I. Introduction

Ce qu'il faut en retenir

Machine Learning et Deep Learning reposent bien plus sur une **abondance matériel et énergétique** que sur **des innovations scientifiques conceptuelles et méthodologiques** concepts/méthodes imaginés dans les années 80 mais concrétisation aujourd'hui:

Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986 Oct 9.

Learning representations by back-propagating errors

**David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California,
San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University,
Pittsburgh, Philadelphia 15213, USA

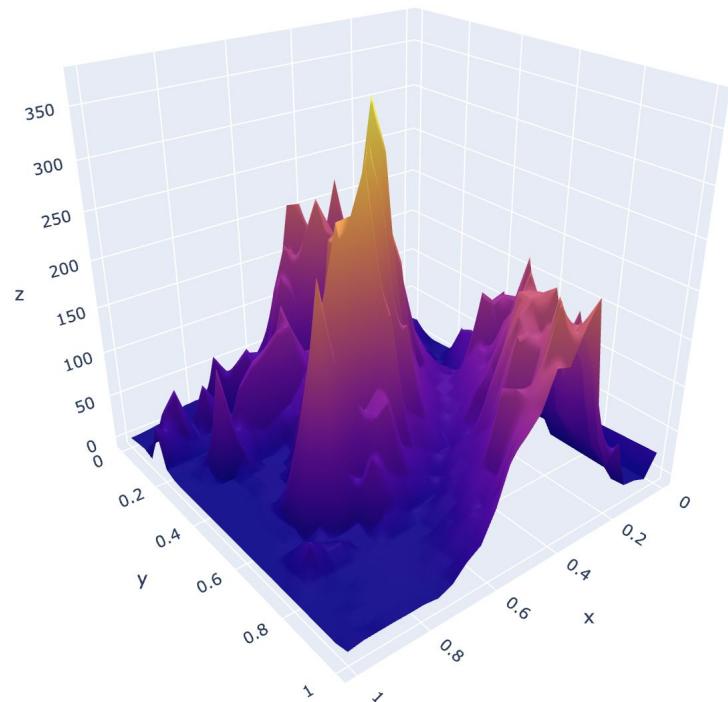
I. Introduction

D. Pourquoi ce regain d'intérêt ?

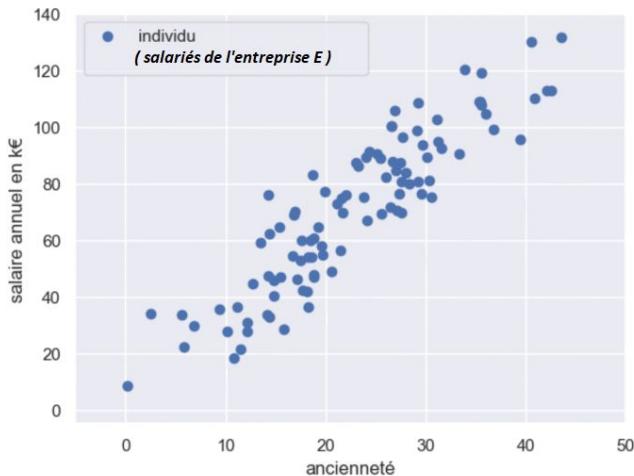


GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO ₂ eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh
LLaMA-7B	A100-80GB	400W	82,432	36 MWh
LLaMA-13B	A100-80GB	400W	135,168	59 MWh
LLaMA-33B	A100-80GB	400W	530,432	233 MWh
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh

Chapitre II. Introduction aux principes de l'IA



II.1 La Régression Linéaire

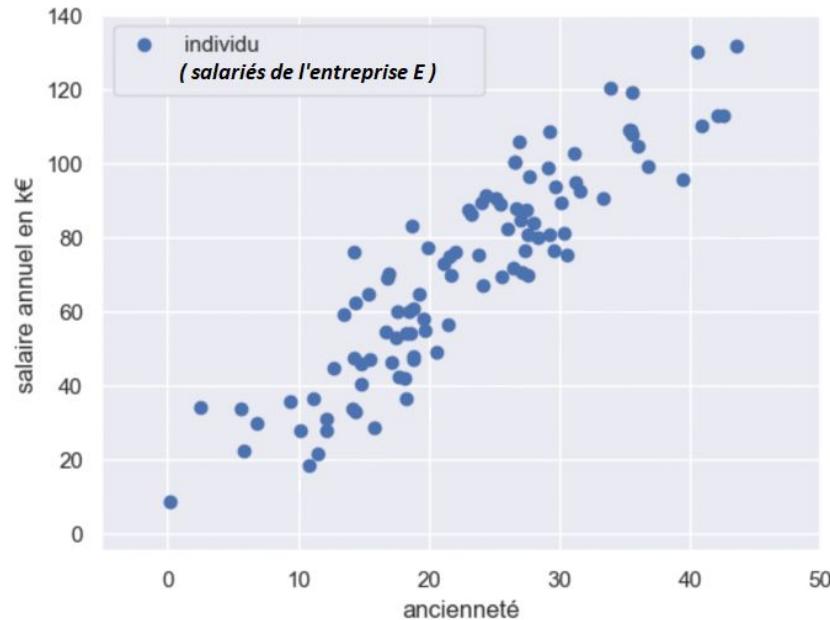


Objectifs :

- Comprendre la notion de **Régression**
- Comprendre la notion d'**Apprentissage supervisé**
- Distinction **Fonction de prédiction/Fonction d'erreur**
- Principe d'Apprentissage par **Descente de gradient**

II.1 La Régression Linéaire

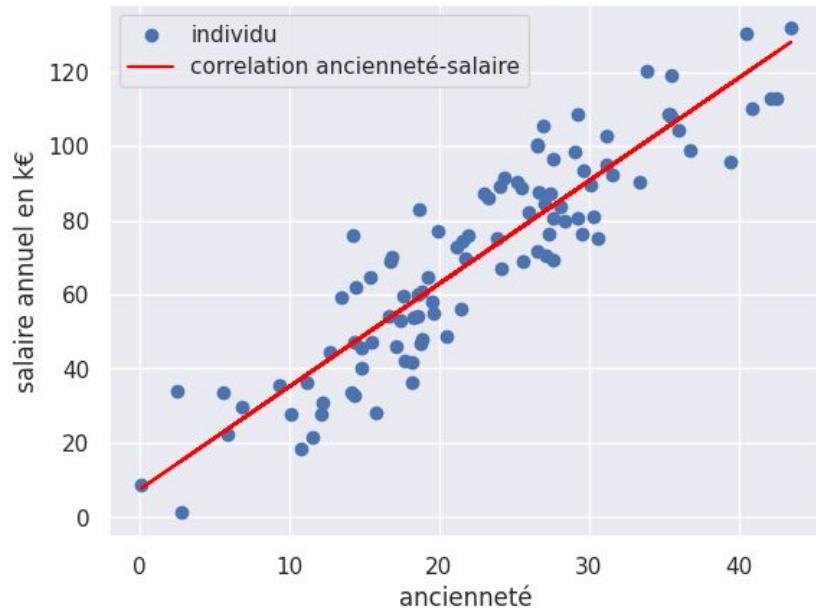
A. Notations



Quel est le but d'une régression linéaire ?

II.1 La Régression Linéaire

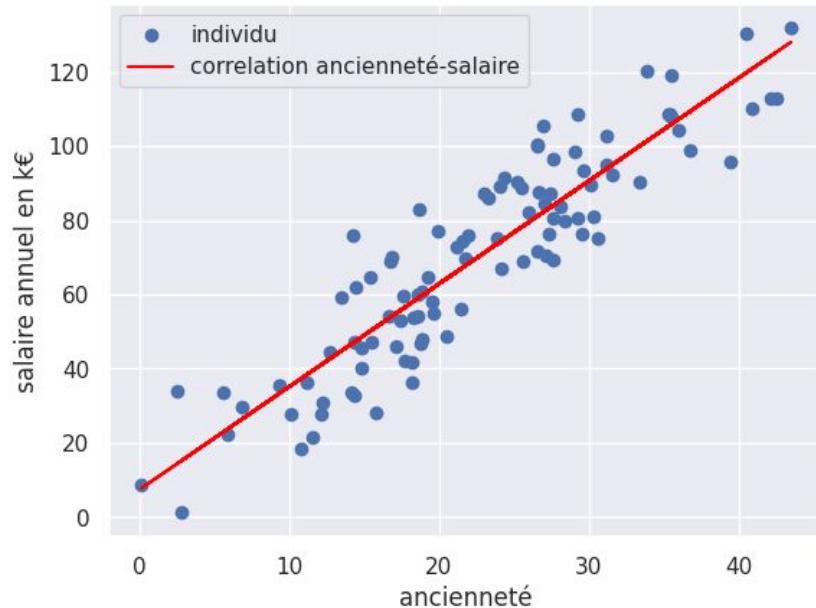
A. Notations



Ici, sachant l'ancienneté d'un salarié, on veut pouvoir prédire son salaire

II.1 La Régression Linéaire

A. Notations

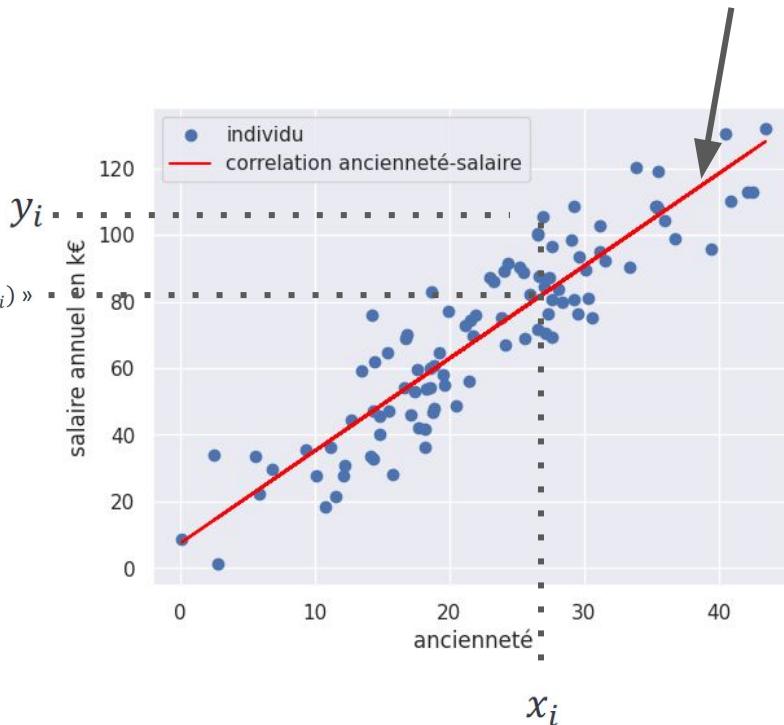


On aura une **prédition** du salaire de la personne en fonction de son ancienneté.

II.1 La Régression Linéaire

A. Notations

fonction \hat{f} de « prédition »



$$\begin{cases} X = (x_1, x_2, \dots, x_i, \dots, x_n) \in R^n \\ Y = (y_1, y_2, \dots, y_i, \dots, y_n) \in R^n \end{cases}$$

Objectif : modéliser une fonction \hat{f} de « prédition »
qui décrive la corrélation (ancienneté, salaire) : $\hat{y} = \hat{f}(x)$

II.1 La Régression Linéaire

B. Comment trouver les coefficients de régression ?

$$a = \frac{\bar{y} \sum x - \sum xy}{\bar{x} \sum x - \sum x^2} \quad \text{et} \quad b = \bar{y} - a \bar{x}$$

où

$$\bar{x} = \frac{\sum x}{m} \quad \text{et} \quad \bar{y} = \frac{\sum y}{m}$$

Moindres carrés ?

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

$$a = \frac{\bar{x}\sum y - \sum xy}{\bar{x}\sum x - \sum x^2} \quad \text{et} \quad b = \bar{y} - a\bar{x}$$

où

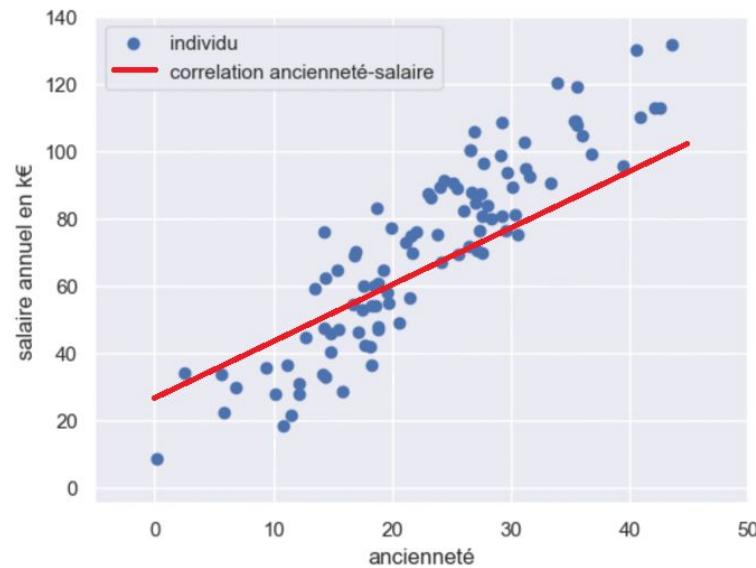
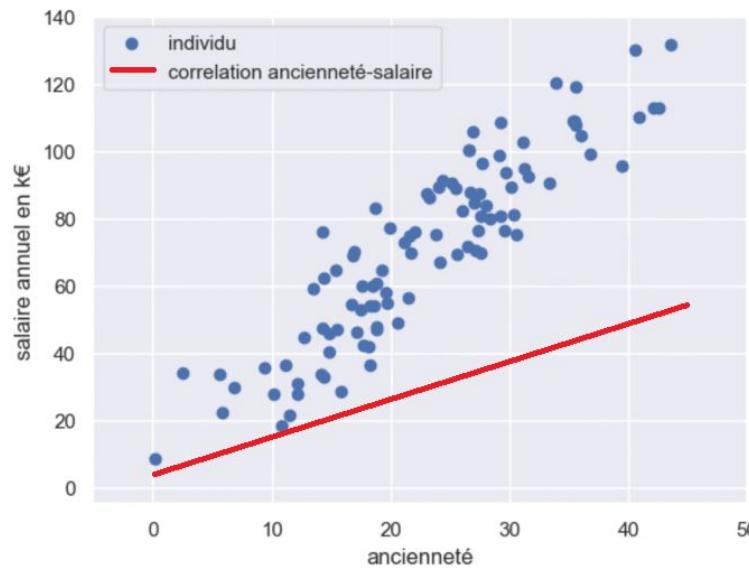
$$\bar{x} = \frac{\sum x}{m} \quad \text{et} \quad \bar{y} = \frac{\sum y}{m}$$

-> Pas d'apprentissage, on donne des formules "explicites" à l'ordinateur

II.1 La Régression Linéaire

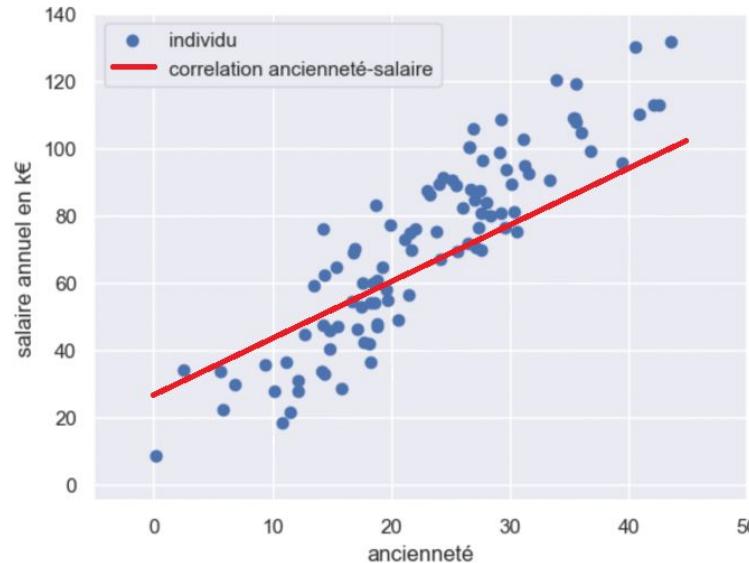
C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

Intuition 1 : jouons avec les coefficients



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage



Comment modéliser cette intuition ?

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

Commençons par simplifier encore notre modèle, fixons $b=0$.

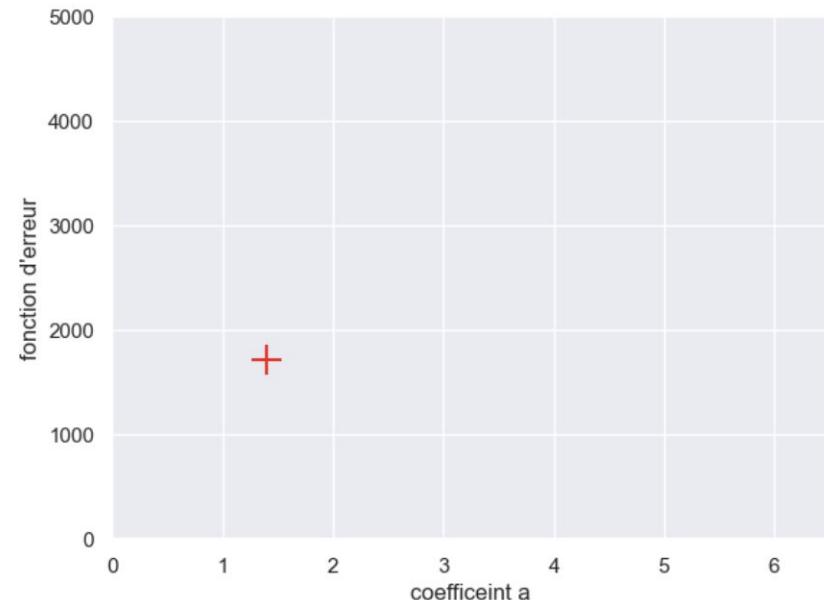
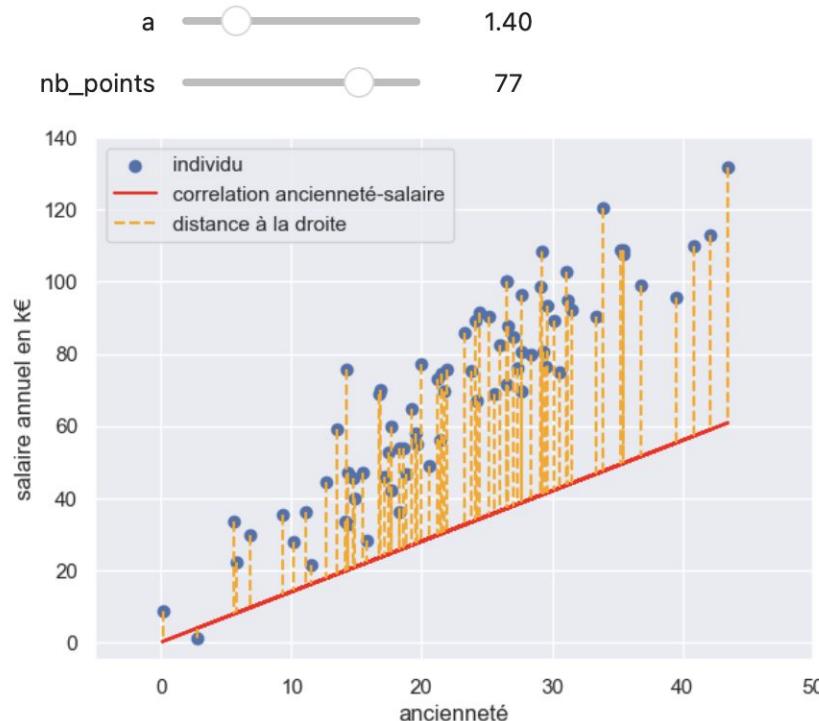
On veut chercher donc uniquement le coefficient a

L'équation de notre fonction de prédiction devient

$$\hat{f}(x) = \hat{y} = ax$$

II.1 La Régression Linéaire

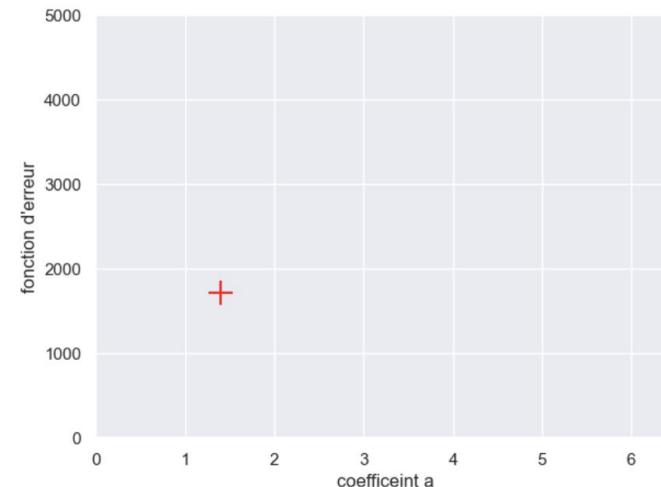
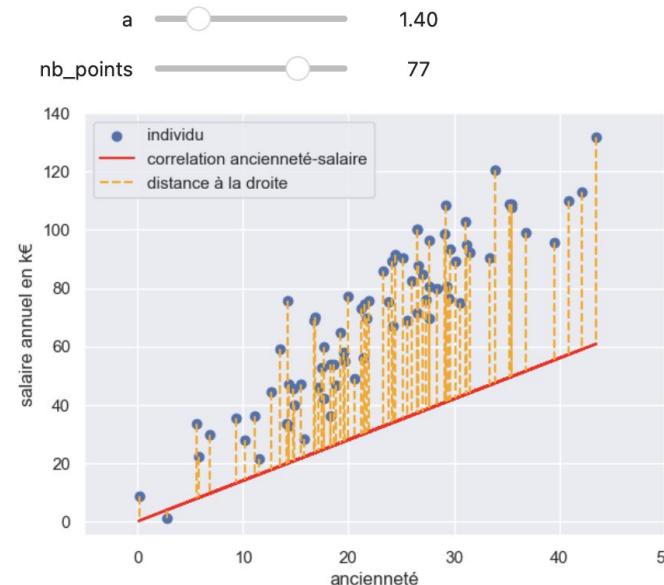
C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

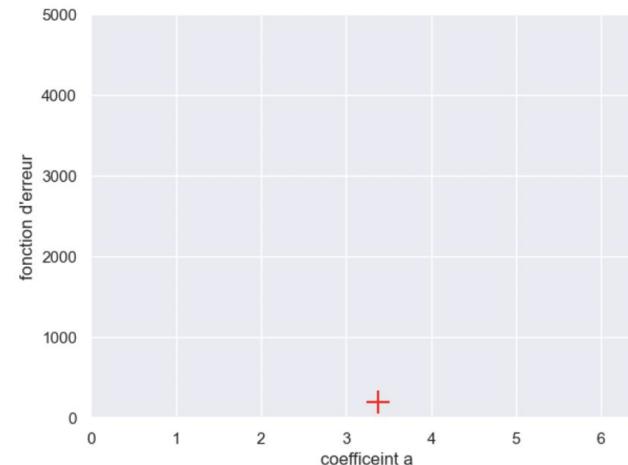
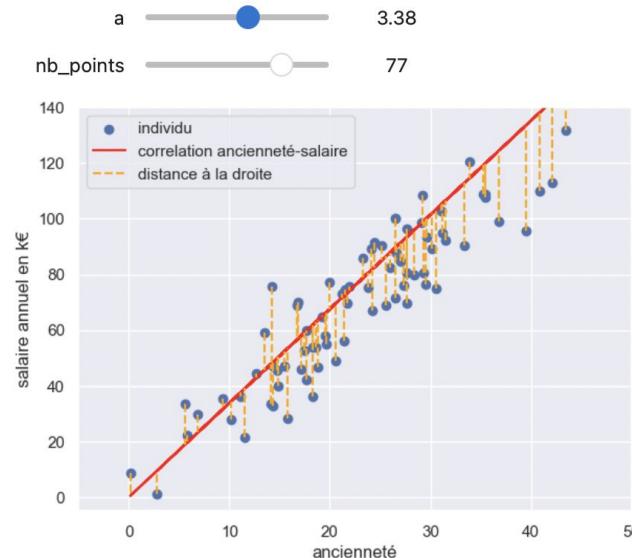
Somme des distances en fonction du coefficient a



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

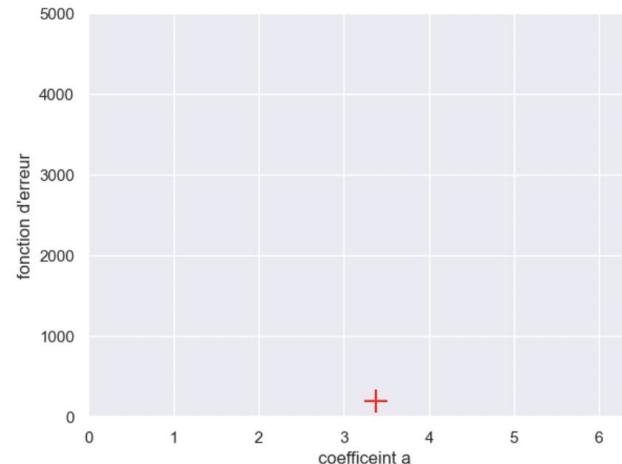
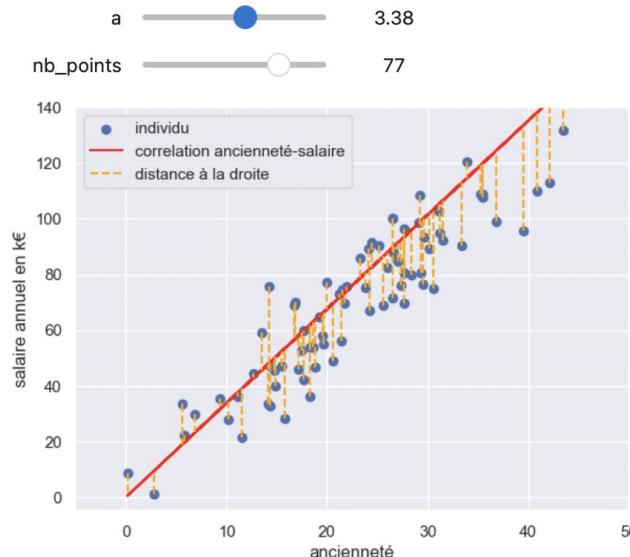
Somme des distances en fonction du **coefficient a**



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

On veut minimiser la somme des distances des points à la droite $\hat{f}(x) = \hat{y} = ax$

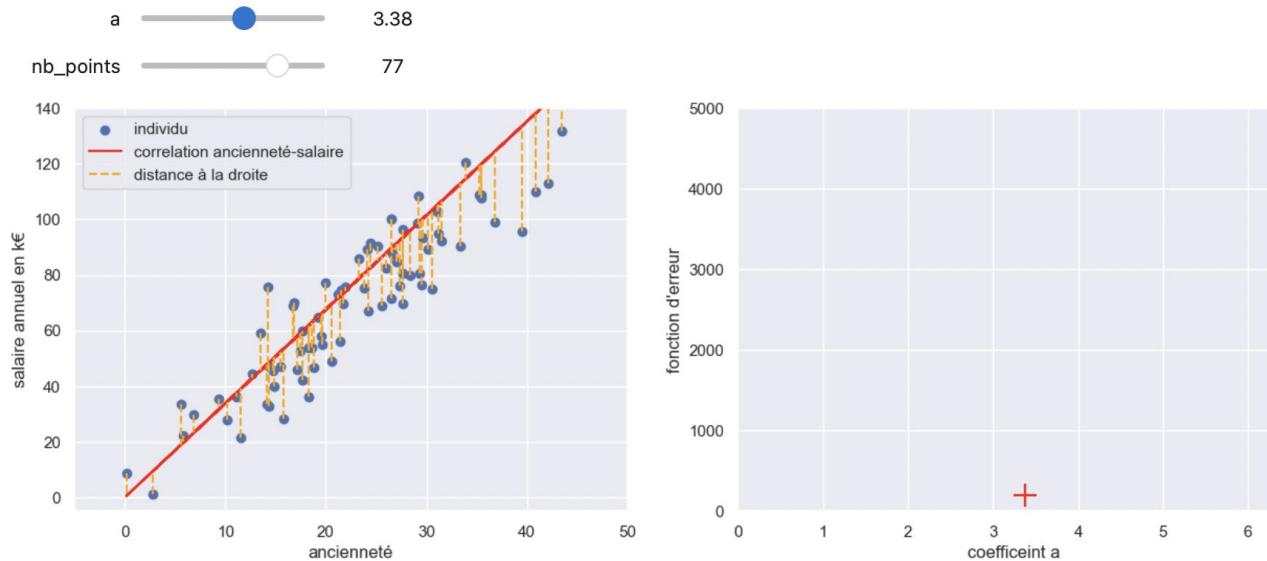


II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

On veut minimiser la somme des distances des points à la droite $\hat{f}(x) = \hat{y} = ax$

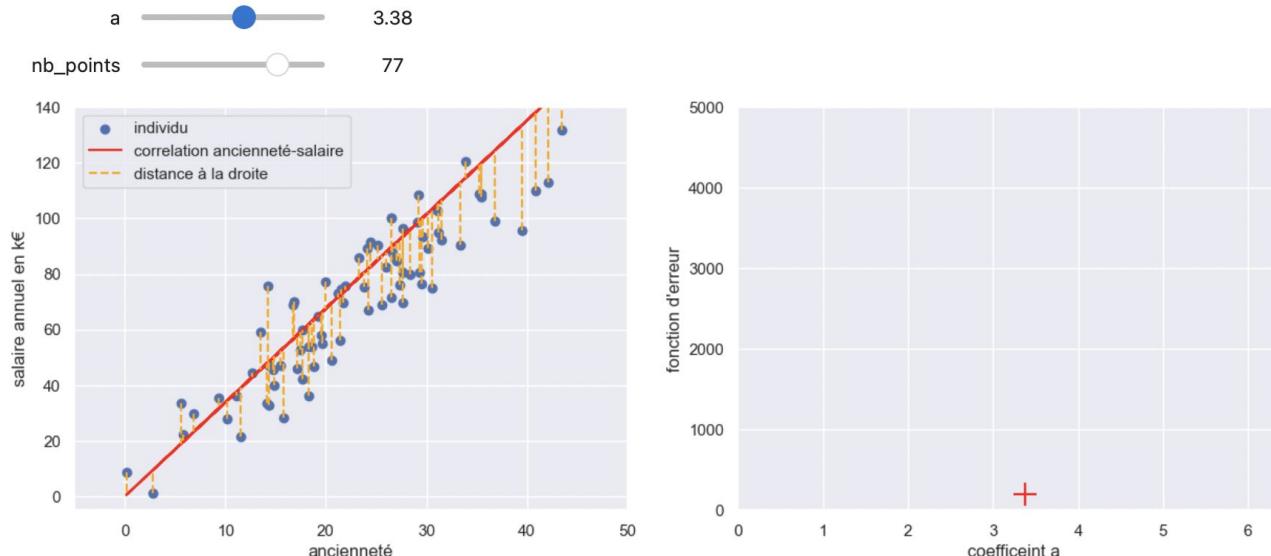
Maintenant qu'on a verbalisé ce qu'on souhaite faire, nous avons besoin de le formaliser "Mathématiquement" et informatiquement



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

On veut minimiser la somme des distances des points à la droite $\hat{f}(x) = \hat{y} = ax$
On généralisera plus tard au cas $f(x) = ax+b$



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

À vous de jouer



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

$$d(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

2ème étape : modéliser la somme des distances

À vous de jouer

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

2ème étape : modéliser la somme des distances

$$\sum_{i=1}^n (ax_i - y_i)^2$$

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

2ème étape : modéliser la somme des distances

3ème étape : on veut minimiser le tout !

$$C(a) = \frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2$$

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut minimiser la somme des distances des points à la droite $f(x) = ax$

1ère étape : modéliser la distance d'un point (x,y) à la droite de prédiction

2ème étape : modéliser la somme des distances

3ème étape : on veut minimiser le tout !

Problème d'optimisation à résoudre :

$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$

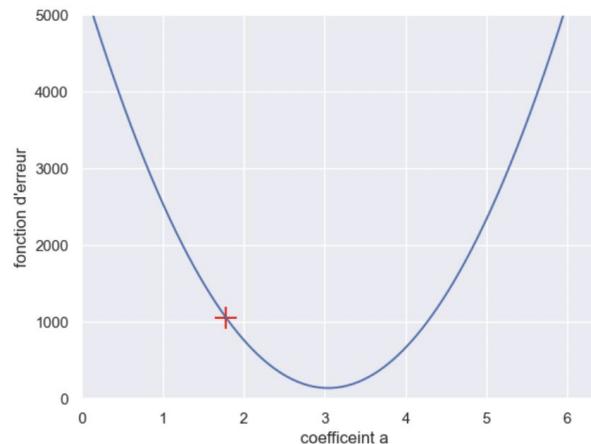
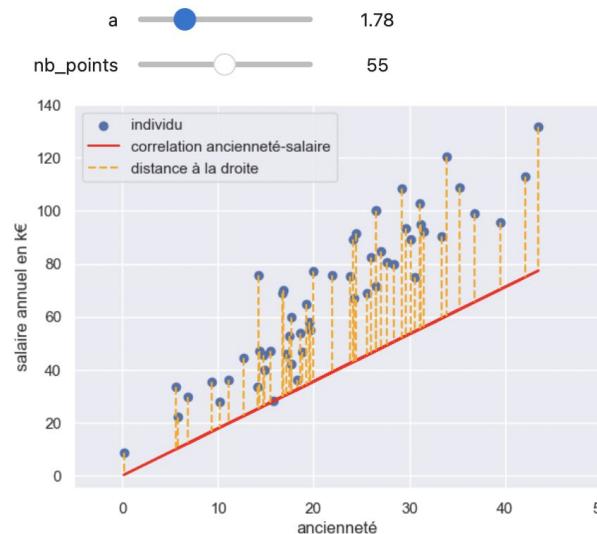
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut résoudre $\min_a (C(a)) = \min_a (\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2)$

Visualisons cette somme de distances de manière plus précise ! (a=1.78)



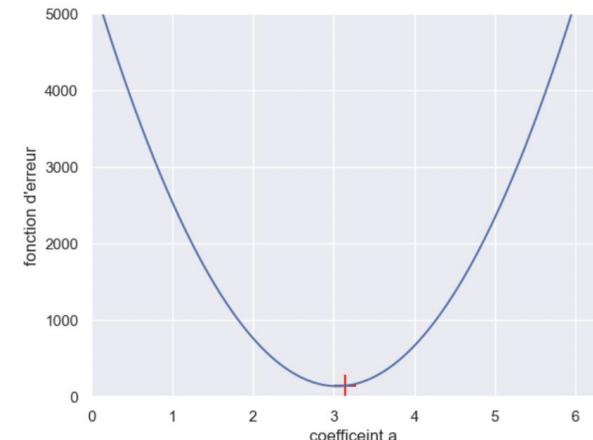
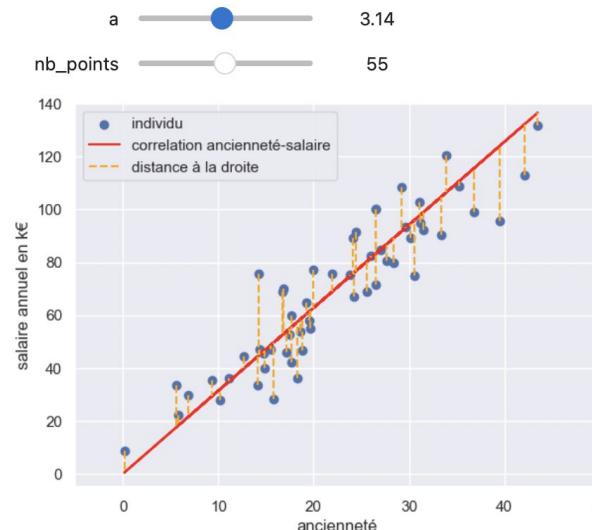
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

On veut résoudre $\min_a (C(a)) = \min_a (\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2)$

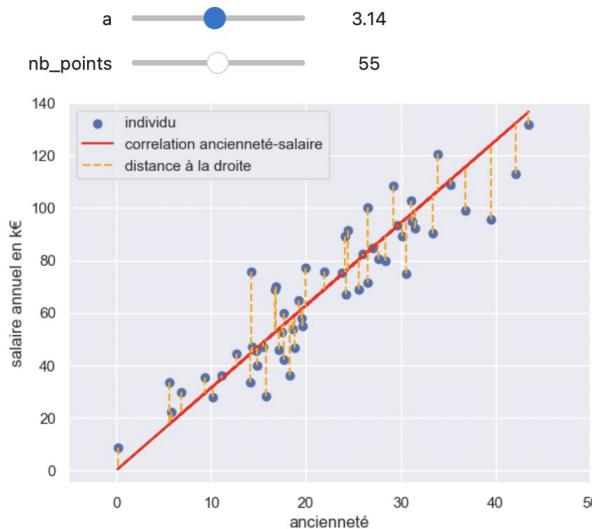
Visualisons cette somme de distances de manière plus précise ! (a=3.14)



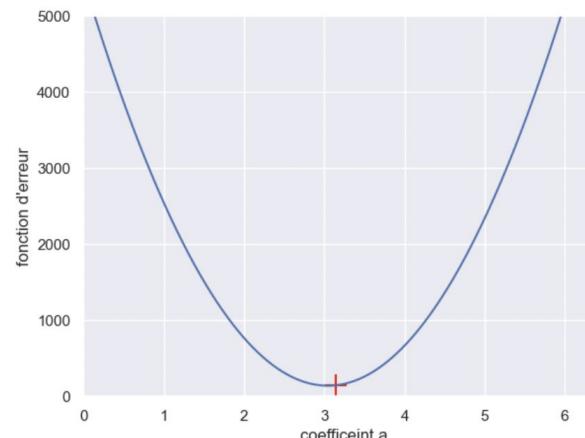
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur



$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$



Un bon physicien résoudrait l'équation :

$$\frac{dC}{da} = 0$$

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$

Un bon physicien résoudrait l'équation :

$$\frac{dC}{da} = 0$$

Mais, on en revient au même problème qu'au début, si on fait ça, on donne explicitement la formule à l'ordinateur pour qu'il calcule la solution => PAS D'APPRENTISSAGE

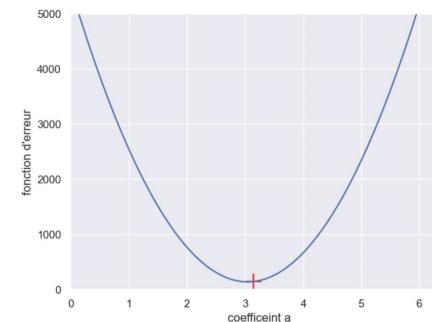
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$

Revenons à l'intuition, comment est-ce qu'on ferait pour le minimiser “à la main”



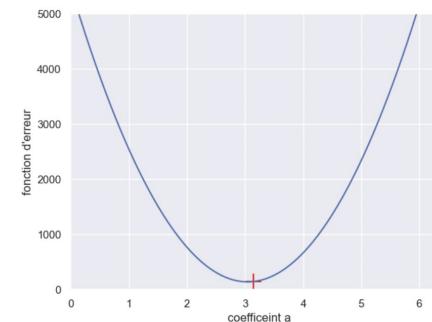
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$

Revenons à l'intuition, comment est-ce qu'on ferait pour le minimiser “à la main”



=> Comment mettre cela en place informatiquement ?

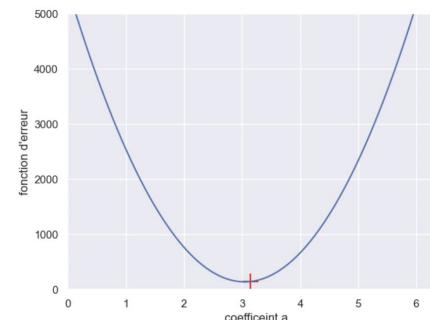
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

a. Modélisation mathématique de l'erreur

$$\min_a(C(a)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i - y_i)^2\right)$$

Revenons à l'intuition, comment est-ce qu'on ferait pour le minimiser “à la main”



=> Comment mettre cela en place informatiquement ?

Essayons de réfléchir à un algo en 2 lignes...

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a



Commençons par calculer $\frac{dC}{da}$

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a



Commençons par calculer $\frac{dC}{da}$

À vous de jouer !

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a

—————> Commençons par calculer $\frac{dC}{da}$

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$



II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$

Maintenant qu'on a la dérivée, on se demande comment mettre à jour le coefficient a...

À vous de jouer ! (disjonction de cas)

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Algorithme d'apprentissage en 2 étapes :

- 1) Calcul de la dérivée de C (fonction de coût)
- 2) Mettre à jour le coefficient a

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$

Maintenant qu'on a la dérivé, on se demande comment mettre à jour le coefficient a...

$$a := a - L \frac{dC(a)}{da}$$

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

b. Modélisation informatique : La descente de gradient

Tant que L'apprentissage n'est pas fini Faire

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$

$$a := a - L \frac{dC(a)}{da}$$

Algorithme de la descente de gradient pour optimiser un paramètre a

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

c. Visualisation d'une descente de gradient avec 1 paramètre à optimiser

Tant que L'apprentissage n'est pas fini Faire

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$

$$a := a - L \frac{dC(a)}{da}$$

Maintenant qu'on a implémenté et compris l'algorithme, regardons en pratique comment la descente de gradient se comporte

Algorithme de la descente de gradient pour optimiser un paramètre a

II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

c. Visualisation d'une descente de gradient avec 1 paramètre à optimiser

Tant que L'apprentissage n'est pas fini Faire

$$\frac{dC(a)}{da} = \frac{2}{n} \sum_{i=1}^n x_i(ax_i - y_i)$$

$$a := a - L \frac{dC(a)}{da}$$

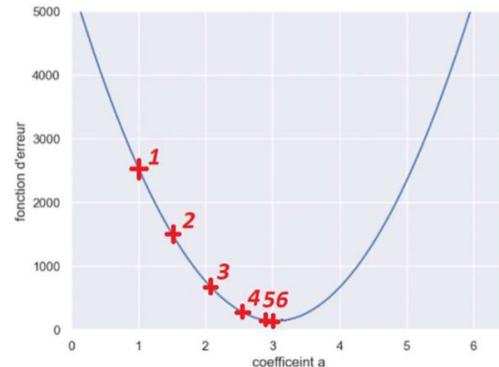
Concentrons-nous
désormais sur le choix du
taux d'apprentissage L

*Algorithme de la descente de gradient pour
optimiser un paramètre a*

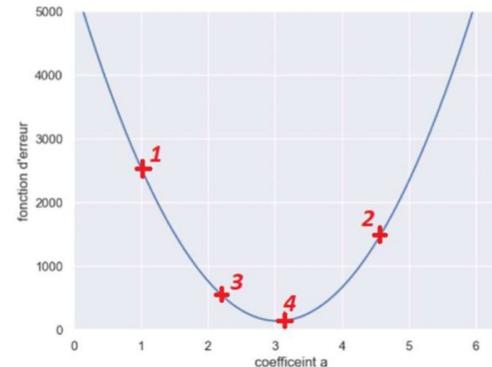
II.1 La Régression Linéaire

C. Qu'est-ce que l'Apprentissage, de l'intuition à la construction d'un algorithme d'Apprentissage

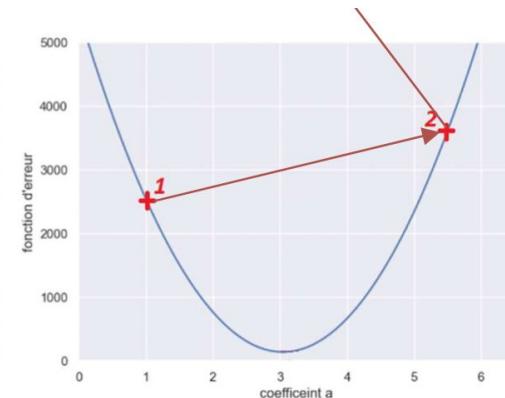
c. Visualisation d'une descente de gradient avec 1 paramètre à optimiser



$$L = 10^{-5}$$



$$L = 10^{-2}$$



$$L \in [0, 1 ; 1]$$

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Jusqu'ici, nous avons travaillé avec la fonction $\hat{f}(x) = ax$

Qu'en est-il si on re prend la fonction $\hat{f}(x) = ax + b$?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Jusqu'ici, nous avons travaillé avec la fonction $\hat{f}(x) = ax$

Qu'en est-il si on re prend la fonction $\hat{f}(x) = ax + b$?

Nouveau problème d'optimisation, cette fois-ci à 2 paramètres :

$$\min_a(C(a, b)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i + b - y_i)^2\right)$$

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Nouveau problème d'optimisation, cette fois-ci à 2 paramètres :

$$\min_a(C(a, b)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i + b - y_i)^2\right)$$

Est-ce que cela change quelque chose à l'algorithme de la descente de gradient ?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Nouveau problème d'optimisation, cette fois-ci à 2 paramètres :

$$\min_a(C(a, b)) = \min_a\left(\frac{1}{n} \sum_{i=1}^n (ax_i + b - y_i)^2\right)$$

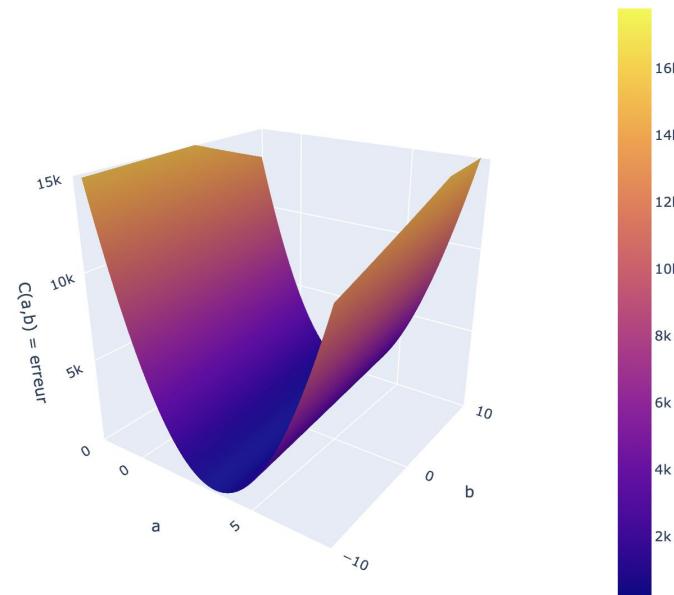
Quel sera alors le nouvel algorithme ?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Erreur en fonction des coefficients a et b : $C(a,b)$

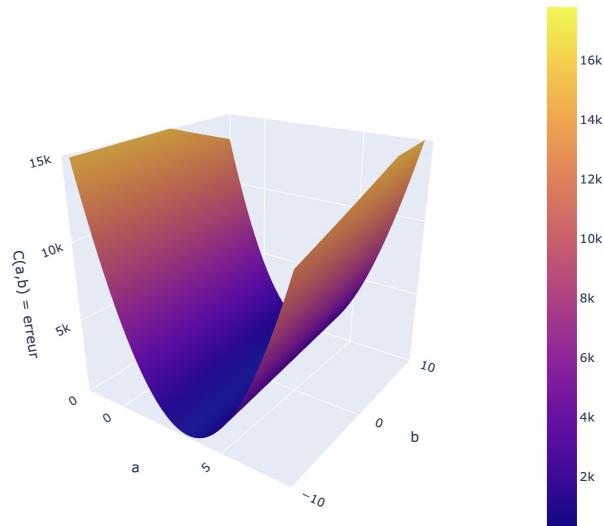


II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Erreur en fonction des coefficients a et b : $C(a,b)$



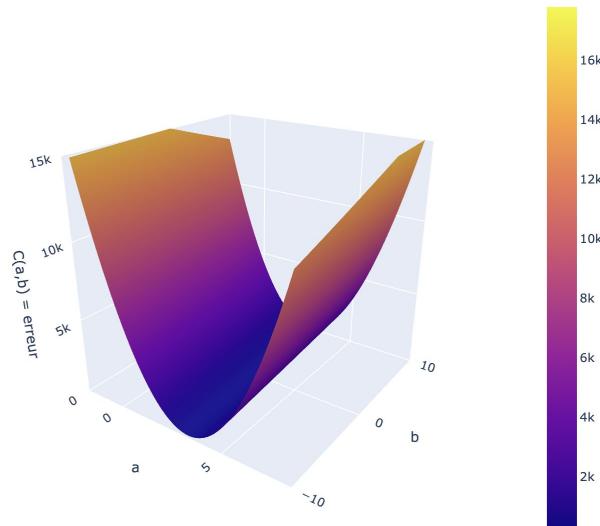
Première conclusion : le paramètre a est beaucoup plus influent que le paramètre b

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

a. Optimisation à deux paramètre

Erreur en fonction des coefficients a et b : $C(a,b)$

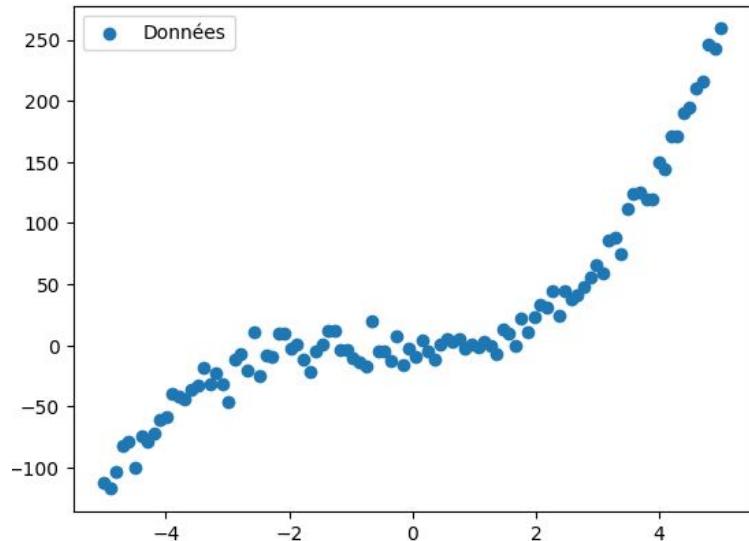


En tout cas, comme on vient de le voir, la descente de gradient est facilement généralisable à 2 paramètres... et même plus !

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

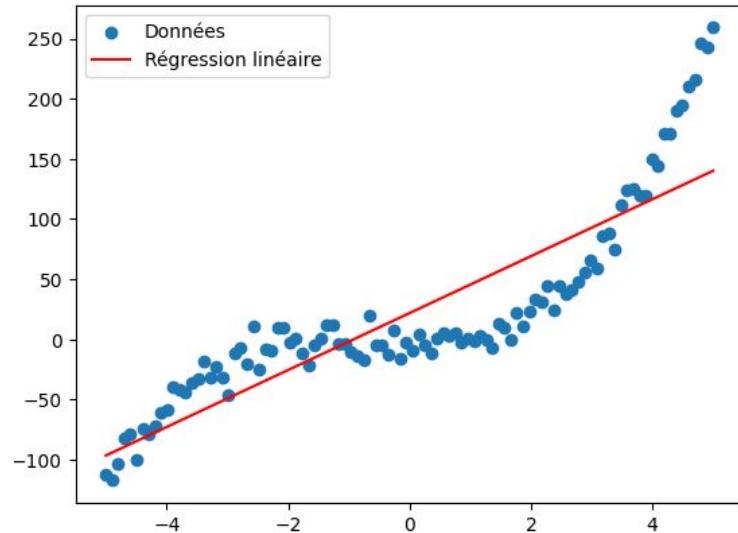


Maintenant, que se passe-t-il avec ce genre de données ?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

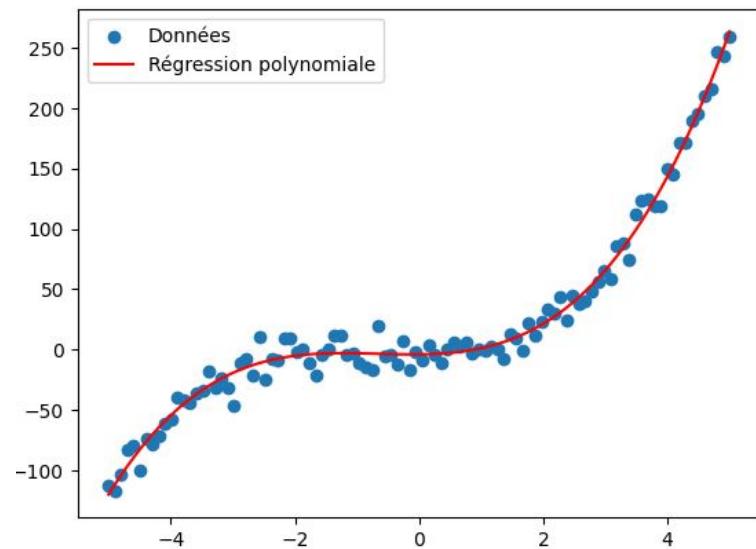


Ici, si on cherche à trouver les coeffs a et b , on aura une droite, et l'erreur sera assez grande ! cela ne fonctionne pas

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

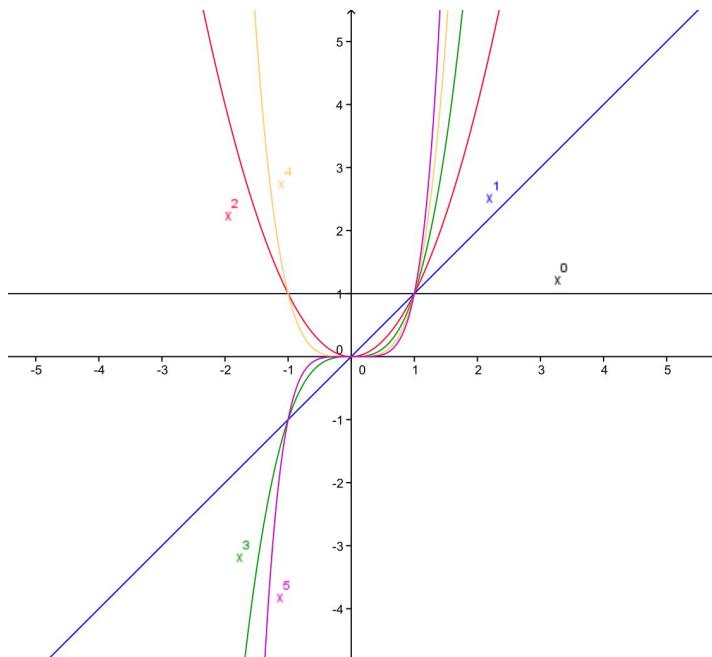


À la place, on peut faire
une régression
polynomiale

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale



Il faut voir ça simplement comme une combinaison linéaire de n fonctions, ou comme **un polynôme de degrès n**

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

Quelle sera la forme de la fonction de prédiction ?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

Quelle sera la forme de la fonction de prédiction ?

$$\hat{f}(x) = a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n + b$$

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

Quelle sera la forme de la fonction de prédiction ?

$$\hat{f}(x) = a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n + b$$

$$\hat{f}(x) = \sum_{i=0}^n a_i x^i + b$$

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

$$\hat{f}(x) = \sum_{i=0}^n a_i x^i + b$$

Cette fois-ci on a donc $n+1$ paramètres à optimiser...

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

b. Régression polynomiale

$$\hat{f}(x) = \sum_{i=0}^n a_i x^i + b$$

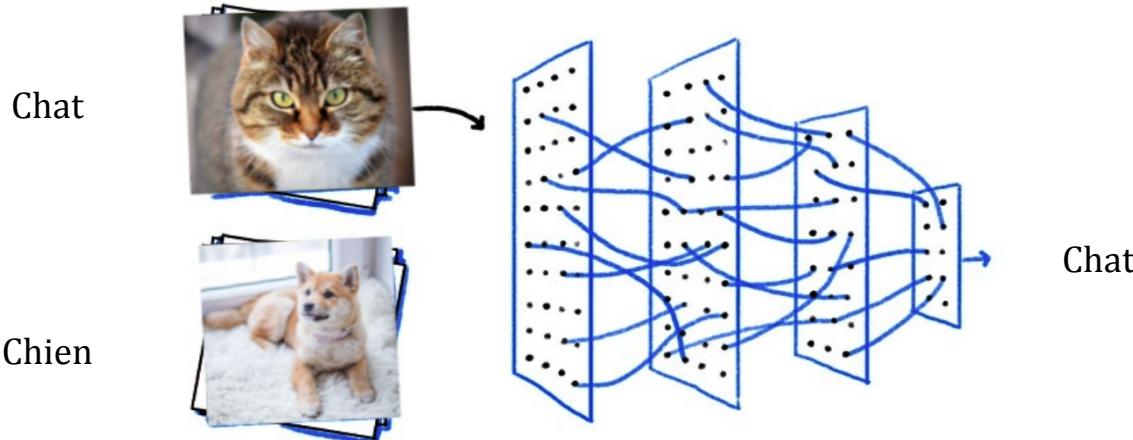
Cette fois-ci on a donc n+1 paramètres à optimiser...

La descente de gradient fonctionne toujours, mais
quelle tête va avoir la fonction d'erreur ?
Et la descente de gradient ?

II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

c. Et si on sort de la régression



À quoi va ressembler la fonction de prédiction ?

Descente de gradient fonctionne toujours ?

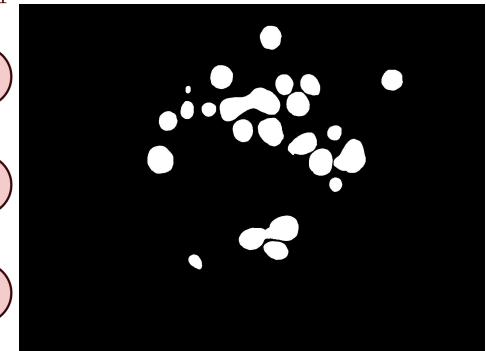
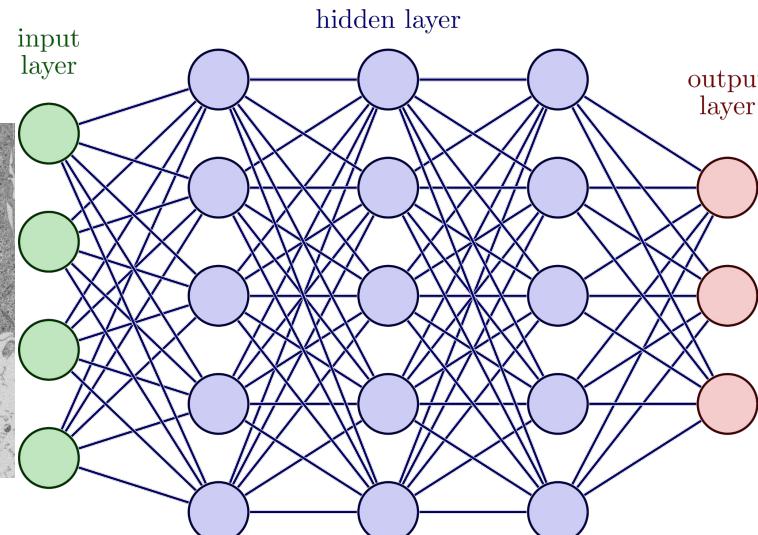
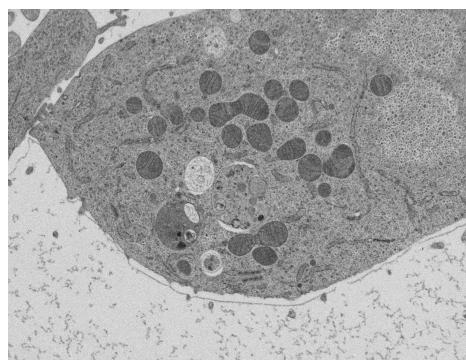
II.1 La Régression Linéaire

D. Généralisation de la notion d'Apprentissage

c. Et si on sort de la régression

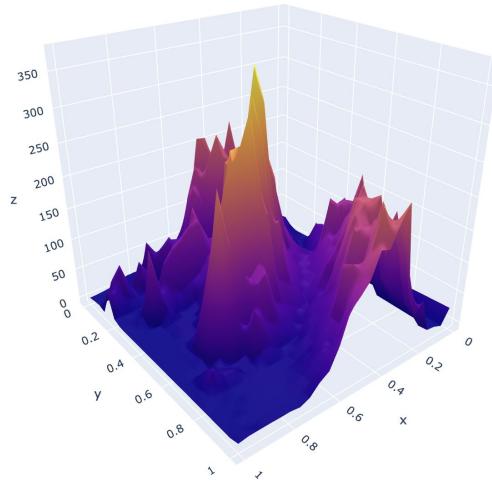
À quoi va ressembler la fonction de prédition ?

Descente de gradient
fonctionne toujours ?



II.1 La Régression Linéaire

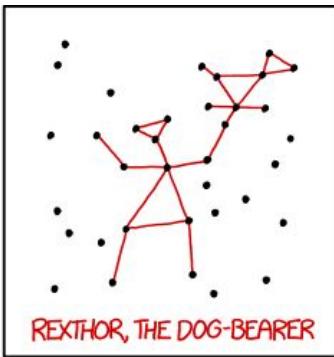
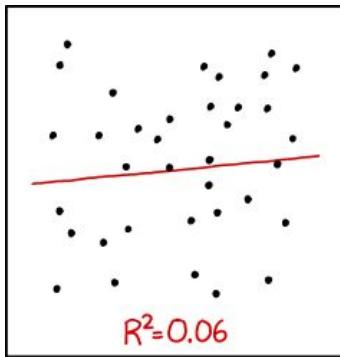
E. Conclusion



La descente de gradient est une méthode générale d'apprentissage, plus spécifiquement d'optimisation de paramètres d'une fonction de prédiction...

Outils très puissant qui va prendre tout son sens lorsque les fonctions de coût à optimiser ne seront plus convexes...

II.2 De la notion de jeu d'entraînement et de test



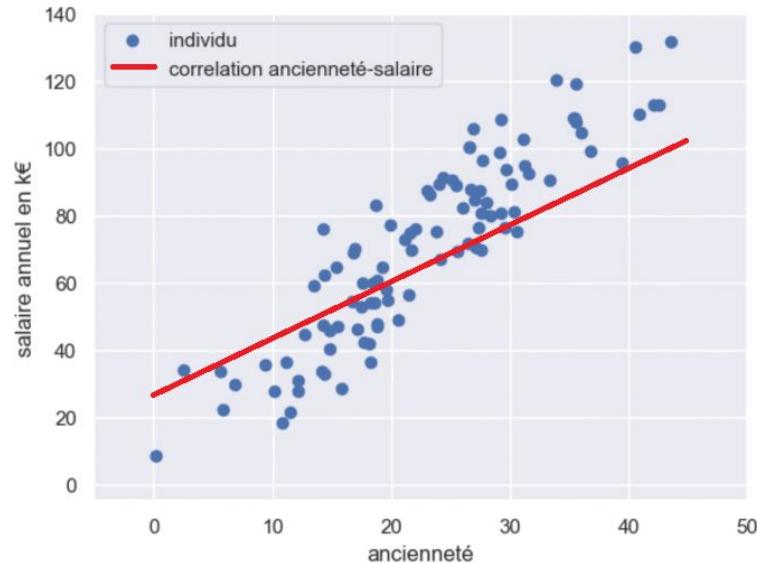
I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER
TO GUESS THE DIRECTION OF THE CORRELATION FROM THE
SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

Objectifs :

- Comprendre la différence entre phase d'entraînement et phase **de test**
- Comprendre les principaux biais et leurs raisons : **sur-apprentissage et sous-apprentissage**

II.2 De la notion de jeu d'entraînement et de test

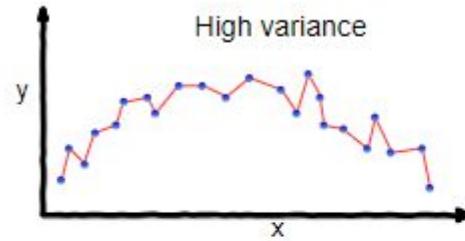
A. Jeu d'entraînement et de test : Quoi et Pourquoi ?



Retour sur notre
régression linéaire

II.2 De la notion de jeu d'entraînement et de test

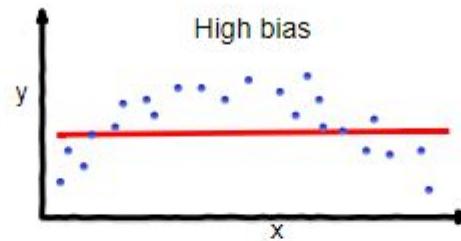
B. Sur-apprentissage (Overfitting)



overfitting

II.2 De la notion de jeu d'entraînement et de test

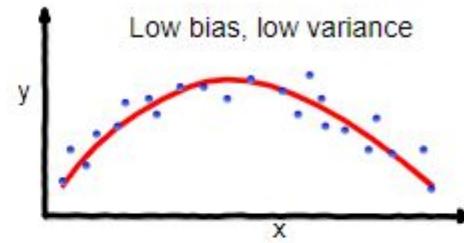
C. Sous-Apprentissage (Underfitting)



underfitting

II.2 De la notion de jeu d'entraînement et de test

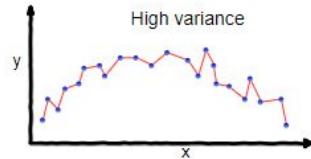
C. Good Fit



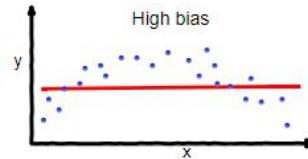
Good balance

II.2 De la notion de jeu d'entraînement et de test

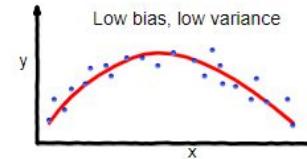
D. Conclusion



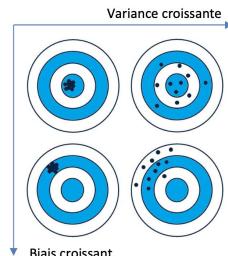
overfitting



underfitting



Good balance



L'humain a un rôle de superviseur de ce que fait l'IA, et un bon ingénieur en IA se doit de savoir diagnostiquer et régler régler ces problèmes

II.3 La Classification par l'algorithme des k plus proches voisins

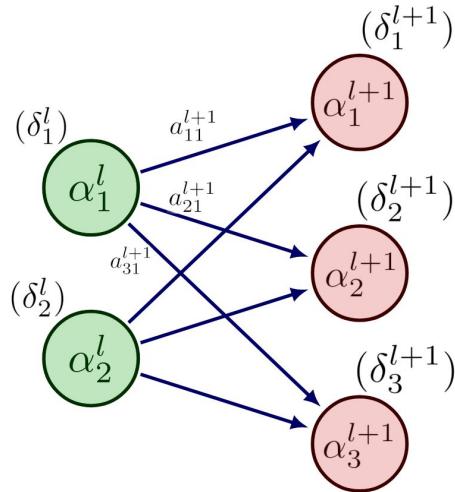
Objectifs :

- Comprendre la notion de **Classification**
- Comprendre l'algorithme des **k plus proches voisins**
- Similarité entre Classification et régression

Retour sur overfitting et underfitting appliqué à la classification ?

Symptoms	Underfitting	Just right	Overfitting
Regression	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Classification			
Deep learning			
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

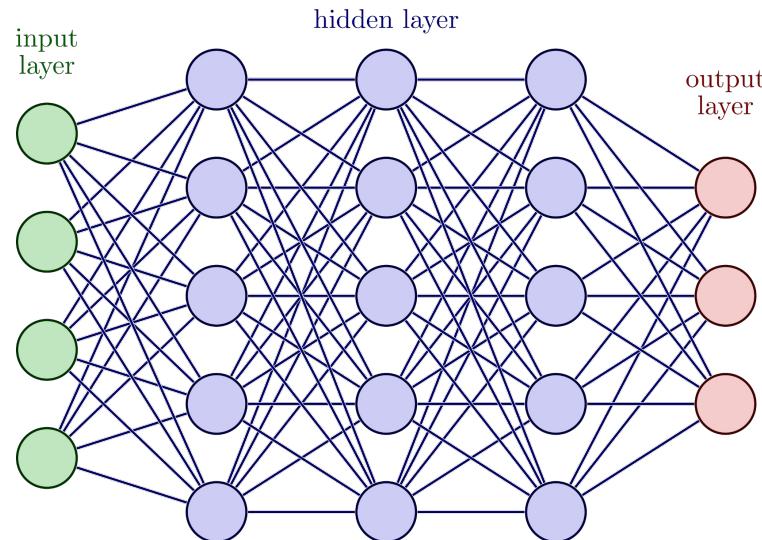
II.4 Introduction aux réseaux de neurones



Objectifs :

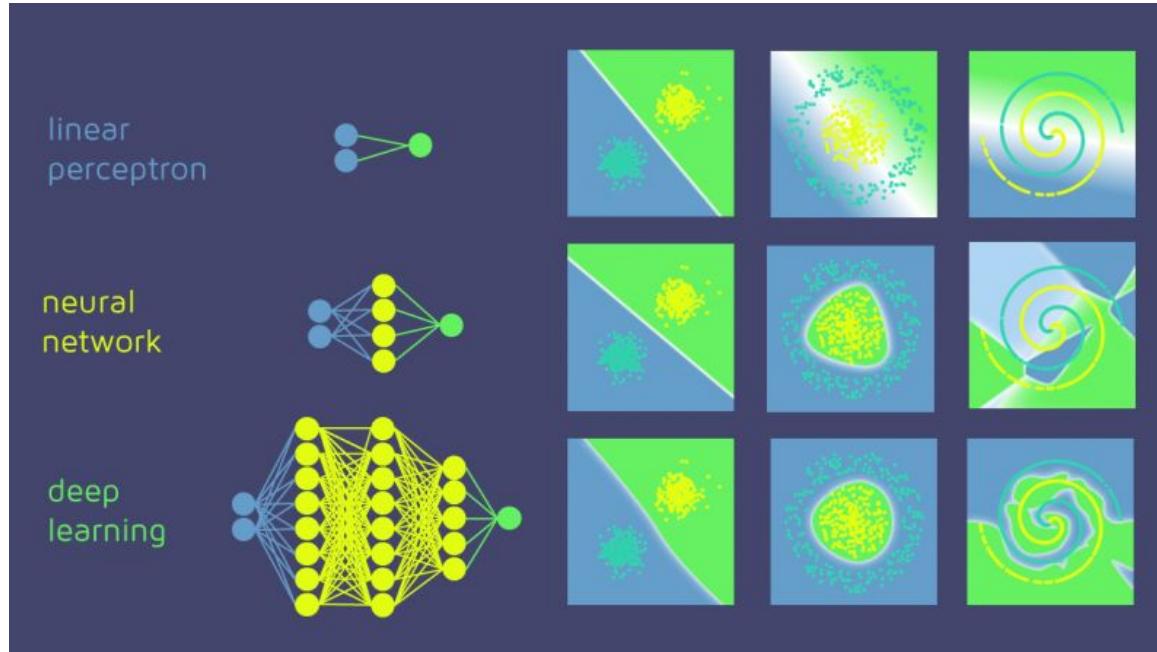
- Comprendre le fonctionnement d'un **réseau de neurones**
- Faire le lien avec la Régression Linéaire
- La descente de gradient via la **backpropagation**

II.4 Introduction aux réseaux de neurones



Qu'est-ce qu'un réseau de neurones ?

II.4 Introduction aux réseaux de neurones

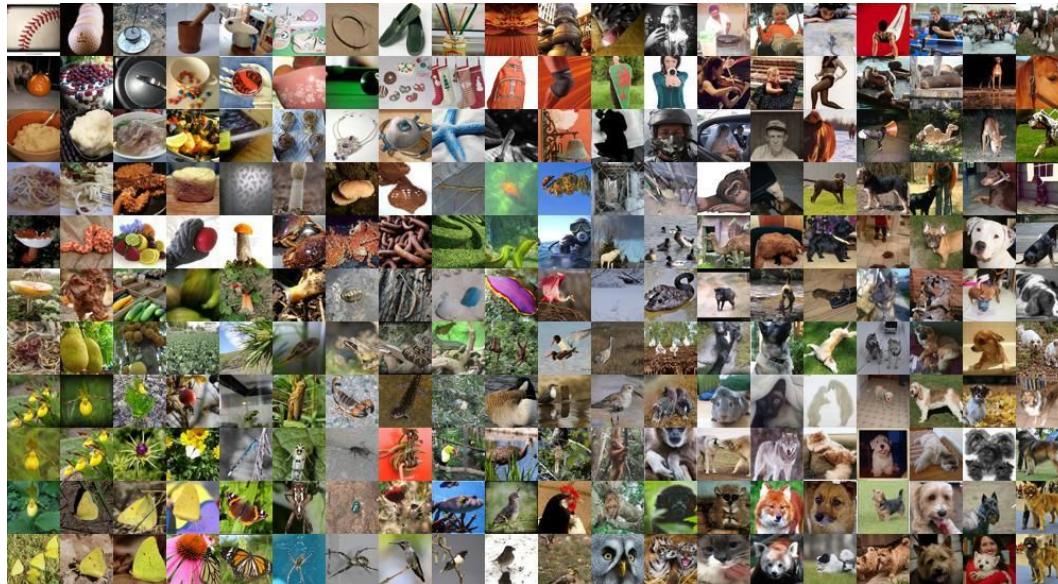


<https://quantdare.com/from-the-neuron-to-the-net/>

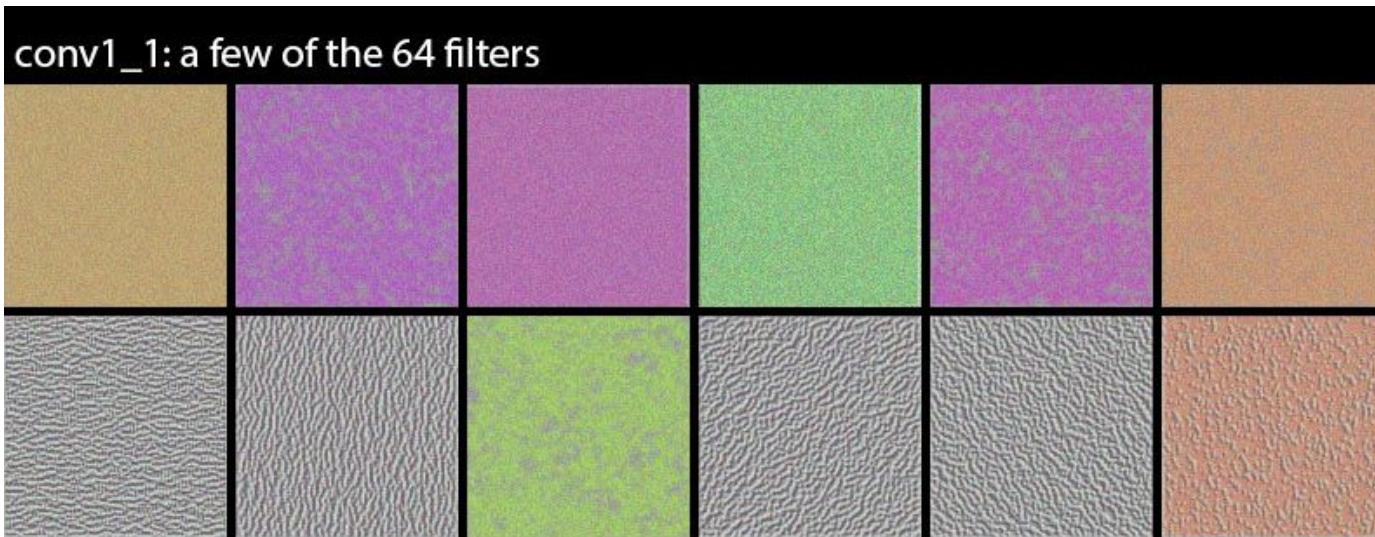
Rappel : l'essence des réseaux de neurones est de pouvoir traiter des problèmes **NON LINÉAIRES**

II.4 Introduction aux réseaux de neurones

Jeu de données typiquement “Non linéaire” : des images

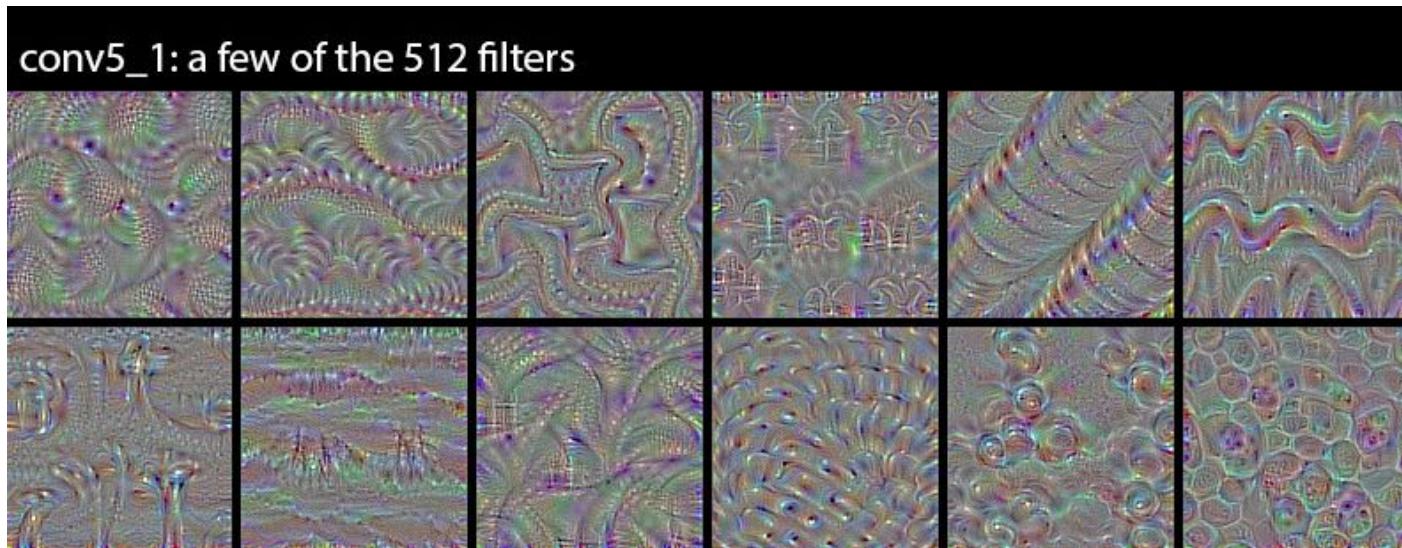


II.4 Introduction aux réseaux de neurones



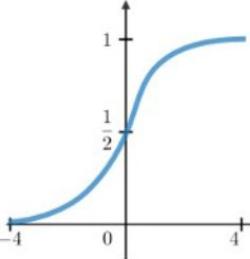
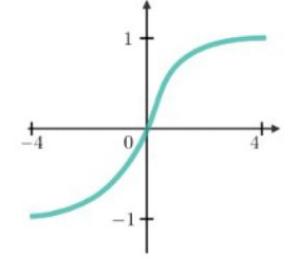
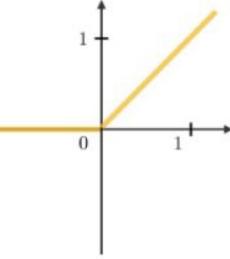
<https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>

II.4 Introduction aux réseaux de neurones



<https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>

II.4 Introduction aux réseaux de neurones

Sigmoïde	tanh	RELU
$\sigma(x) = \frac{1}{1 + e^{-x}}$	$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\sigma(x) = \max(0, x)$
		

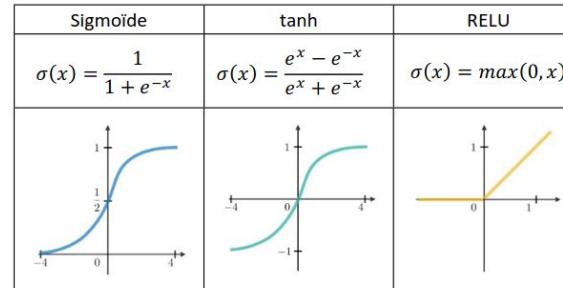
Fonctions d'activations "non linéaires"

II.4 Introduction aux réseaux de neurones

4.1 Notations et définitions

σ « fonction d'activation » non linéaire.
3 fonctions sont utilisées en IA pour jouer ce rôle :

$$\begin{aligned}\sigma : R &\rightarrow R \\ x &\rightarrow \sigma(x)\end{aligned}$$



a_{jk}^l « poids » de la connexion du $k^{\text{ème}}$ neurone de la couche $l - 1$ vers le $j^{\text{ème}}$ neurone de la couche l .

b_j^l « biais » du $j^{\text{ème}}$ neurone de la couche l . Avec $l \in (0, L)$

α_j^l « activation » du $j^{\text{ème}}$ neurone de la couche l , avec
« n » le nombre de neurone à la couche $l - 1$, telle que :

$$\alpha_j^l = \sigma \left(\sum_{k=0}^n a_{jk}^l \alpha_k^{l-1} + b_j^l \right)$$

L'activation d'un neurone de la couche l , est une combinaison linéaire de toutes les activations de la couche précédente, pondérée par les poids a_{jk}^l

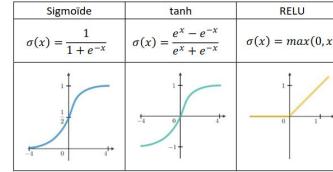
Pour simplifier la suite, on notera : $z_j^l = \sum_{k=0}^n a_{jk}^l \alpha_k^{l-1} + b_j^l$ de sorte que $\alpha_j^l = \sigma(z_j^l)$

II.4 Introduction aux réseaux de neurones

4.1 Notations et définitions

σ « fonction d'activation » non linéaire.
3 fonctions sont utilisées en IA pour jouer ce rôle :

$$\begin{aligned}\sigma : R &\rightarrow R \\ x &\rightarrow \sigma(x)\end{aligned}$$



a_{jk}^l « poids » de la connexion du $k^{\text{ème}}$ neurone de la couche $l - 1$ vers le $j^{\text{ème}}$ neurone de la couche l .

b_j^l « biais » du $j^{\text{ème}}$ neurone de la couche l . Avec $l \in (0, L)$

α_j^l « activation » du $j^{\text{ème}}$ neurone de la couche l , avec
« n » le nombre de neurone à la couche $l - 1$, telle que :

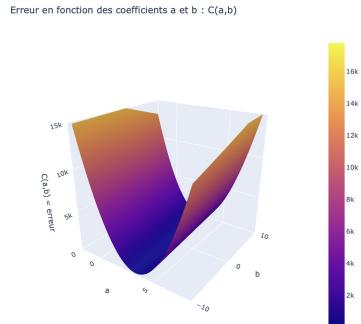
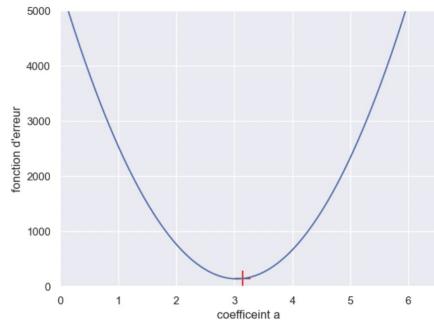
$$\alpha_j^l = \sigma \left(\sum_{k=0}^n a_{jk}^l \alpha_k^{l-1} + b_j^l \right)$$

L'activation d'un neurone de la couche l , est une combinaison linéaire de toutes les activations de la couche précédente, pondérée par les poids a_{jk}^l

Pour simplifier la suite, on notera : $z_j^l = \sum_{k=0}^n a_{jk}^l \alpha_k^{l-1} + b_j^l$ de sorte que $\alpha_j^l = \sigma(z_j^l)$

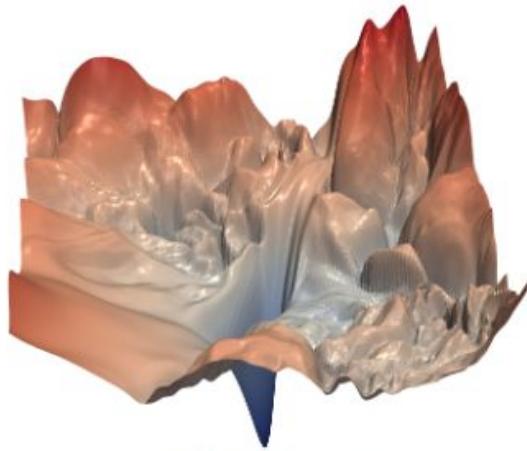
Et maintenant, à vos feuilles et stylos !

II.4 Introduction aux réseaux de neurones

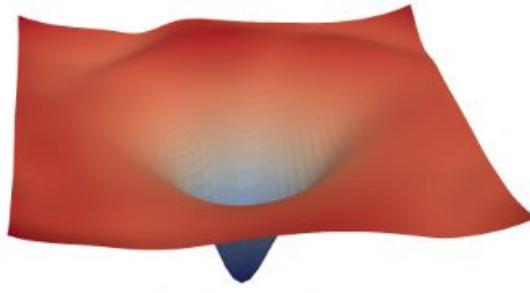


Par rapport aux
précédentes fonctions de
coût... on est en dimension
 $N \gg 0$

II.4 Introduction aux réseaux de neurones



(a) without skip connections

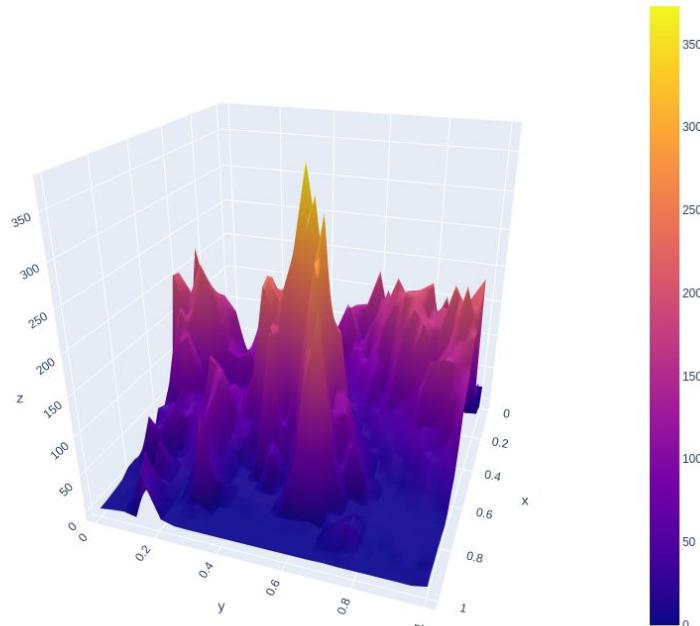


(b) with skip connections

Néanmoins, il existe plusieurs méthodes pour visualiser tout de même la fonction de coût (cf <https://arxiv.org/pdf/1712.09913.pdf>)

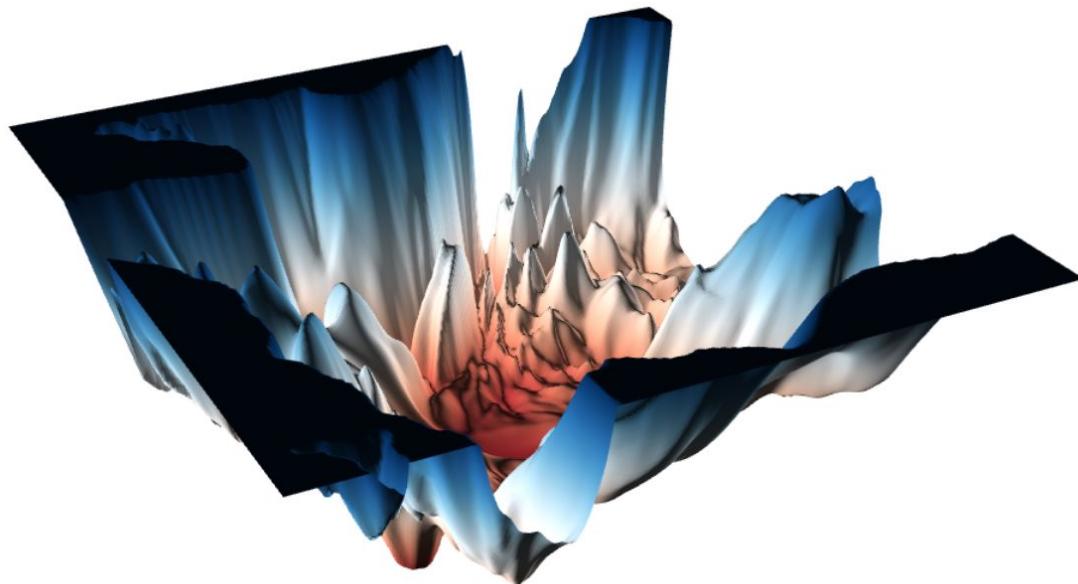
II.4 Introduction aux réseaux de neurones

Fonction d'erreur non convexe d'un réseau de neurone



On comprend vraiment l'aspect important de la descente de gradient : Il y a plein de minima quasi équivalents

II.4 Introduction aux réseaux de neurones



Voir aussi :

<https://www.telesens.co/loss-landscape-viz/viewer.html>

III. Quelques notions supplémentaires

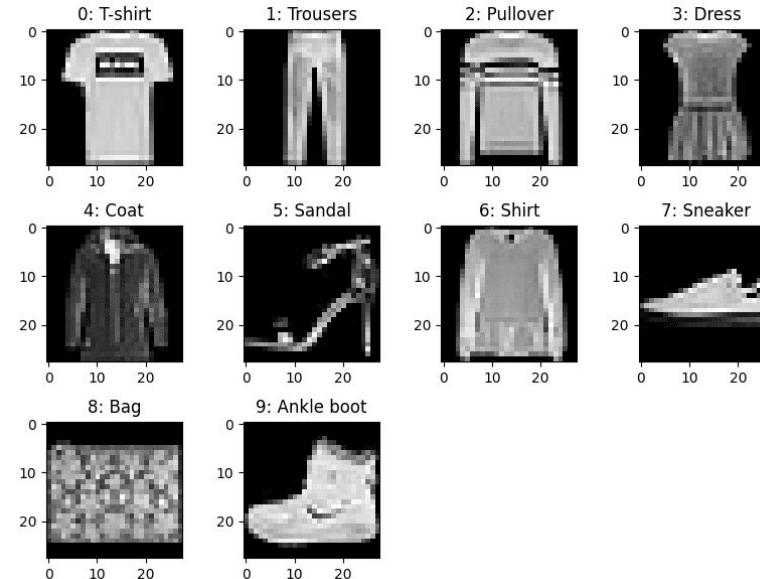
Objectifs :

- Matrice de confusion
- Distinction **Données d'entraînement/Données de test**

III. Quelques notions supplémentaires

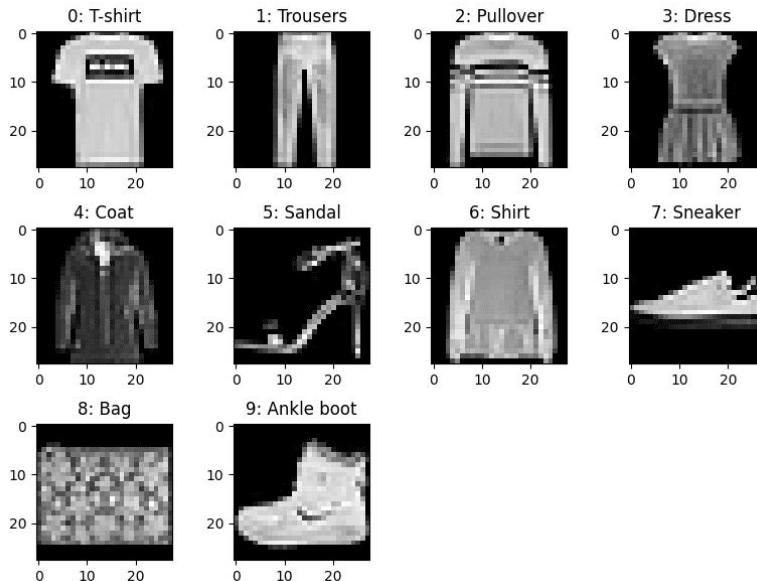
1. La matrice de confusion

Exemple : le jeu de données Fashion-MNIST



III. Quelques notions supplémentaires

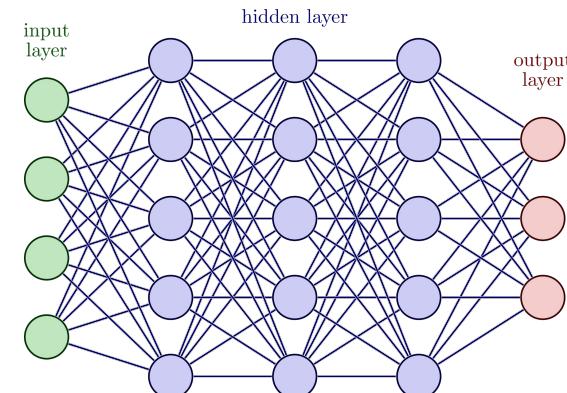
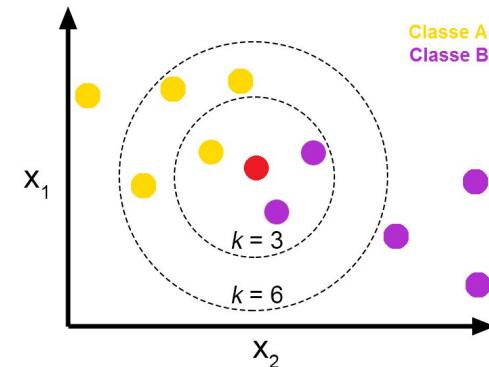
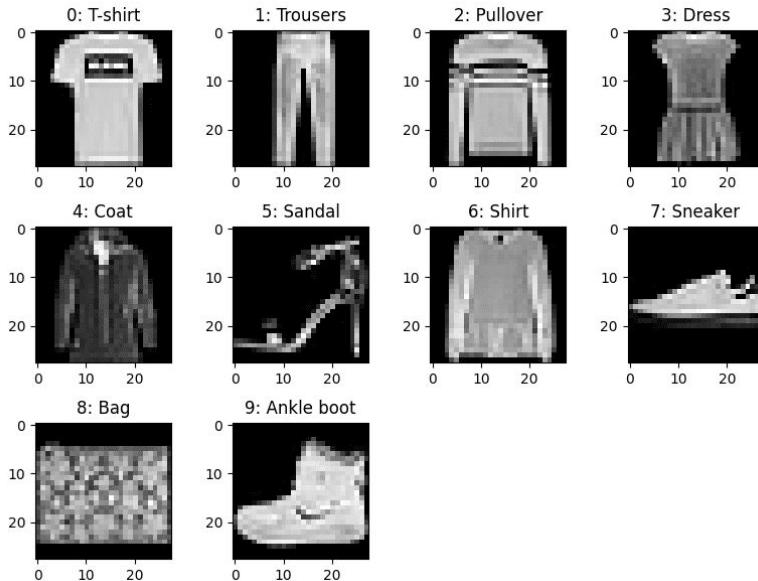
1. La matrice de confusion



Question : Quel modèle utiliser ?

III. Quelques notions supplémentaires

1. La matrice de confusion

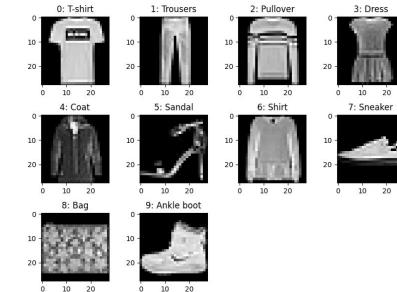


III. Quelques notions supplémentaires

1. La matrice de confusion

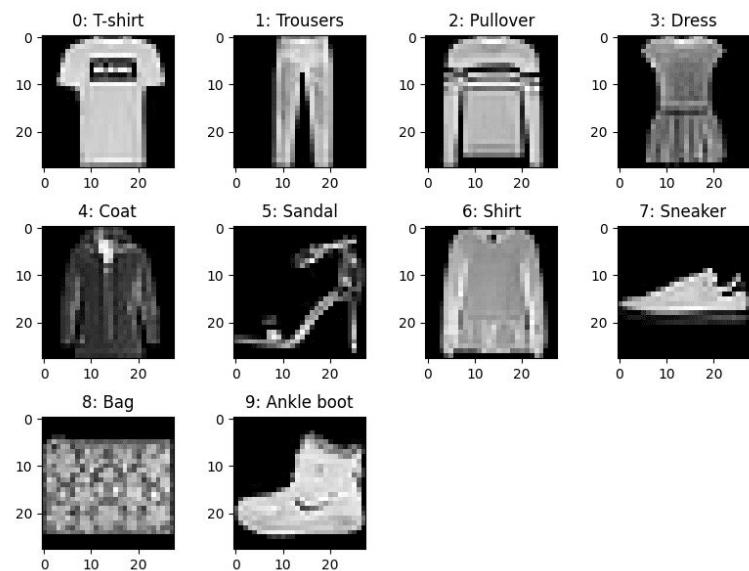
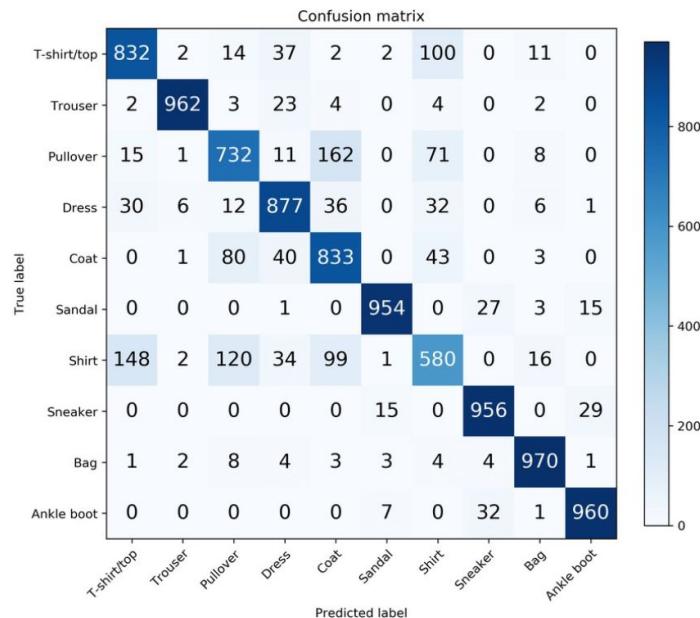
Exemple : le jeu de données Fashion-MNIST

- 70.000 images de taille 64x64 px
- Séparation en :
 - > 60.000 images pour entraîner le modèle
 - > 10.000 images pour tester le modèle



III. Quelques notions supplémentaires

1. La matrice de confusion



IV. Et pour les concours ?

Objectifs :

- Cerner les points clés
- Essayer d'anticiper les éventuelles questions ?

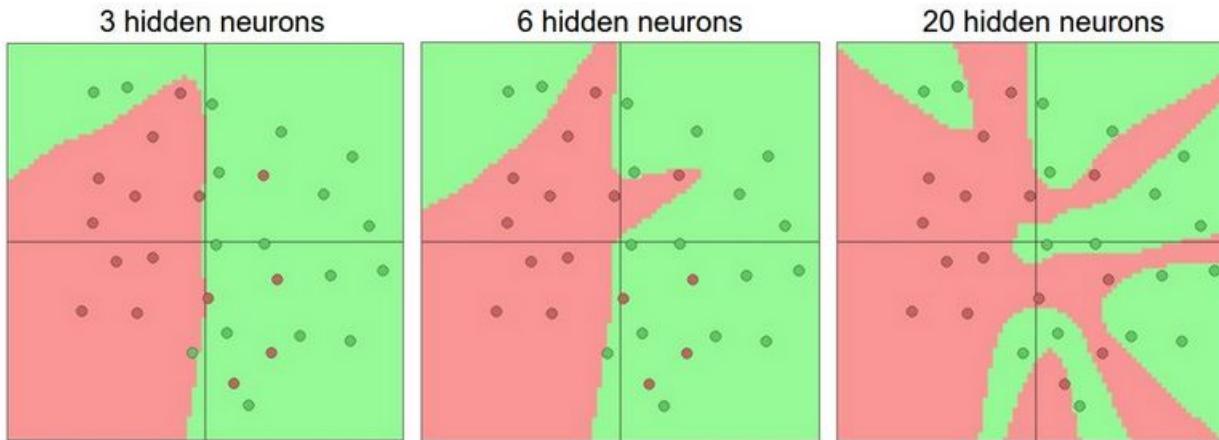
V. Pour aller plus loin, quelques remarques générales

Objectifs :

- Avoir un regard critique sur l'IA
- Prendre le recul nécessaire pour mieux comprendre

Pour finir

Notion de non linéarité...



Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath. You can play with these examples in this [ConvNetsJS demo](#).