

Intelliaence Artificielle Machine Learning

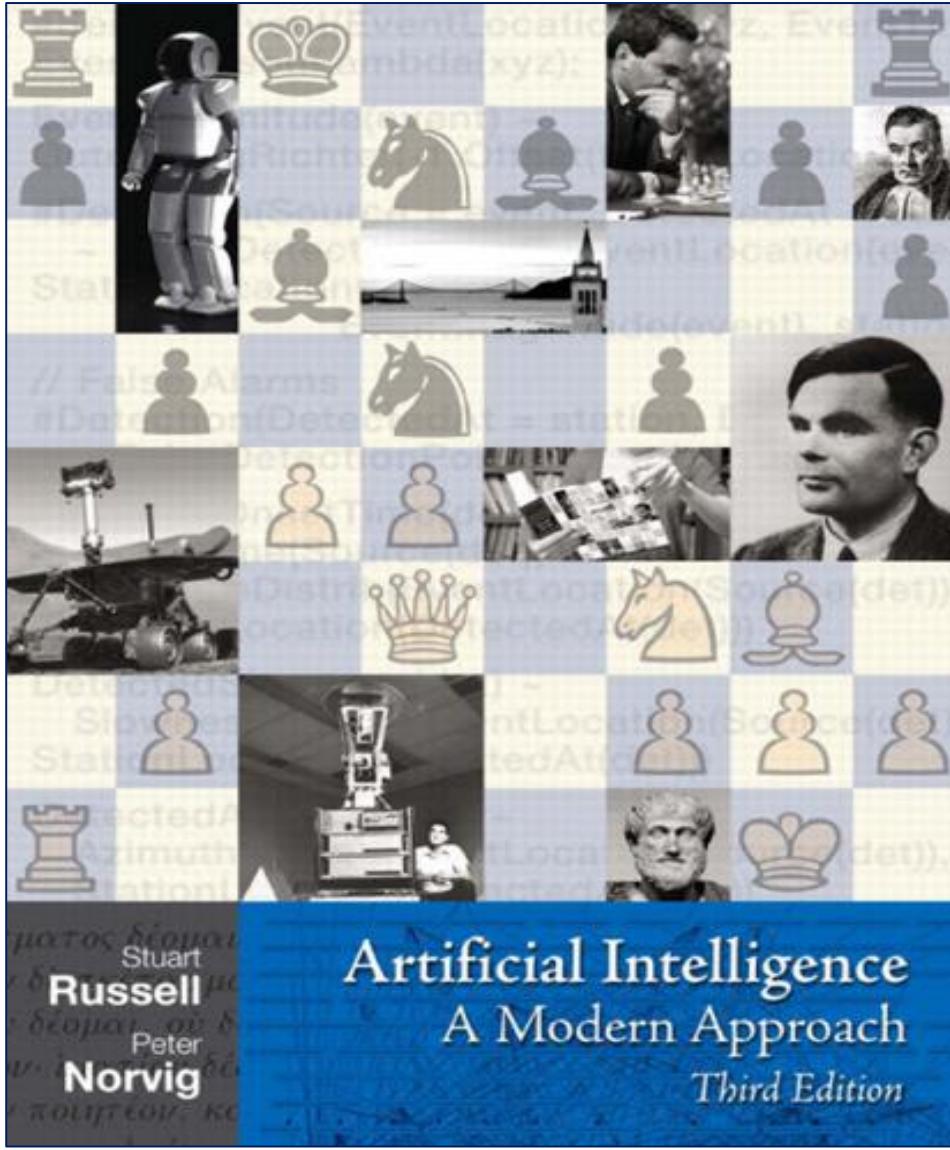
AI



Abdelaziz Sahbani

abdelazizsahbani@yahoo.fr

Textbooks



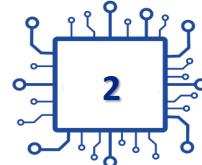
O'REILLY®

Hands-On Machine Learning with Scikit-Learn & TensorFlow

CONCEPTS, TOOLS, AND TECHNIQUES
TO BUILD INTELLIGENT SYSTEMS



Aurélien Géron



Contacts

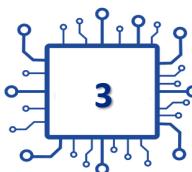
- Google Classroom :

- Espace de cours : Intelligence Artificielle et Machine Learning
- Code : **tl3fquj**

<https://classroom.google.com/c/NjMyODUxOTgzNzEz?cjc=tl3fquj>

- Emails :

- abdelaziz.sahbani@fsb.ucar.tn



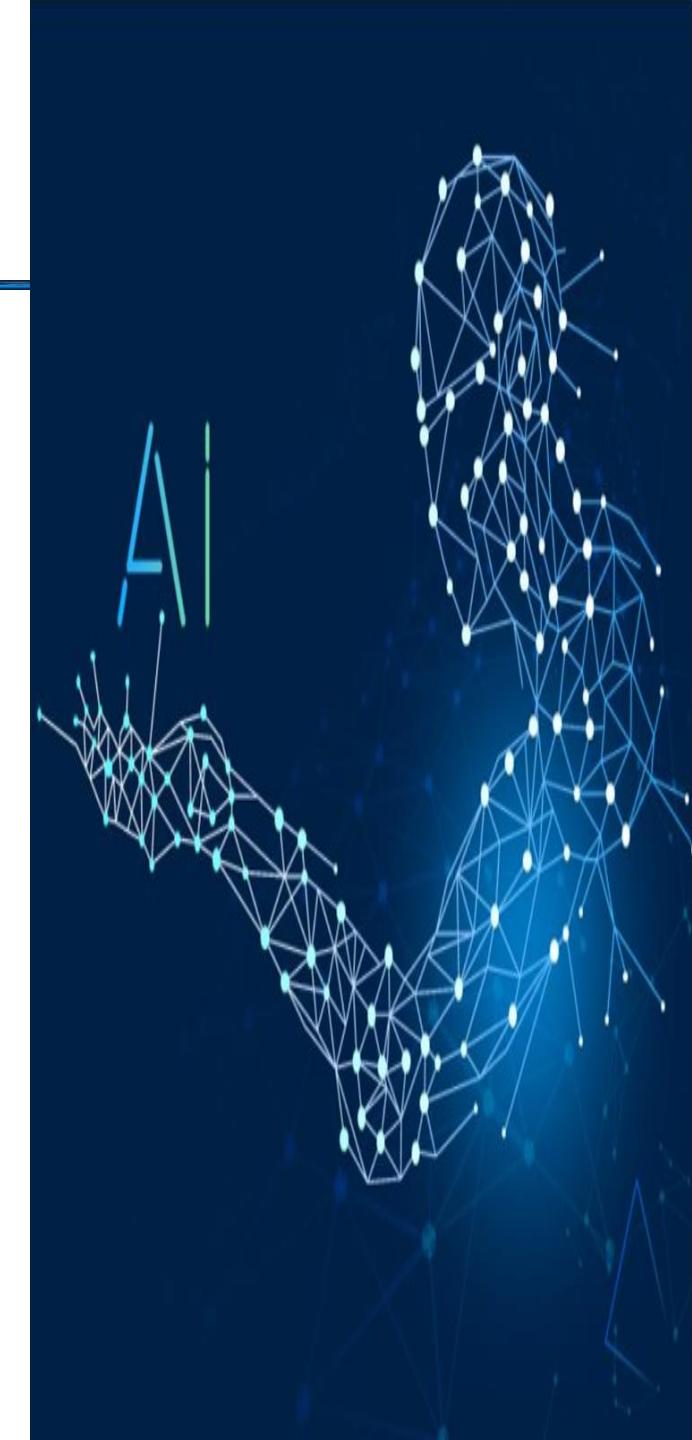
Plan du module

Partie 1 : Fondements de l'Intelligence Artificielle

- Introduction à l'Intelligence Artificielle
- Représentation des connaissances et systèmes experts
- Les agents intelligents
- Les algorithmes de recherche et de jeux

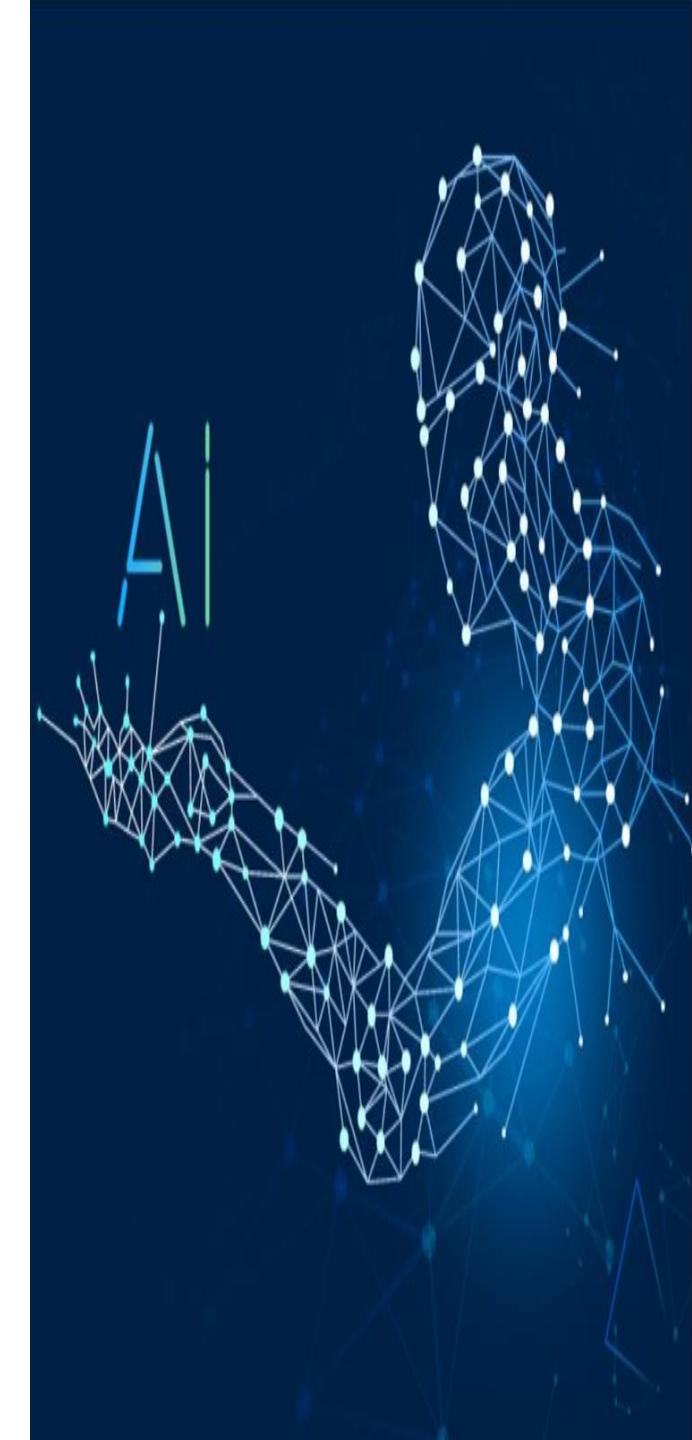
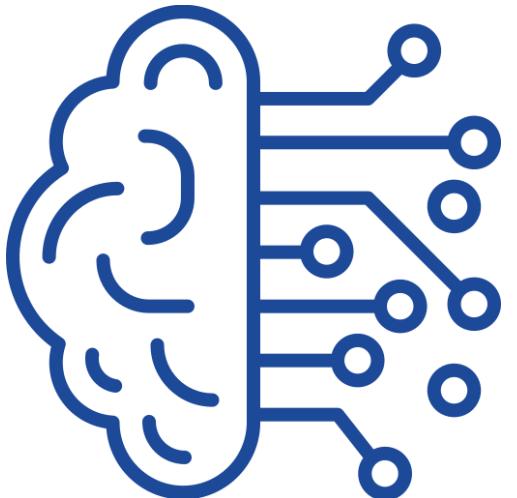
Partie 2 : Machine Learning

- Introduction à la Machine learning
- Apprentissage Supervisé
 - La régression
 - La classification
- Apprentissage non supervisé
- Les réseaux de neurones



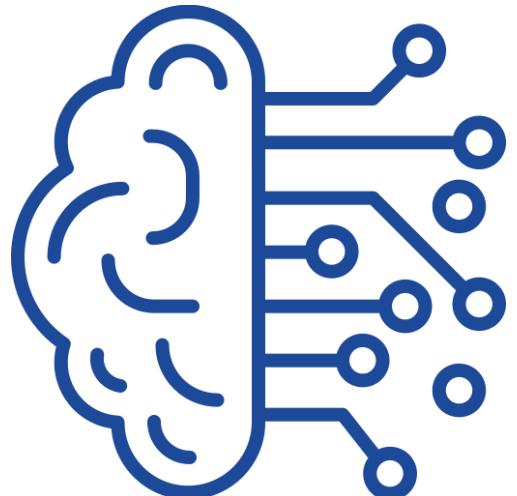
Partie 2 :

Machine Learning



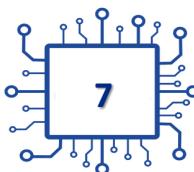
Chapitre 1

Introduction à la Machine learning



Plan du chapitre

- Définitions de machine learning
- Types d'algorithme d'apprentissage
- Concepts fondamentaux en ML



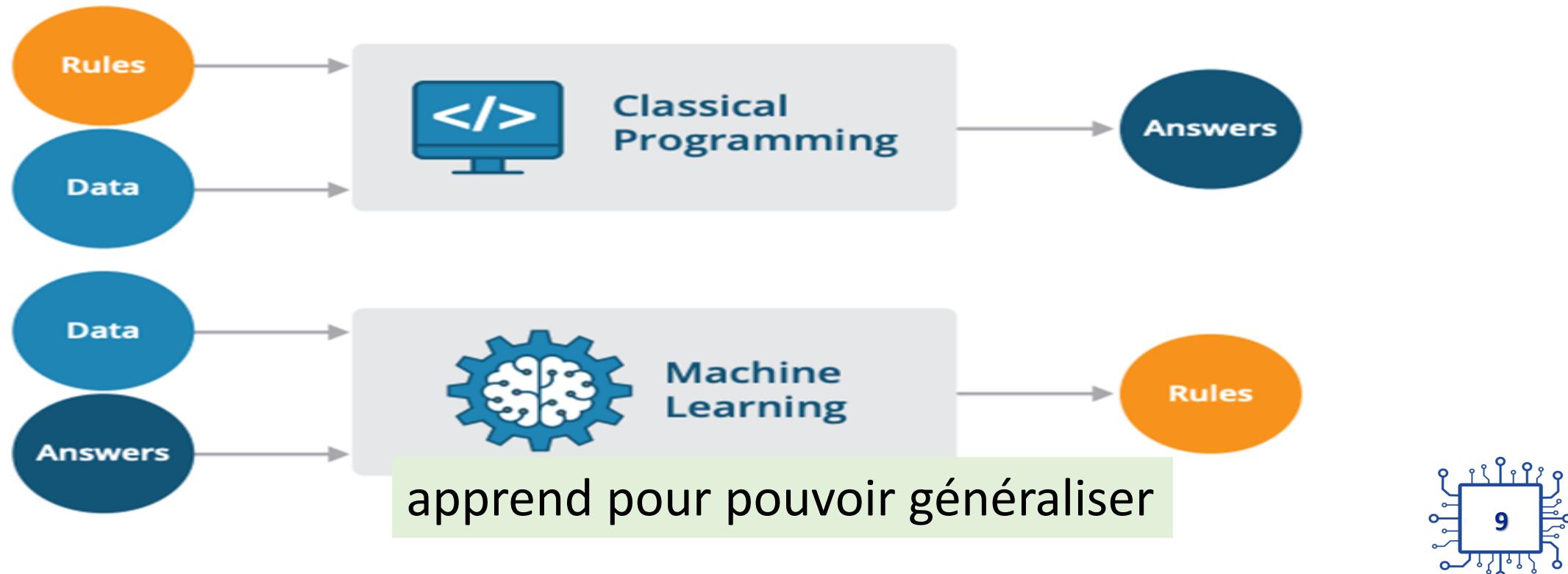
Objectifs

- Définir le machine learning
- Identifier si un problème relève ou non du machine learning
- Donner des exemples de cas concrets relevant de grandes classes de problèmes de machine learning.



Qu'est-ce que le machine learning ?

- Machine learning = apprentissage automatique,
- **Arthur Samuel (1959)** : Donner aux ordinateurs la capacité d'apprendre et à accomplir des tâches sans être explicitement programmé.

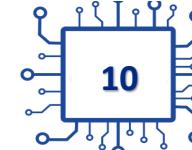
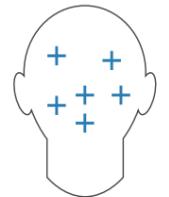


Quand utiliser le machine learning ?

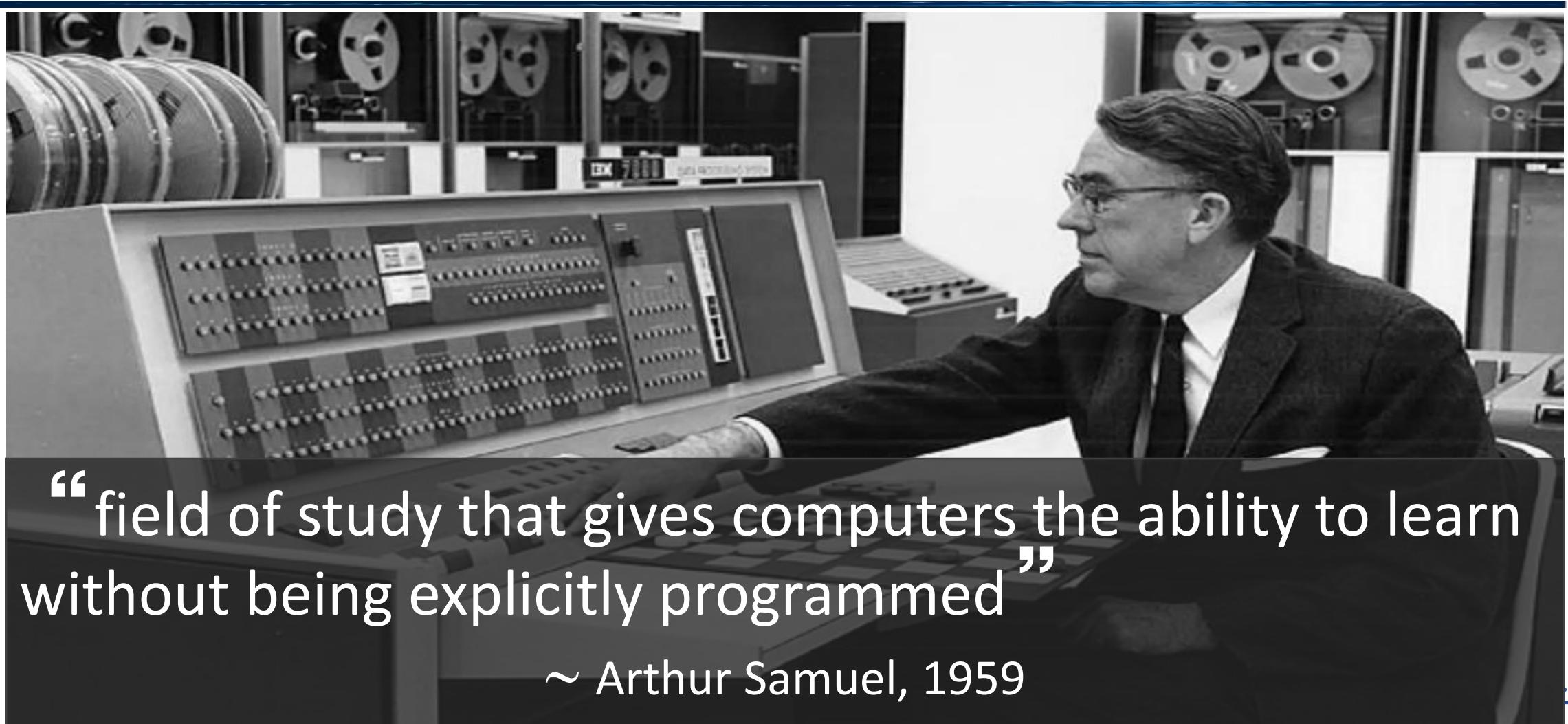
Utiliser l'apprentissage automatique lorsque on est confronté à une tâche ou à un problème complexe impliquant **une grande quantité de données** et de nombreuses variables, sans aucune règle ou équation connue.

Par exemple :

- Règles liant les sorties aux entrées sont souvent inconnues : cas de la reconnaissance faciale, reconnaissance de la parole,...
- Les règles liant les sorties aux entrées changent constamment : cas de la détection de fraude
- La nature des données ne cesse de changer → le programme doit s'adapter : cas de la prévision de la demande d'énergie, prédition du prix du pétrole, ...



Définition du machine learning (1/2)



“field of study that gives computers the ability to learn without being explicitly programmed”

~ Arthur Samuel, 1959

Définition du machine learning (2/2)



“ Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience.

~ Tom Mitchell,
Machine Learning, McGraw Hill, 1997

Carnegie Mellon University
Machine Learning



Un programme apprend lors d'une expérience E par rapport à une tâche T, si sa performance sur T, mesurée par P, s'améliore avec l'expérience E



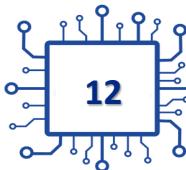
Experience in playing chess → E



Sometimes you win
Sometimes you lose → T



Performance measure → P



Ingrédients du machine learning

- Le machine learning repose sur deux piliers fondamentaux :
 - les données, qui sont les exemples à partir duquel l'algorithme va apprendre
 - l'algorithme d'apprentissage, qui est la procédure que l'on fait tourner sur ces données pour produire un modèle.
- Aucun algorithme d'apprentissage ne pourra créer un bon modèle à partir de données qui ne sont pas pertinentes :
 - garbage in, garbage out : un algorithme d'apprentissage auquel on fournit des données de mauvaise qualité va faire des prédictions de mauvaise qualité.
- Un modèle appris avec un algorithme inadapté sur des données pertinentes ne pourra pas être de bonne qualité

Types d'algorithme d'apprentissage

- Les systèmes du machine learning peuvent être classés en fonction du type de **supervision** dont ils bénéficient pendant l'apprentissage. Il existe quatre grandes catégories :

- Apprentissage supervisé
- Apprentissage non supervisé,
- Apprentissage semi-supervisé,
- Apprentissage par renforcement

SUPERVISED
LEARNING



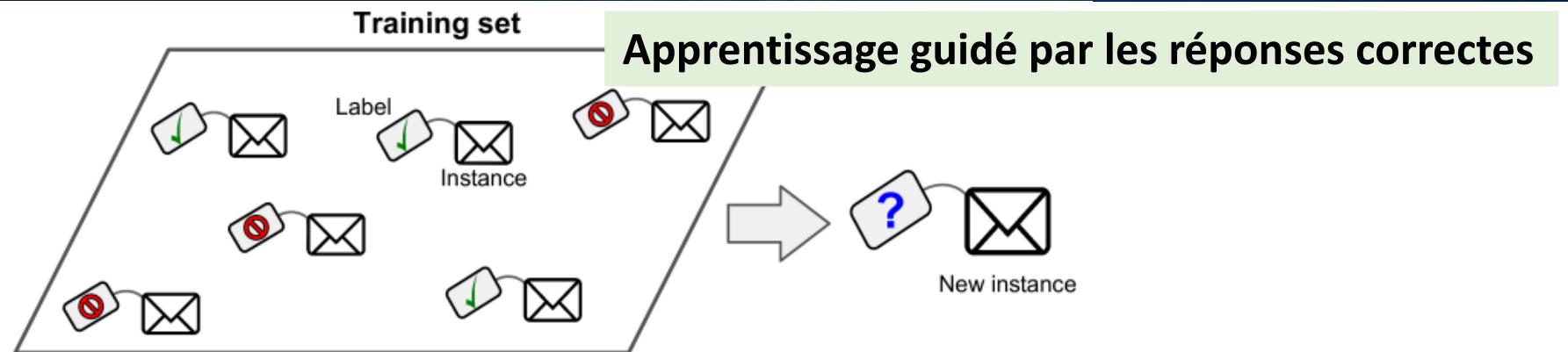
UNSUPERVISED
LEARNING



REINFORCEMENT
LEARNING



Apprentissage supervisé (1/4)



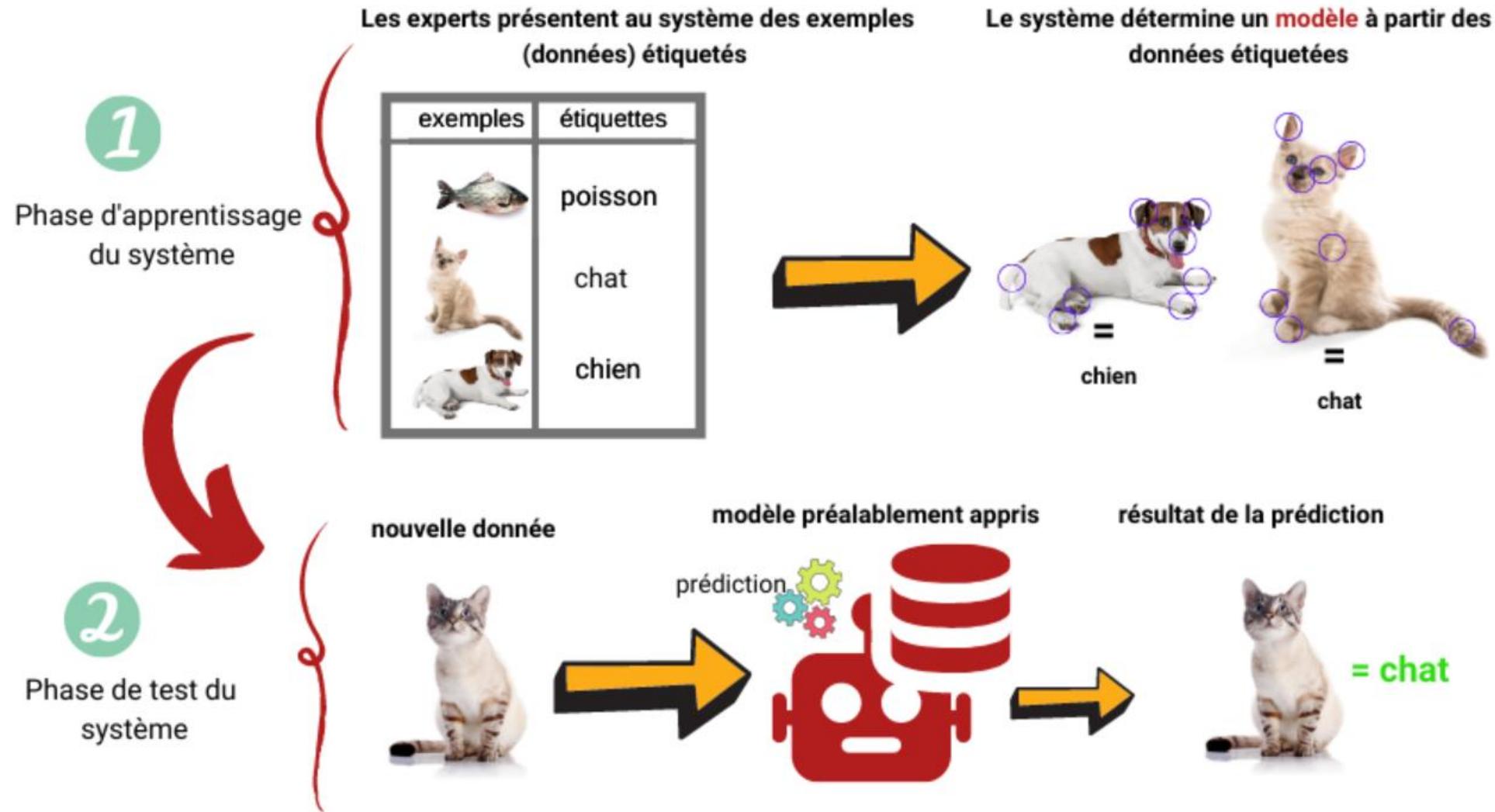
- En apprentissage supervisé, **les données d'apprentissage** introduites à l'algorithme **contiennent les solutions désirées**, nommées étiquettes ou **labels** :
 $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$] **les données d'apprentissage D**
- La sortie désirée ou cible aura une représentation différente selon le problème à résoudre:
 - Dans un problème de classification en C classes : sortie = valeur discrète (index de 0 à C-1)
 - Dans un problème de régression : sortie = valeur réelle ou continue

Apprentissage supervisé (2/4)

- Une tâche d'apprentissage supervisé typique est la **classification**
- La classification prédit des classes = catégories
- Exemples :
 - Détection Spam / not spam dans les emails
 - La classification Cancer / not cancer dans les images médicales
 - La classification Fake news / not fake news dans les journaux
 - La reconnaissance des chiffres manuscrites 0-9
 - La classification chien / chat

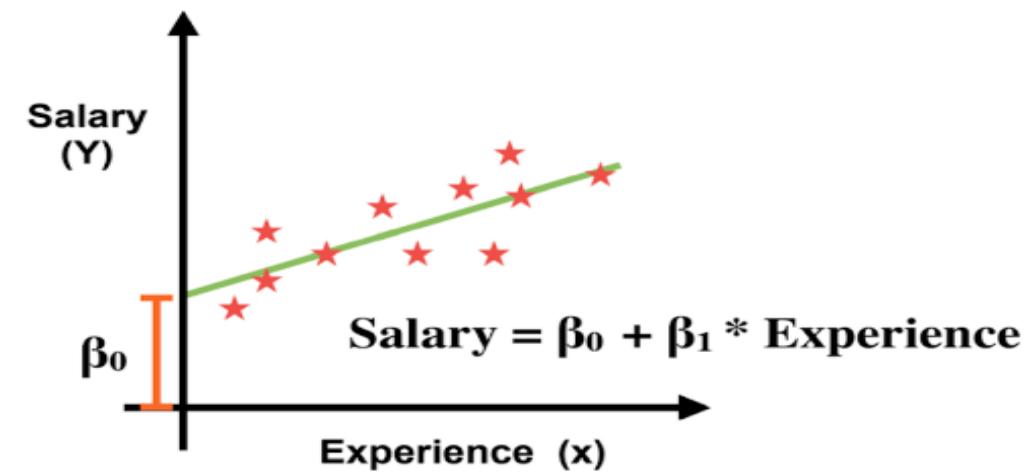
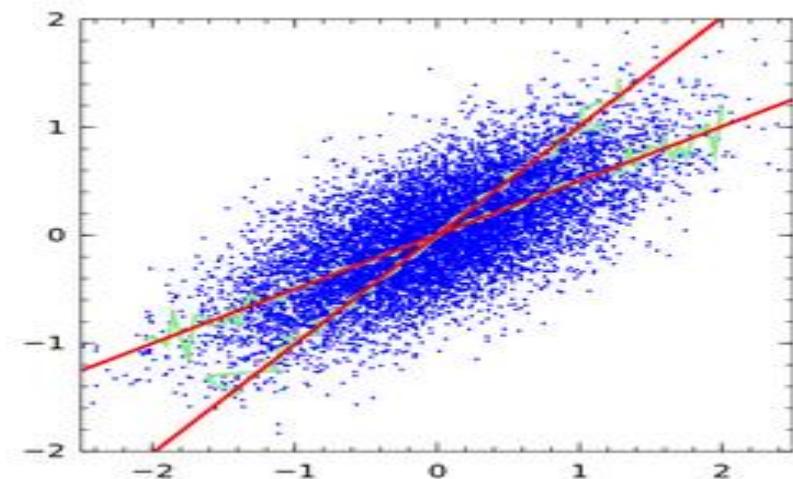
Apprentissage supervisé (3/4)

La classification chien / chat



Apprentissage supervisé (4/4)

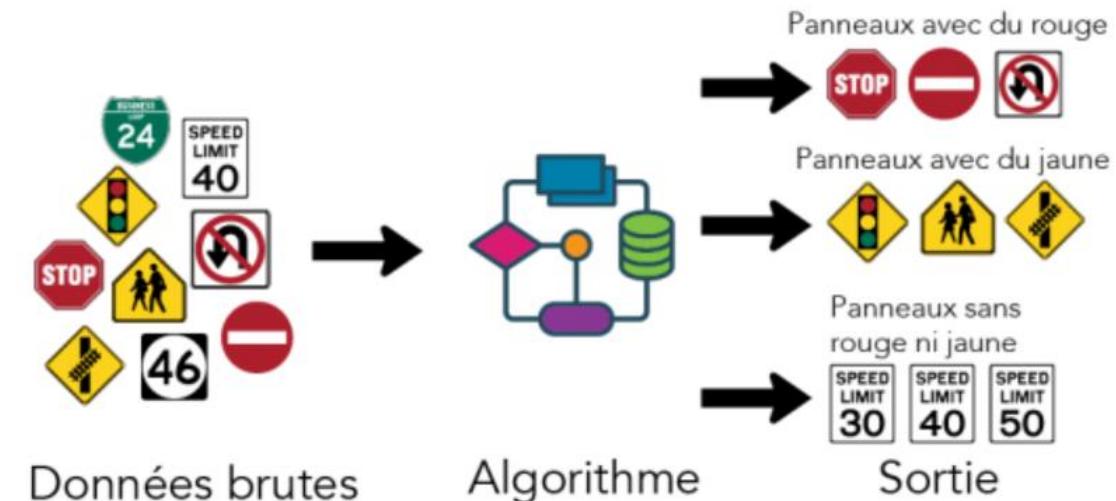
- Une autre tâche typique consiste à prédire une valeur numérique cible, telle que le prix d'une maison, en fonction d'un ensemble de caractéristiques (taille, pièces, etc.) appelées prédicteurs : la régression.
- Pour entraîner le système, il faut lui donner de nombreux exemples de maisons, incluant à la fois leurs prédicteurs (taille par exemple) et leurs étiquettes (prix).



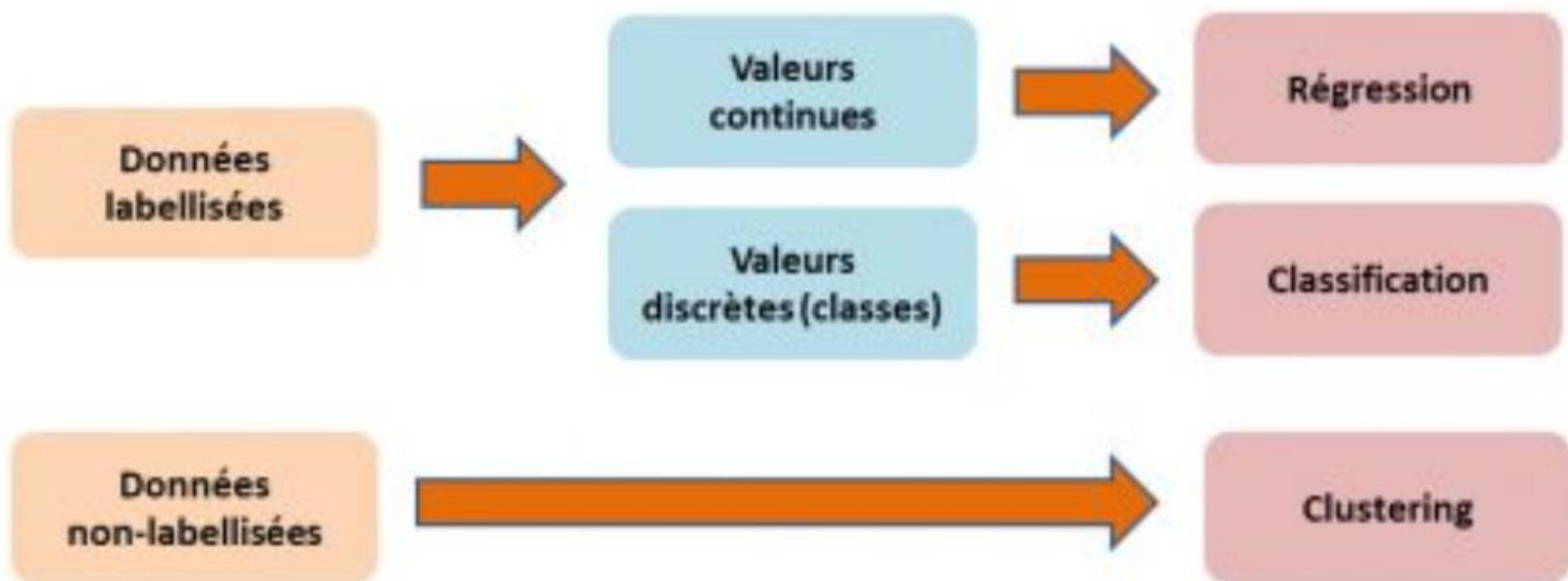
Apprentissage non supervisé

- L'apprentissage non supervisé (clustering) est une branche de la machine learning caractérisée par l'analyse et le regroupement de données **non-étiquetées**.
- (x_1, x_2, \dots, x_N) : base de données D sans labels

- Le système essaie d'apprendre en cherchant des propriétés ou des caractéristiques **internes** dans les données pour les distinguer,
- Les données sont regroupées en fonction de leurs similarités et de leurs différences, dans des groupes (ou des clusters) de sorte que :
 - les éléments au sein d'un même groupe soient très similaires,
 - les éléments dans des groupes différents soient très différents



A retenir



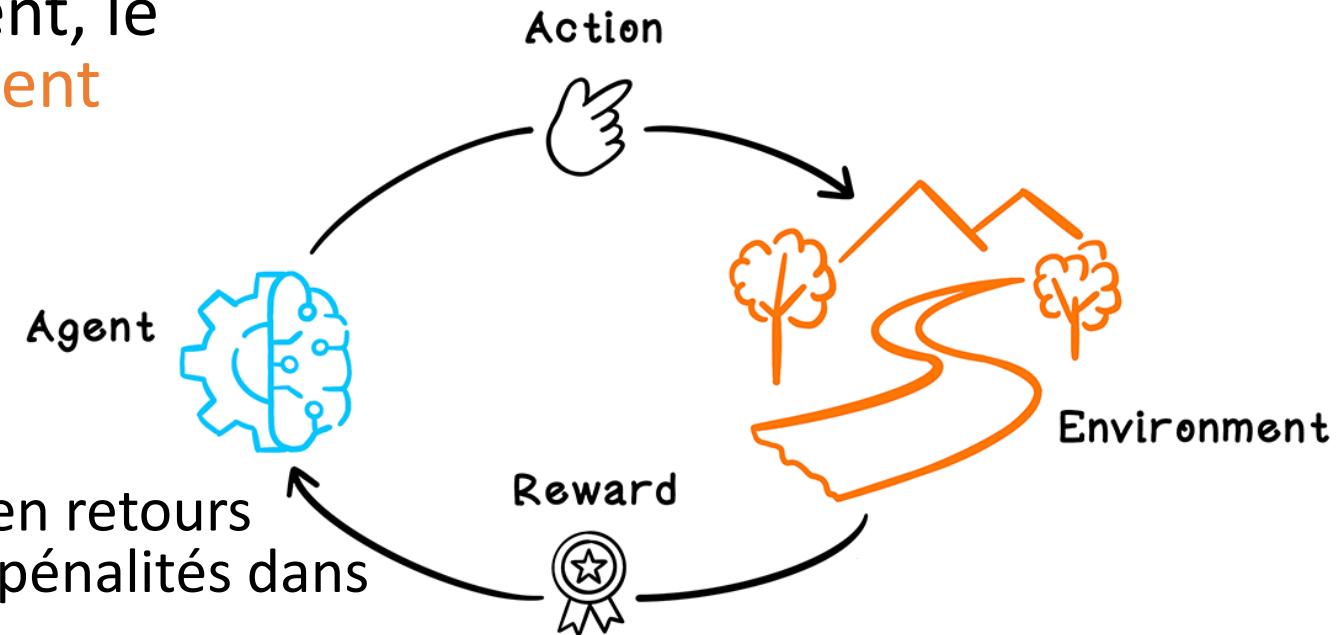
Apprentissage Semi-supervisé

- Dans de nombreuses applications, nous trouvons des bases de données contenant beaucoup de données non étiquetées et peu de données étiquetées → Il faut **combiner des algorithmes supervisés et non supervisés** pour profiter de toutes les données.
- Les algorithmes semi-supervisés sont en général une combinaisons d'algorithmes supervisés et non supervisés
- Exemple : Google Photos
 - Commence par le groupement des photos similaires (unlabeled data)
 - Une fois quelqu'un identifié une photo, le système donne la même étiquette à toutes les photos similaires

Apprentissage par Renforcement

- En apprentissage par renforcement, le système apprenant est appelé **agent**

- L'agent est en train de :
 - observer l'environnement,
 - prendre des actions,
 - avoir des récompenses (rewards) en retours lorsque les actions sont bonnes et pénalités dans le cas contraire



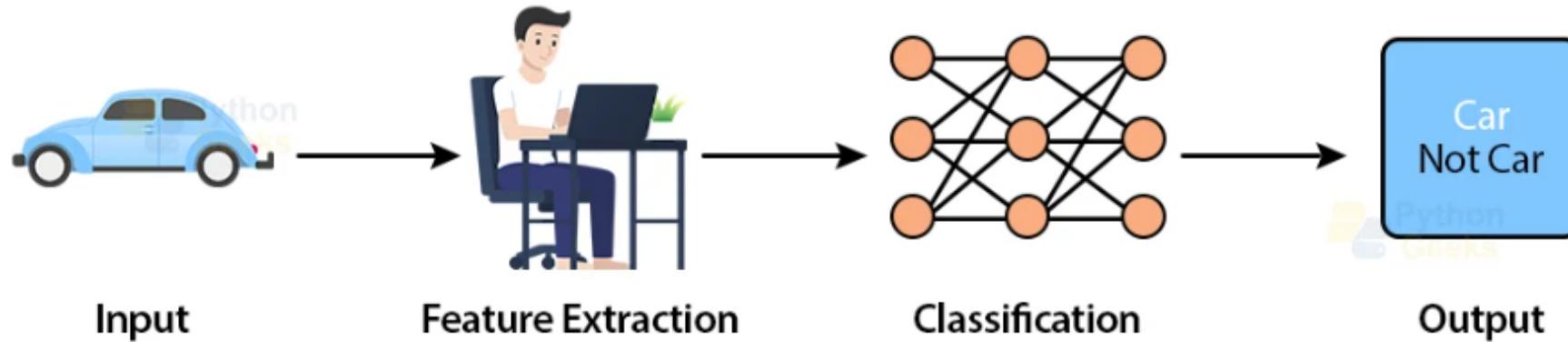
- L'agent doit apprendre par lui-même quelle est la meilleure stratégie, pour obtenir la meilleure récompense au fil du temps.

Exemples

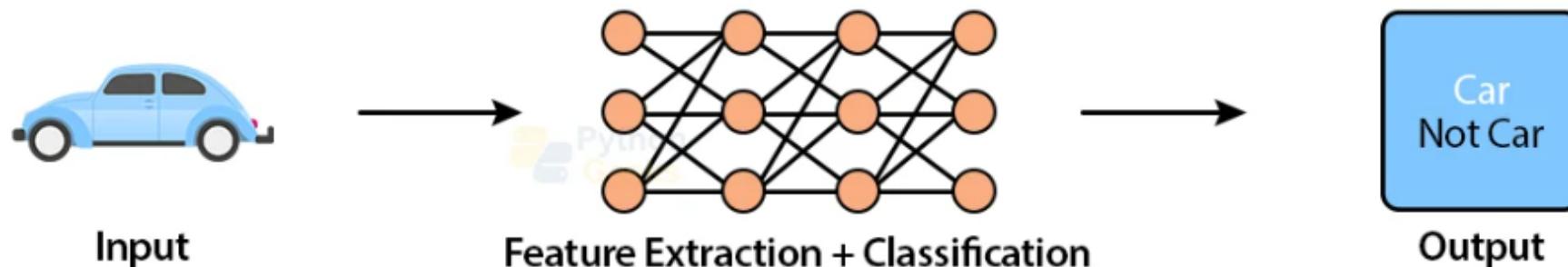
- Supposons que l'on dispose d'une collection d'articles de journaux. Nous souhaitons identifier des groupes d'articles portant sur un même sujet → **apprentissage non supervisé**
- Supposons que l'on dispose d'une base de données regroupant les caractéristiques de logements dans une ville : superficie, quartier, étage, prix, année de construction, nombre d'occupants, montant des frais de chauffage. Nous souhaitons prédire la facture de chauffage à partir des autres caractéristiques pour un logement qui n'appartiendrait pas à cette base → **apprentissage supervisé : régression**
- Supposons que l'on dispose des commentaires des clients d'une société sur FB et nous souhaitons analyser leurs sentiments pour orienter les services de la société → **apprentissage supervisé : classification**

Machine Learning vs Deep Learning (1/2)

Machine Learning



Deep Learning



Machine Learning vs Deep Learning (2/2)

Machine learning

- ✓ Permet aux machines de prendre leurs propres décisions, sur la base de données antérieures
- ✓ N'a besoin que d'une petite quantité de données d'entraînement
- ✓ Fonctionne bien sur les systèmes bas de gamme
- ✓ La plupart des fonctionnalités doivent être identifiées à l'avance et codées manuellement

Deep learning

- ✓ Permet aux machines de prendre des décisions à l'aide des réseaux de neurones artificiels
- ✓ A besoin d'une grande quantité de données de formation
- ✓ A besoin de systèmes haut de gamme pour fonctionner
- ✓ La machine apprend les caractéristiques à partir des données fournies



Concepts fondamentaux en ML : données (1/3)

Instances	Attributs			
	Nom	Diplôme	ancienneté	Titularisation
	Nesrine	Bac	6	oui
	Houda	ingénieur	2	oui
	Ahmed	licence	2	non
	Ahlem	Master	3	non
	Samir	License	4	non
	Hanin	License	5	oui
Hadi	ingénieur	6	oui	

Qualitatif (Categorical)

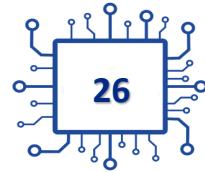
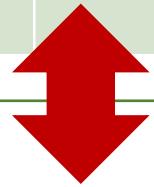
Quantitatif (Numeric)

Ordinal

Nominal

Continu

Discret



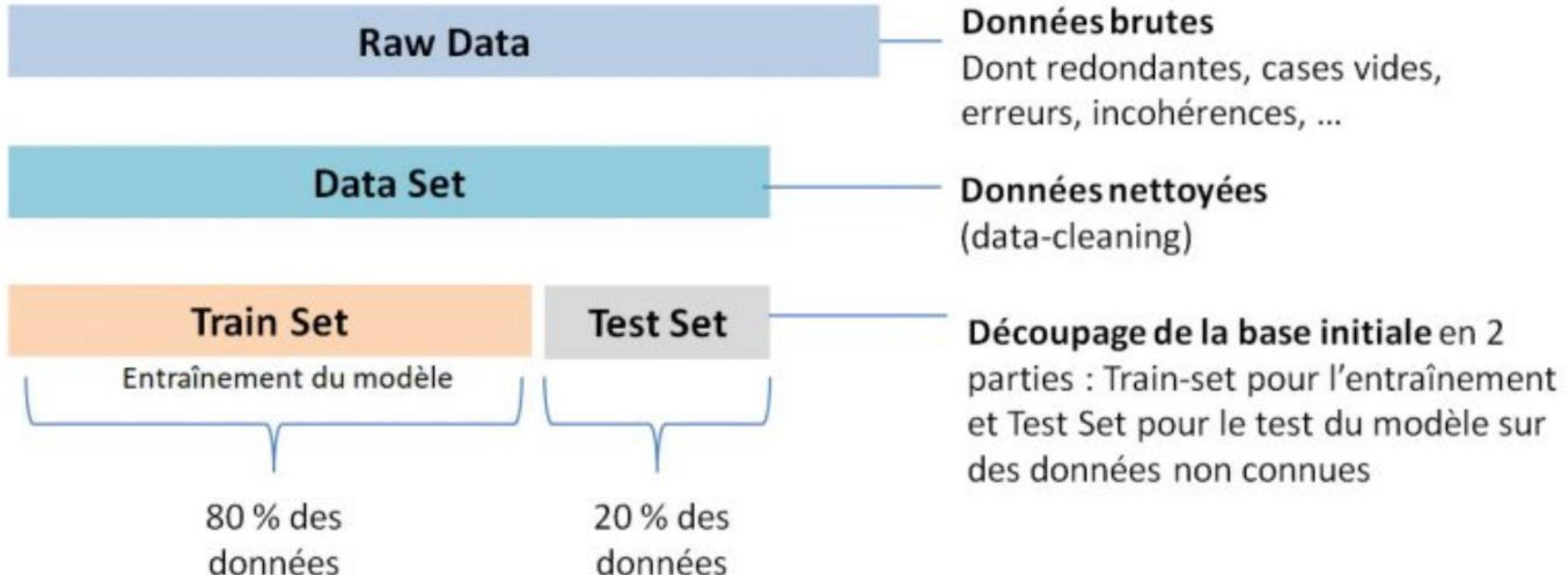
Concepts fondamentaux en ML : données (2/3)

#	Id	Name	Birthday	Gender	IsTeacher	#Students	Country	City
1	111	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	Spain	Madrid
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

Annotations pointing to specific errors:

- Invalid duplicate item: Points to the Id column for row 6.
- Incorrect format: Points to the Birthday column for row 6.
- Attribute dependency: Points to the Gender column for row 9.
- Missing value: Points to the City column for row 2.
- Invalid value: Points to the Gender column for row 5.
- Value that should be in another column: Points to the City column for row 7.
- Misspelling: Points to the City column for row 10.

Concepts fondamentaux en ML : données (3/3)



Travailler avec des données réelles

- Popular open datasets :
 - UC Irvine Machine Learning Repository
 - Kaggle datasets
 - Amazon's AWS datasets

- portals (they list open datasets):
 - <http://dataportals.org/>
 - <http://opendatamonitor.eu/>
 - <http://quandl.com/>

Merci pour votre attention

AI



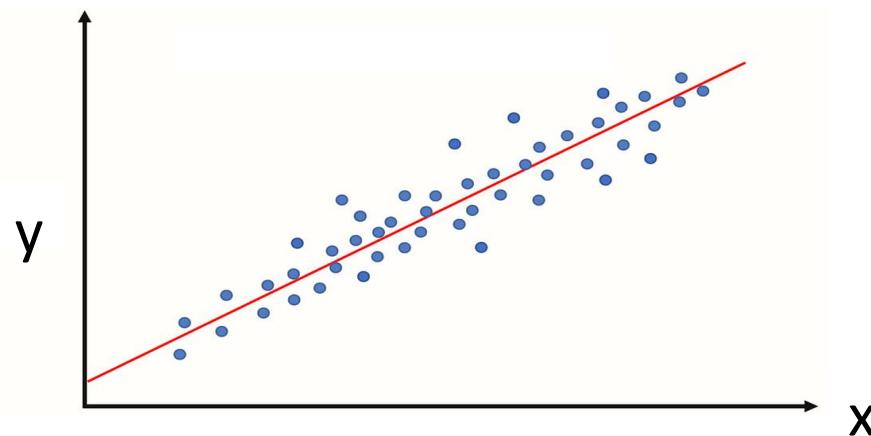
Des questions ?



Chapitre 2:

Apprentissage supervisé

Régression linéaire



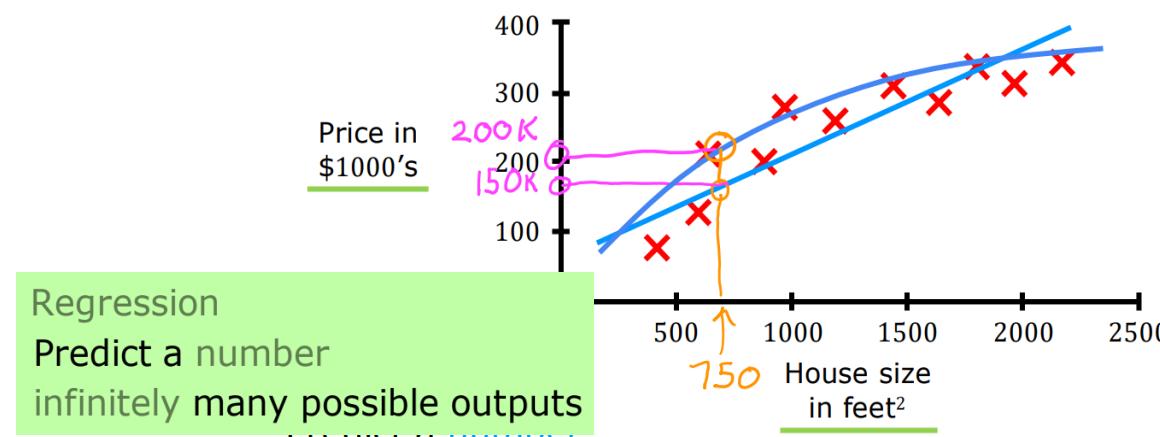
Plan

- Motivation
- La régression
- La fonction de prédiction
- La fonction coût de l'erreur
- Visualisation de la fonction coût
- L'algorithme gradient descent
- Influence du taux d'apprentissage sur l'algorithme
gradient descent
- La régression polynomiale

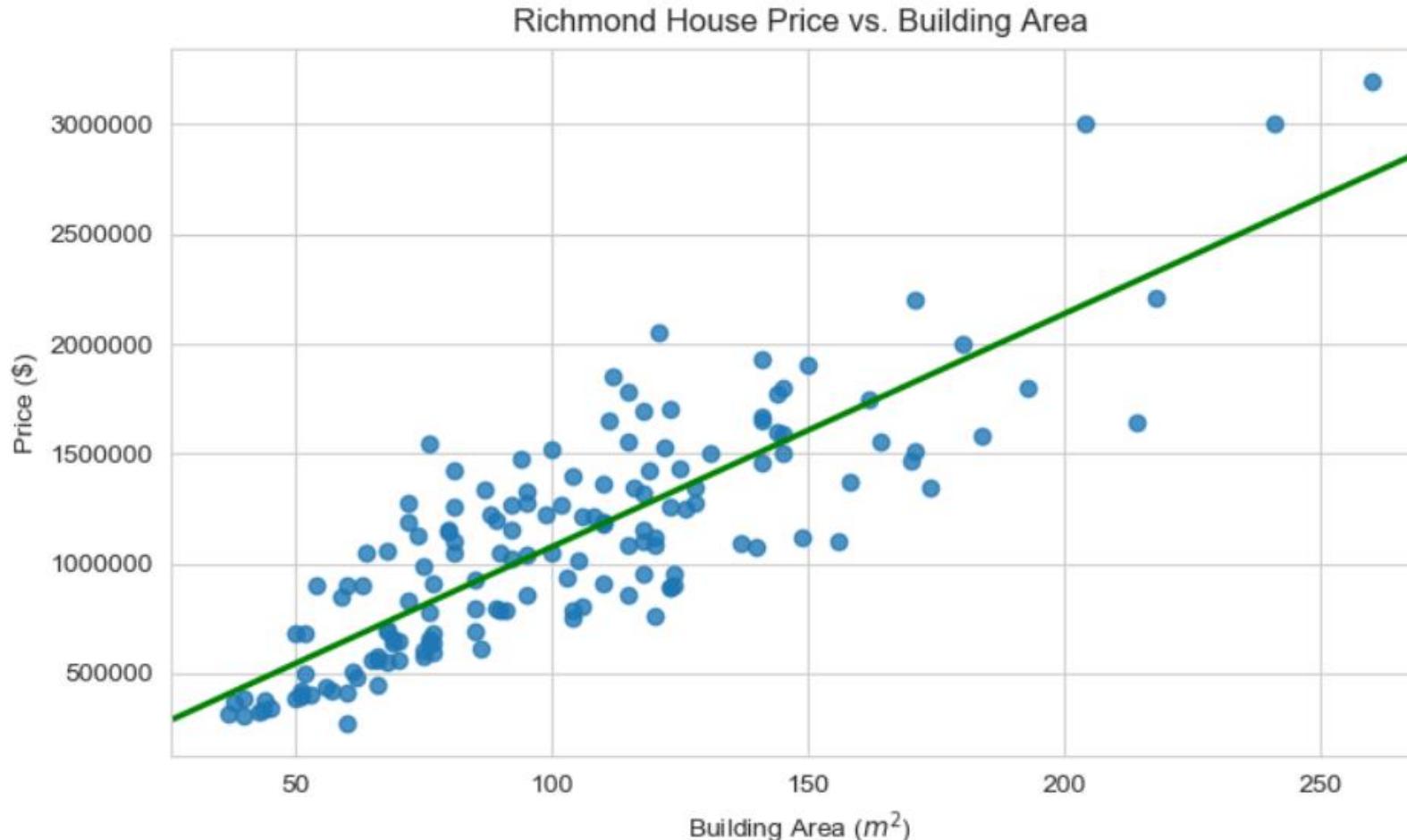


La régression

- En apprentissage supervisé, lorsque les étiquettes y prennent des valeurs scalaires infinies, on parle de problème de régression.
- Exemple : prédiction du prix d'une maison (output), connaissant un certain nombre d'attributs (superficie, chambres, etc.)
- Pour faire apprendre le système, nous donnons plusieurs exemples de maisons, avec leurs attributs (superficie) et leurs labels (prix).
- Après avoir établi le modèle (courbe bleu), nous pouvons prédire le prix de n'importe quelle maison

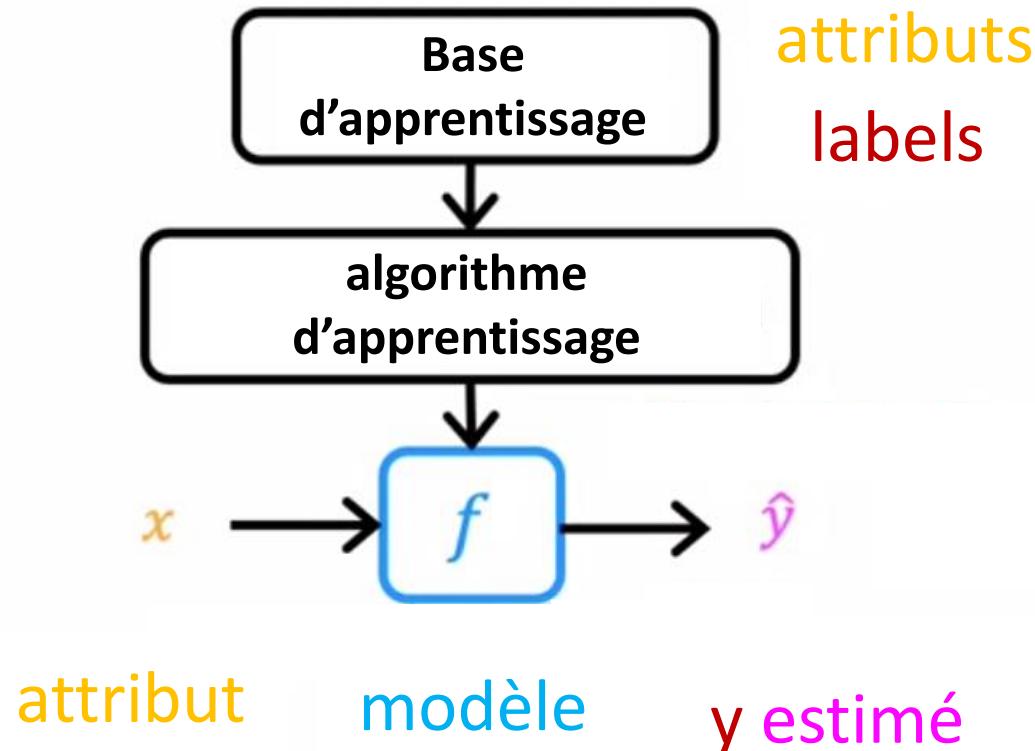


Régression linéaire



- Régression linéaire: une méthode de prédiction pour laquelle la cible (sortie) a une relation linéaire avec la variable d'entrée.

Terminologies pour la régression linéaire

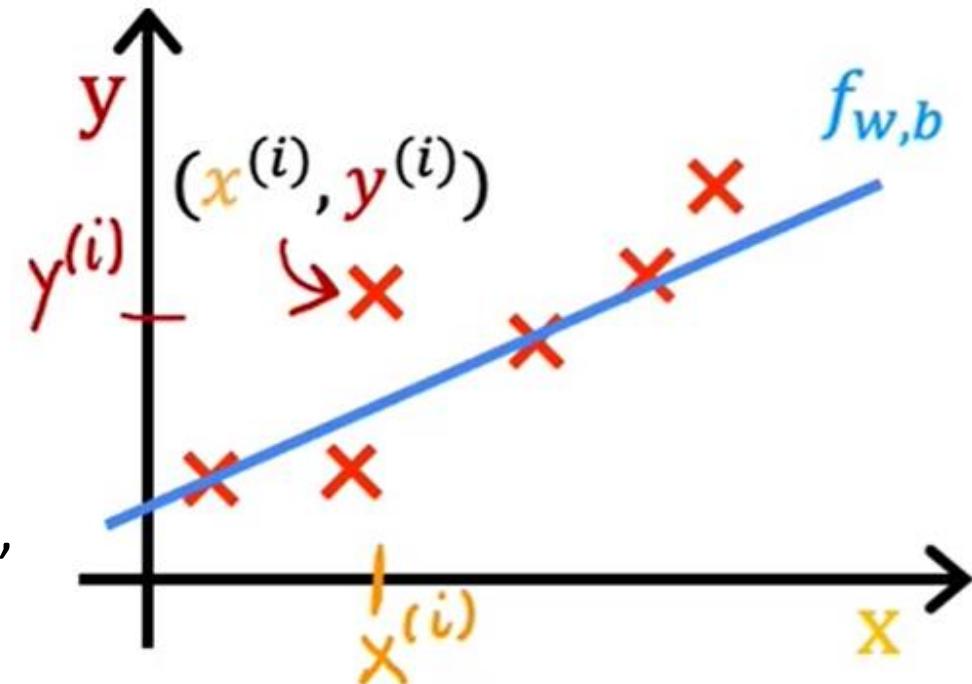


- x : variable d'entrée → **attributs**
- y : variable de sortie → **label**
- $(x^{(i)}, y^{(i)})$: i ème échantillon d'apprentissage
- \hat{y} : valeur estimée de y
- m : nombre d'échantillons d'apprentissage

Comment trouver la fonction de prédiction f ?

- La fonction f du modèle de la régression linéaire à une variable est une droite qui s'approchera le plus possible des données d'apprentissage
- La fonction de prédiction étant une droite, elle s'écrira mathématiquement sous la forme :

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$



- Les coefficients de cette droite sont :
 - w : désigne le paramètre poids (weight),
 - b : désigne le paramètre bias.

Estimation des coefficients de la droite w et b

- Les paramètres b et w sont obtenus par la méthode des moindres carrés, c'est-à-dire en minimisant la somme des carrés des résidus du modèle sur la base d'apprentissage : $\sum_{n=1}^N |y_n - b - w \cdot x|^2$

- Notons W le vecteur colonne des paramètres, Y le vecteur colonne des étiquettes, et X la matrice des observations :

$$W = \begin{pmatrix} b \\ w \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nd} \end{pmatrix}$$

- Avec ces notations, on obtient l'estimation de W au sens des moindres carrés par les équations dites **normales** (forme matricielle) :

$$W = (X^T X)^{-1} X^T Y$$

à la condition que $X^T X$ soit une matrice inversible

Exercice corrigé (1/3)

- Les données de ventes sur 4 semaines (en milliers) sont indiquées dans le tableau ci-dessous. Appliquer la régression linéaire sous forme matricielle pour prédire les ventes de la 5ème semaine

x_i (semaine)	y_i : vente (en milliers)
1	1
2	3
3	4
4	8

Exercice corrigé (2/3)

■ linear regression in matrix form = normal equation

- Here, the independent variable X is given as:

- $X^T = [1 \ 2 \ 3 \ 4]$

- The dependent variable is given as follows:

- $Y^T = [1 \ 3 \ 4 \ 8]$

- The data can be given as follows:

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix}, \quad Y = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix}$$

x_i (Week)	y_j (Sales in Thousands)
1	1
2	3
3	4
4	8

Exercice corrigé (3/3)

- The computation order of this equation is shown step by step as:

1. Computation of $(X^T X) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$

• $y = -1.5 + 2.2 * x$

x = 5 weeks

• $y = -1.5 + 2.2 * 5 = 9.5$

2. Computation of matrix inverse of $(X^T X)^{-1} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$

3. Computation of $((X^T X)^{-1} X^T) = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix}$

4. Finally, $((X^T X)^{-1} X^T) Y = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix} = \begin{pmatrix} -1.5 \\ 2.2 \end{pmatrix} \begin{matrix} \text{(Intercept)} \\ \text{slope} \end{matrix}$

La fonction coût de l'erreur (1/2)

- On définit l'erreur unitaire entre une valeur observée $y^{(i)}$ et une valeur prédite $\hat{y}^{(i)}$, comme suit : $(\hat{y}^{(i)} - y^{(i)})^2$
- Trouver le meilleur couple (b, w) revient à minimiser le coût global des erreurs unitaires qui se définit comme suit :

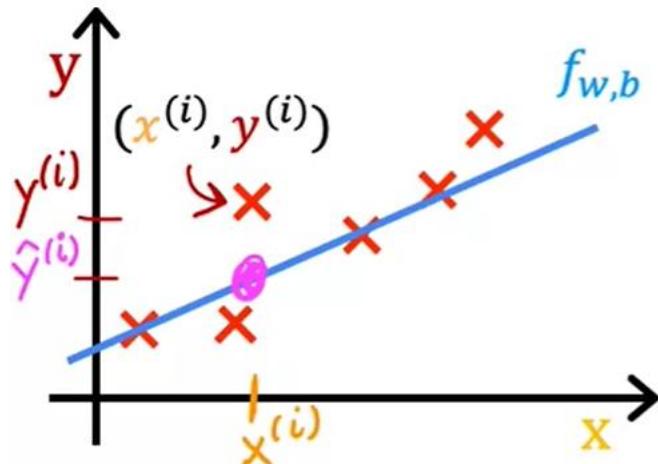
$$\sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

m : nombre d'échantillons
d'apprentissage

- La fonction de coût est définie comme suit :

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

La fonction coût de l'erreur (2/2)



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = w x^{(i)} + b$$

Comment retrouver w and b qui rendent $\hat{y}^{(i)}$ le plus proche possible de $y^{(i)}$ pour tout $(x^{(i)}, y^{(i)})$?



Cost function: Squared error cost function

$$\bar{J}(w, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

m = number of training examples

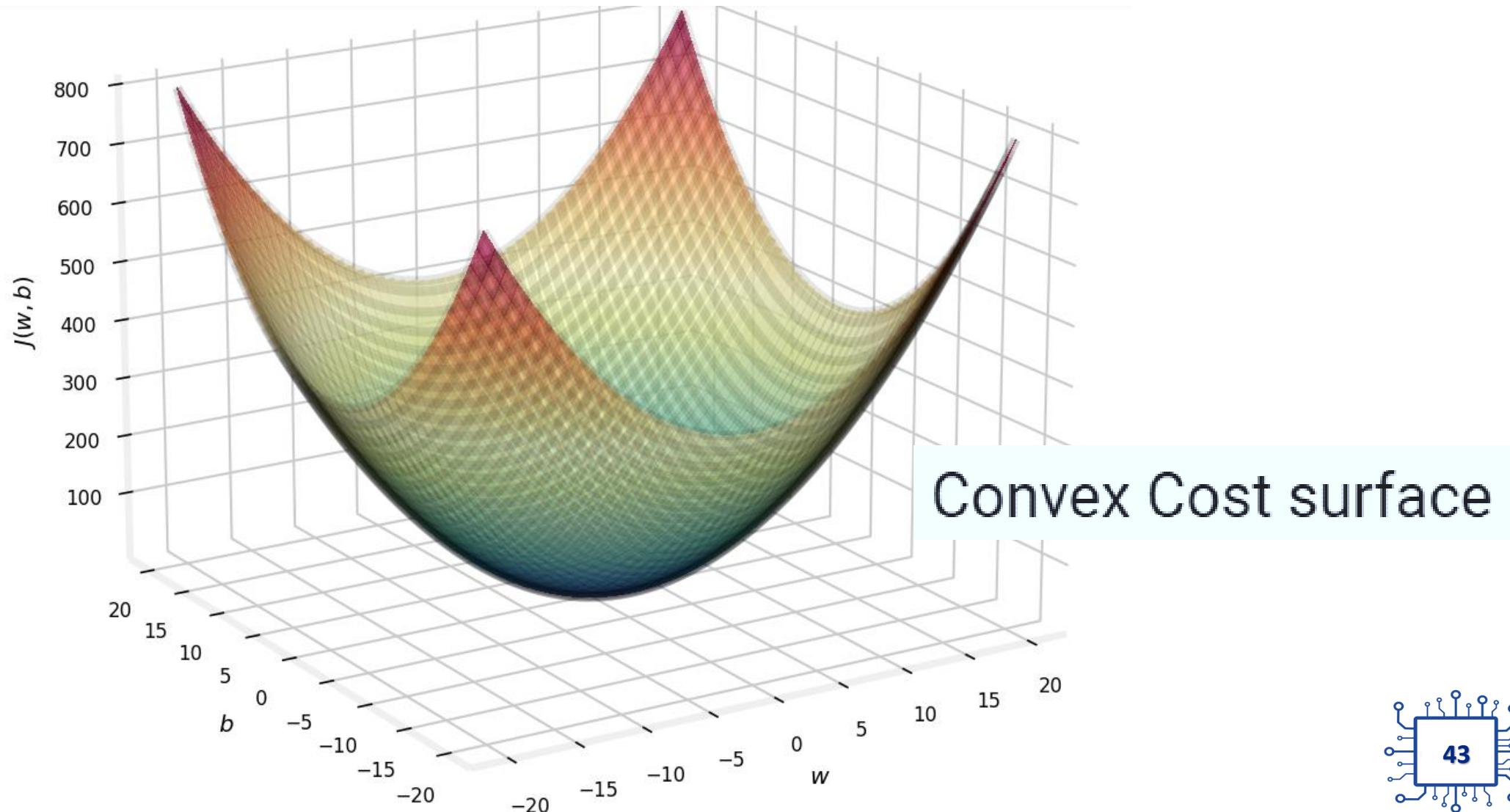
$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

But de la régression linéaire:

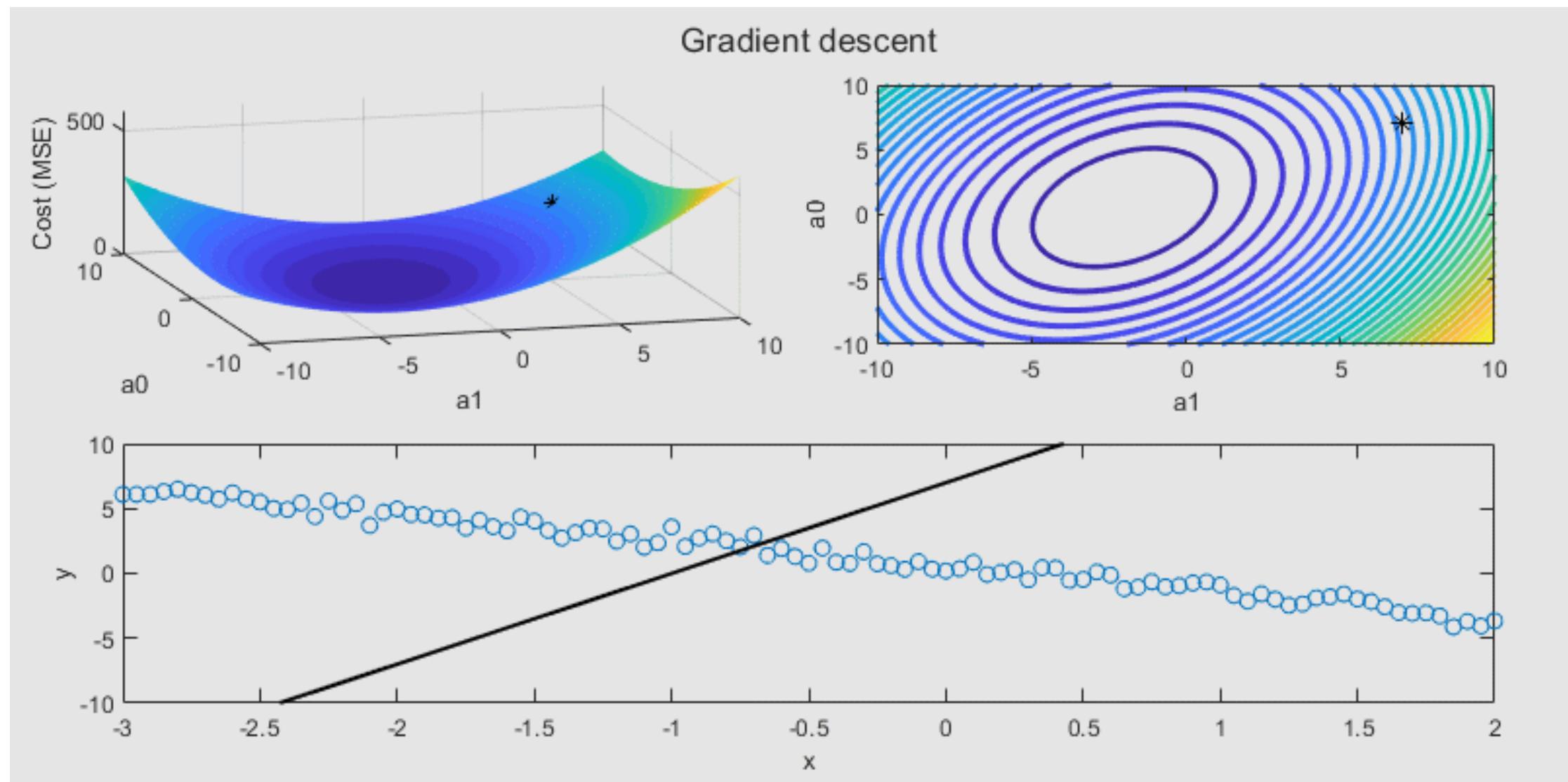
minimize $J(w, b)$



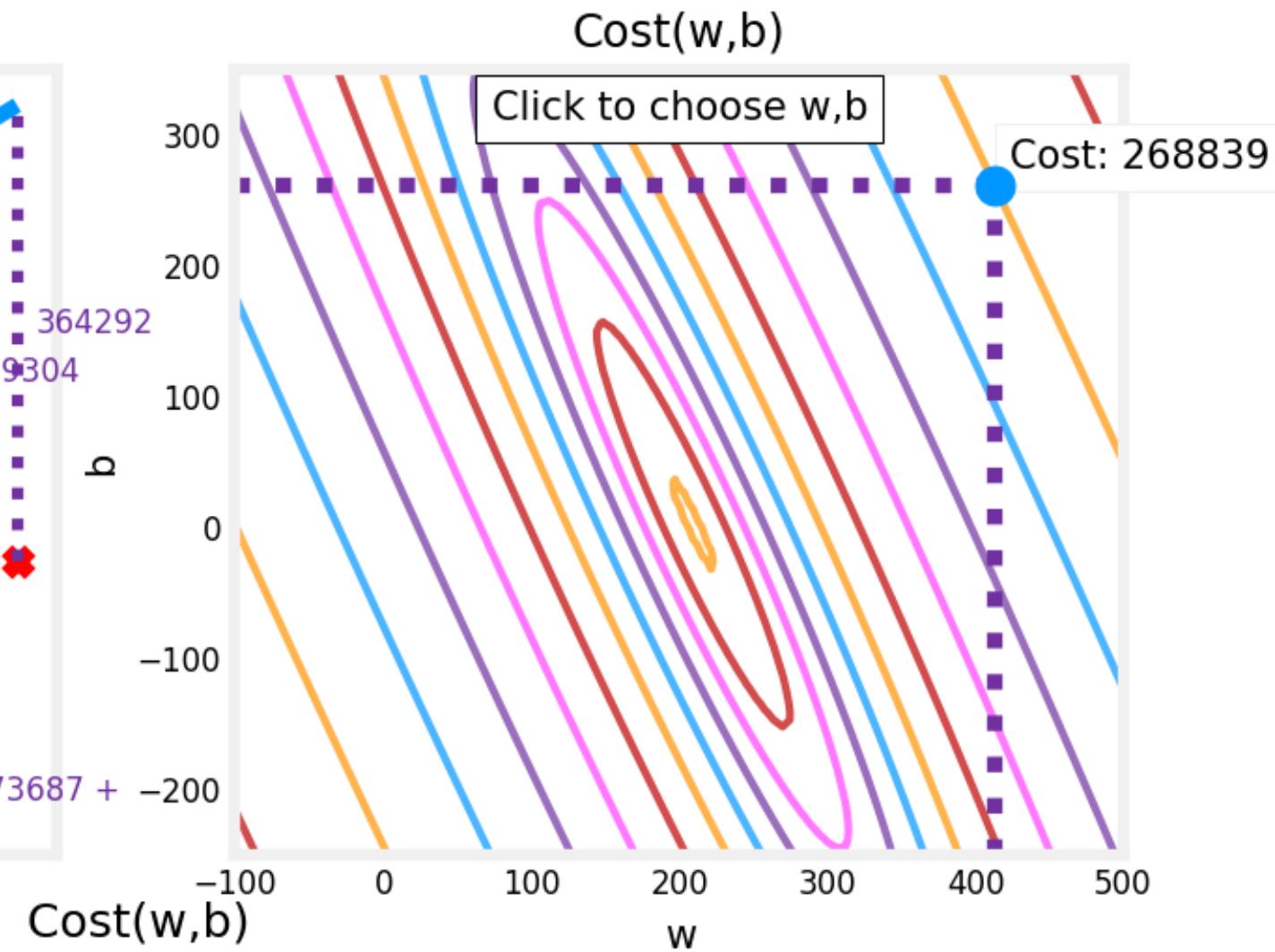
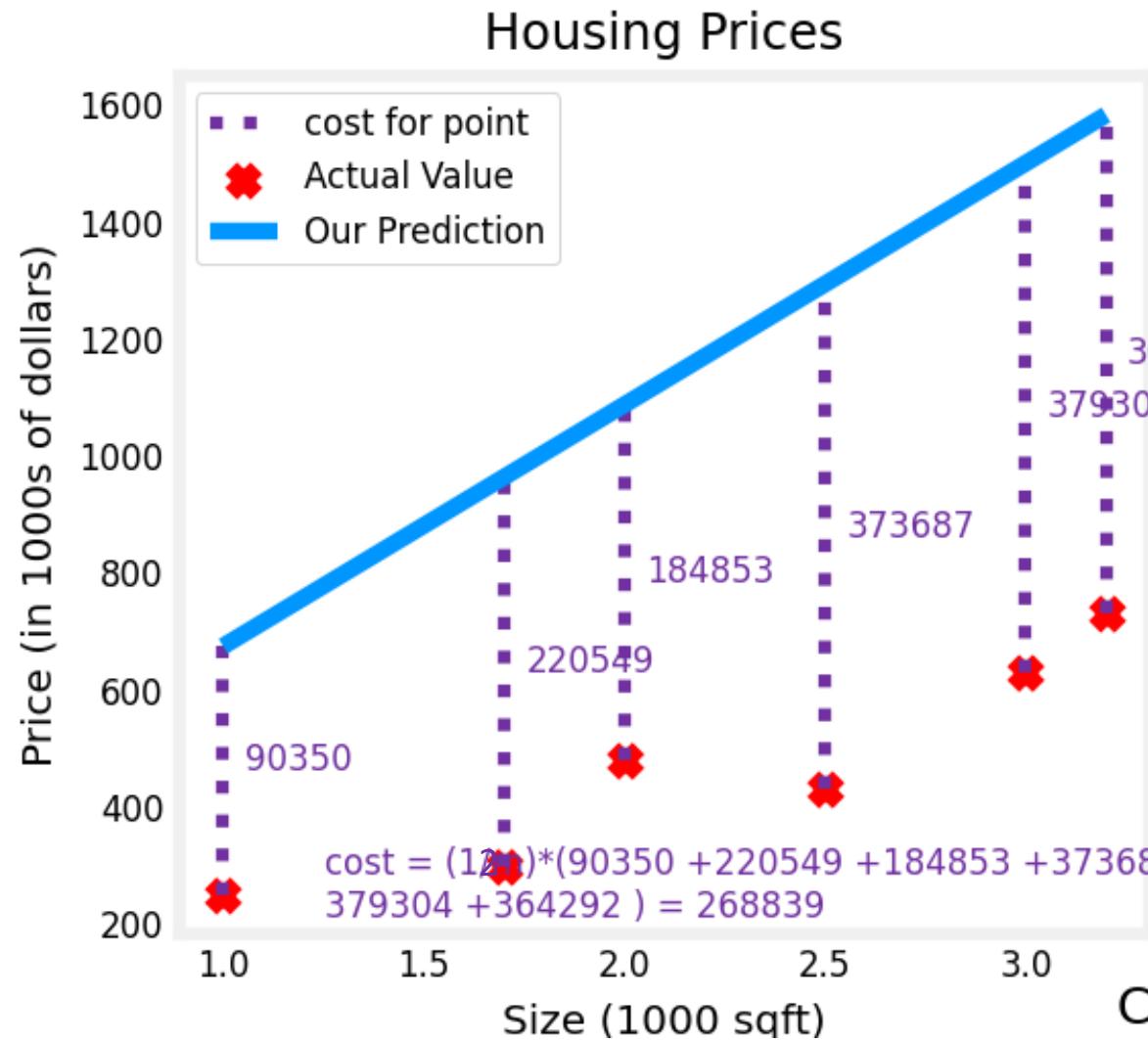
Visualiser la fonction coût (1/5)



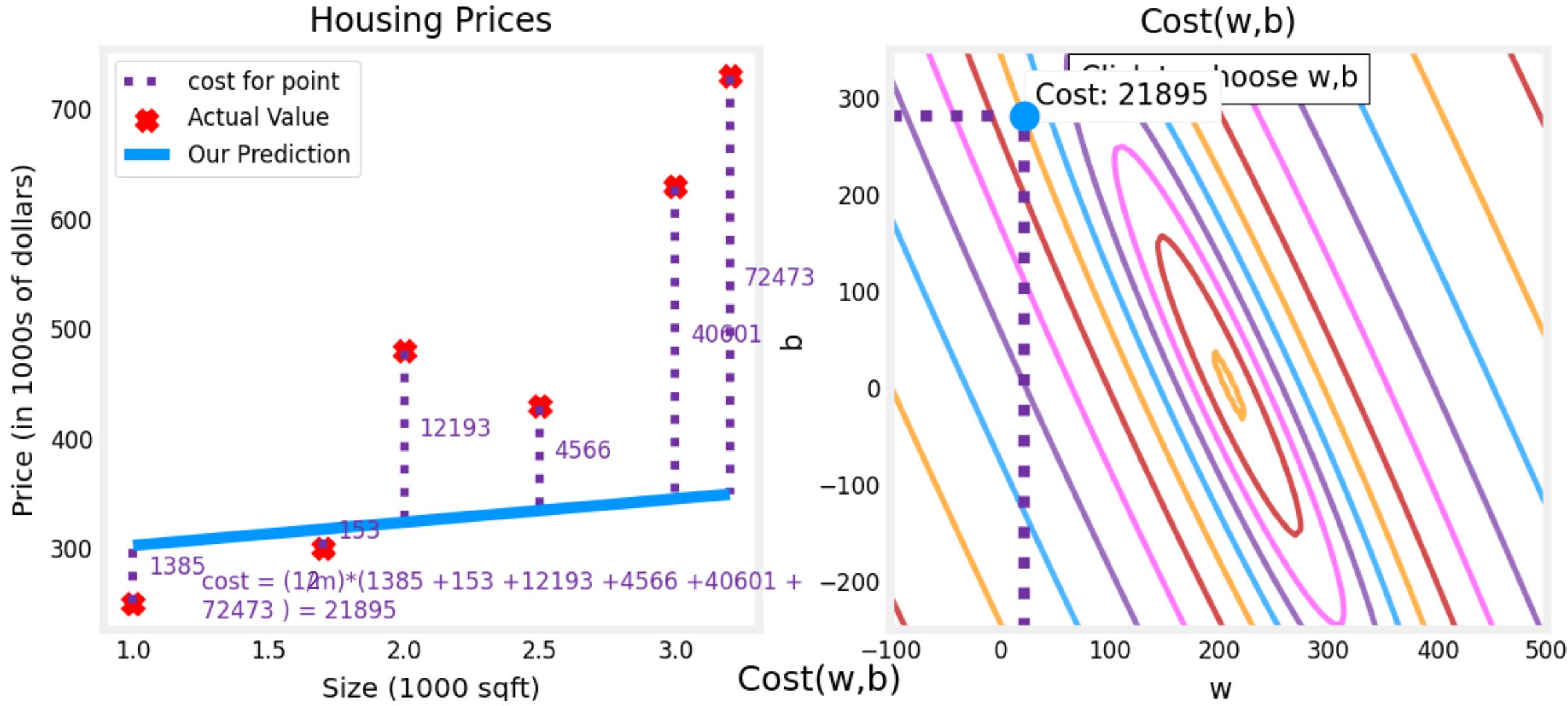
Visualiser la fonction coût (2/5)



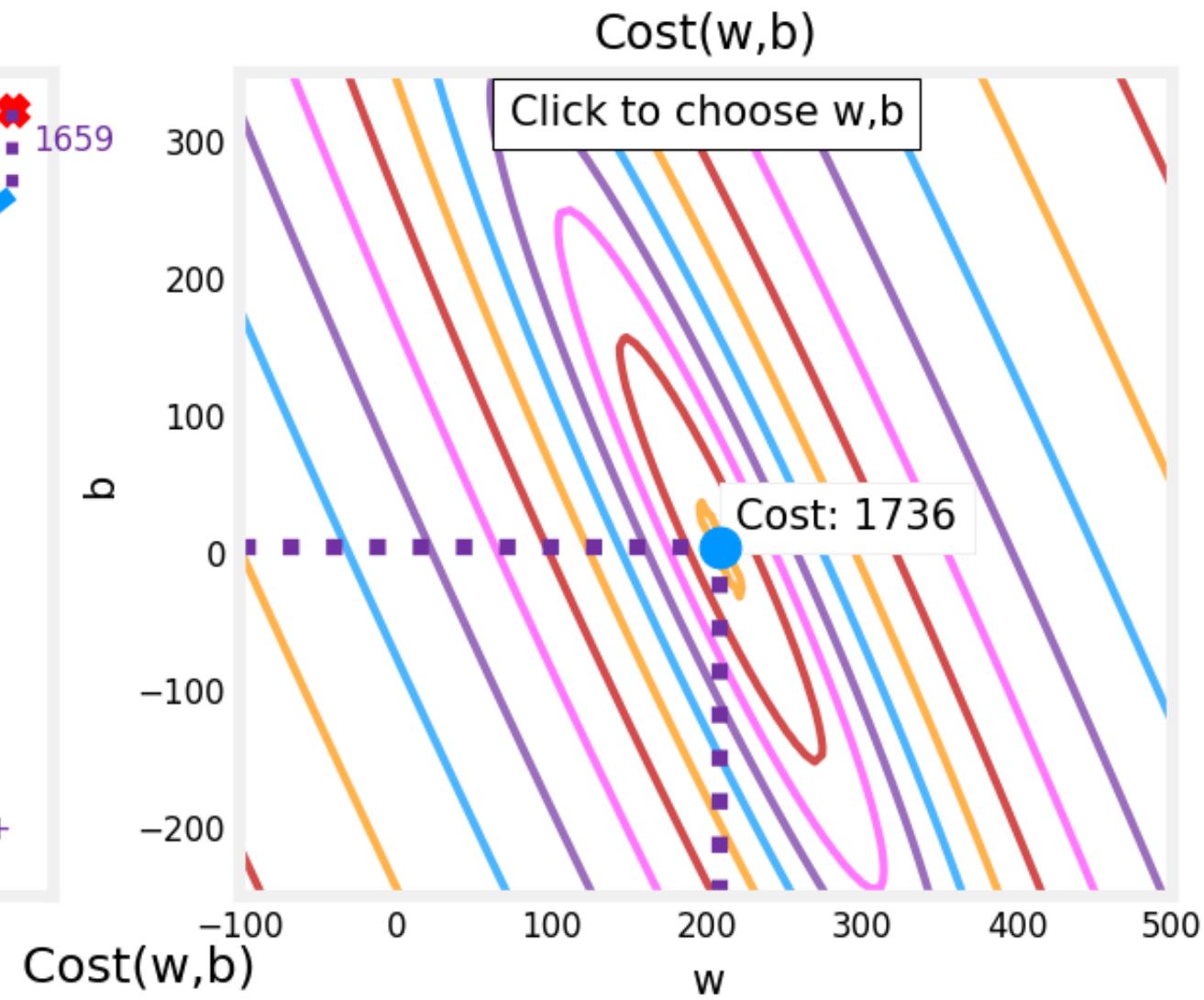
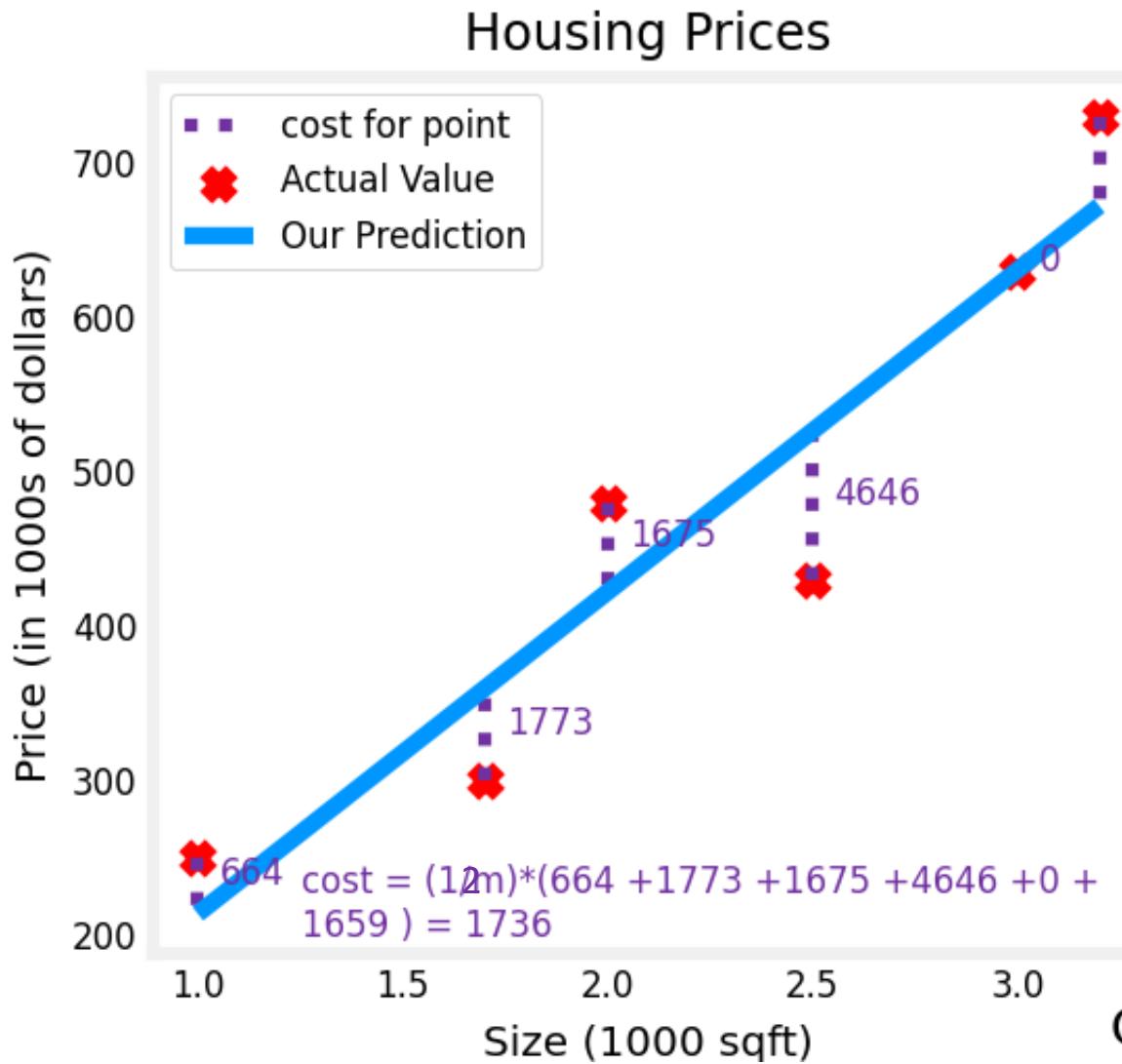
Visualiser la fonction coût (3/5)



Visualizing the cost function (4/5)



Visualizing the cost function (5/5)



Algorithme Gradient descent



- Gradient descent est un algorithme d'optimisation itératif

- vise à minimiser la fonction de coût $J(w, b)$ sur tous les échantillons d'apprentissage $(x^{(i)}, y^{(i)})$ définie par :

$$J(w, b) = \frac{1}{2m} \sum_{i=0}^{m-1} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

- Idée principale :

- une personne se demande si elle doit faire un tout petit pas dans une direction, et si elle veut arriver au fond de la vallée le plus rapidement possible : Quelle direction doit-elle choisir pour faire ce petit pas ?
 - Mathématiquement, c'est la direction de la descente la plus rapide
 - À partir de ce nouveau point, elle peut à nouveau regarder autour d'elle et décider quelle direction mènerait le plus rapidement à la descente.
 - Faire encore un pas, encore un pas, et ainsi de suite, jusqu'à se retrouver au fond de cette vallée : ce qui est un minimum local,
 - Ce qu'elle vient de faire : passer par les étapes de l'algorithme gradient descent.

Étapes de l'algorithme gradient descent

- Initialiser w et b aléatoirement,
- Choisir le learning rate
- Répéter jusqu'au convergence :

$$\begin{cases} w = w - \alpha \frac{\partial J(w, b)}{\partial w} \\ b = b - \alpha \frac{\partial J(w, b)}{\partial b} \end{cases}$$

Learning rate

où les paramètres w, b doivent être mis à jour simultanément.

- Le gradient est défini par :

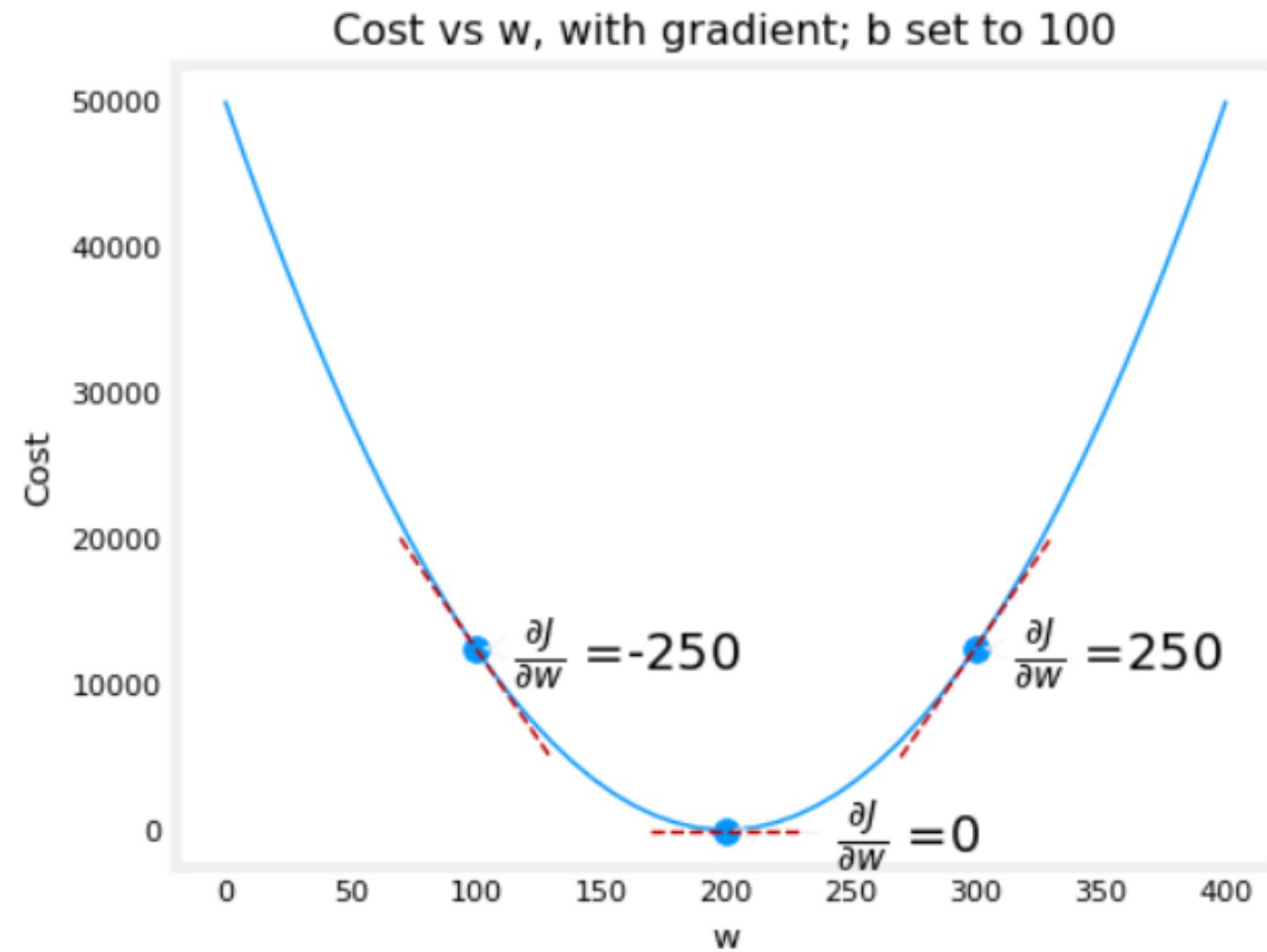
$$\frac{\partial J(w, b)}{\partial w} = \frac{1}{m} \sum_{i=0}^{m-1} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$\frac{\partial J(w, b)}{\partial b} = \frac{1}{m} \sum_{i=0}^{m-1} (f_{w,b}(x^{(i)}) - y^{(i)})$$

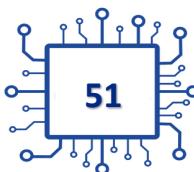
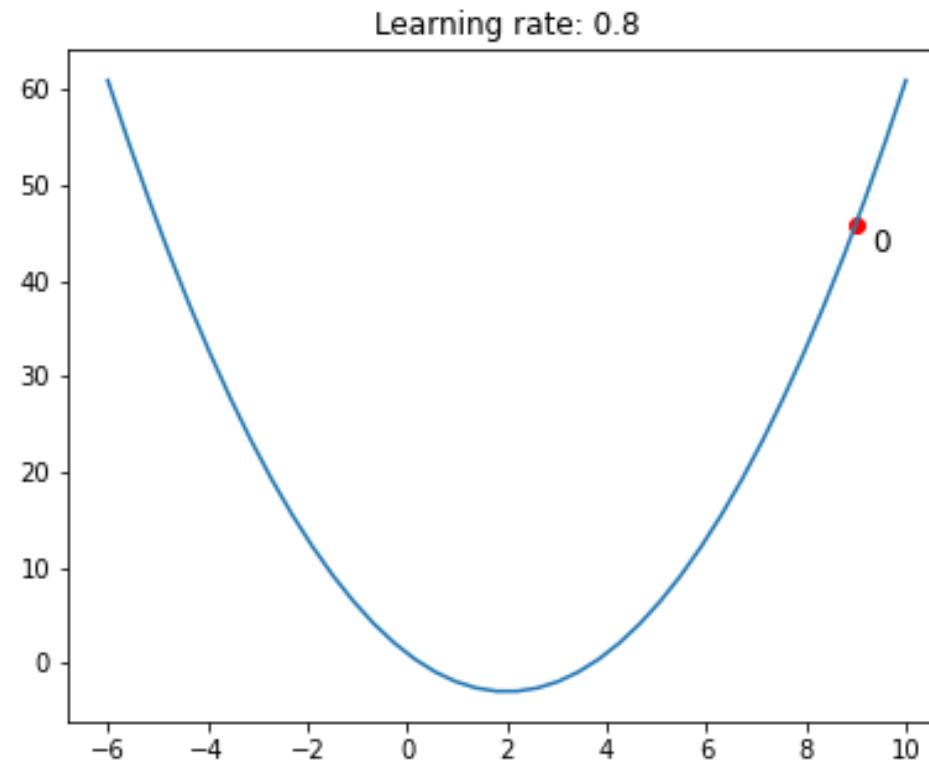
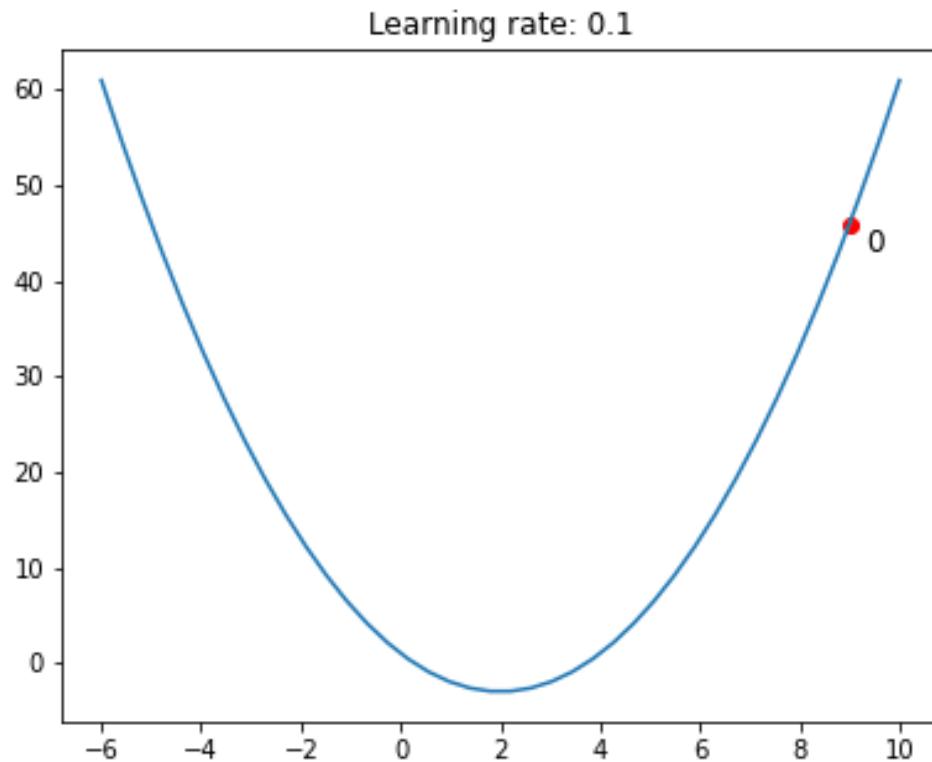
Intuition du Gradient descent

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

- La courbe montre $\partial J(w, b) / \partial w$ ou la pente de la courbe coût en 3 points :
- À droite du tracé, la dérivée est **positive**, tandis qu'à gauche elle est **négative**..
- En raison de la «forme du bol», les dérivés mèneront toujours une descente en pente vers le bas où le gradient est nul.



Comment le taux d'apprentissage (learning rate) affecte le comportement du gradient descent



Comment le taux d'apprentissage (learning rate) affecte le comportement du gradient descent

- Si α est très petite:

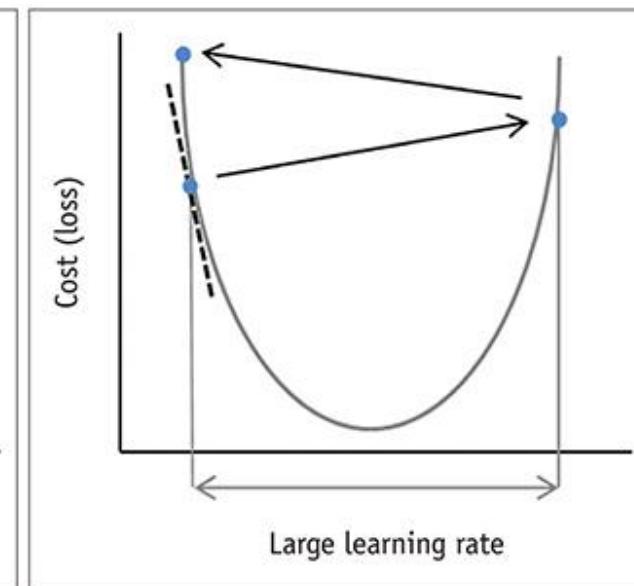
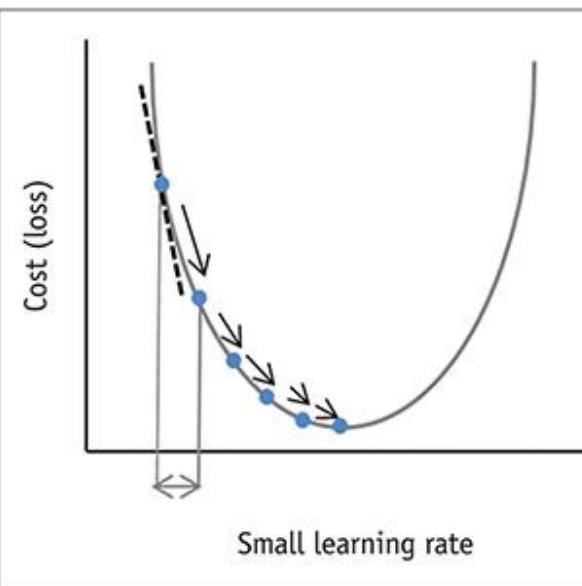
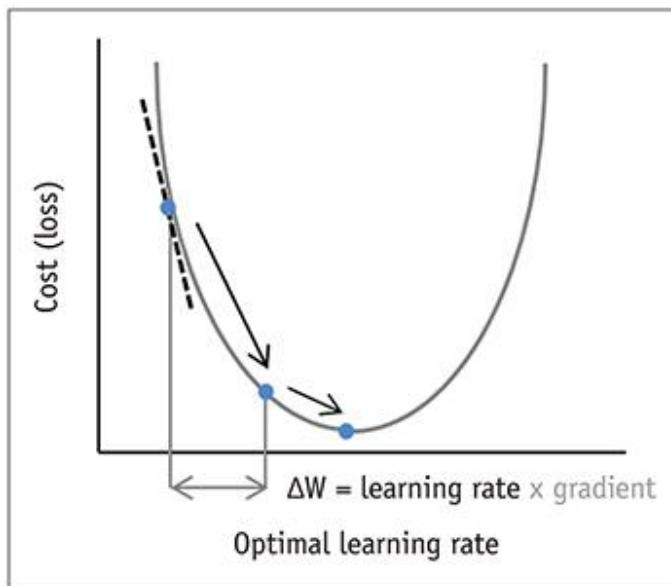
- Gradient descent may be slow : slow convergence

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

Learning rate

- Si α est très grande :

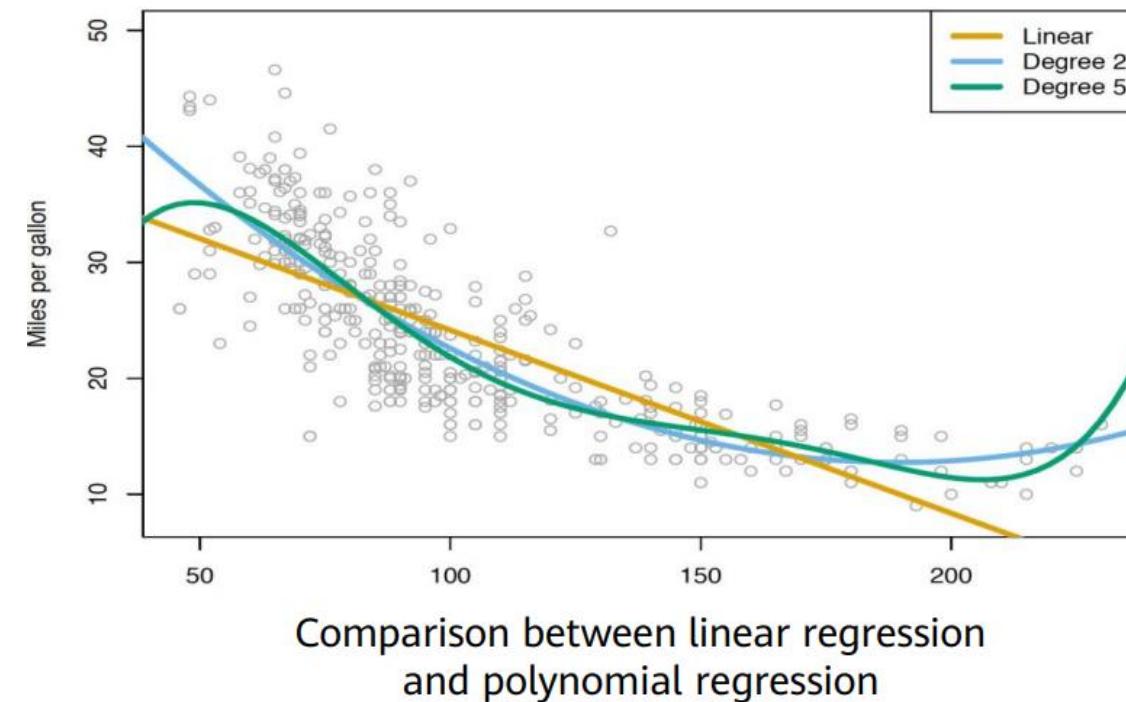
- gradient descent fait des dépassemens, n'atteint jamais le minimum
 - gradient descent échoue à converger: diverge



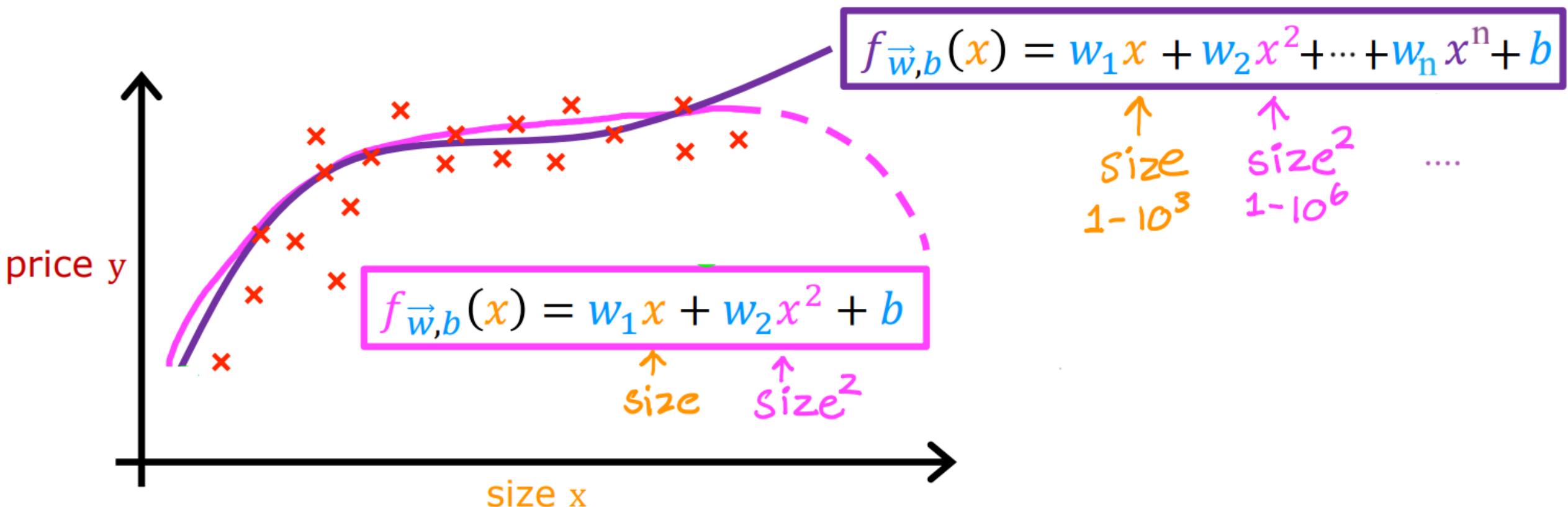
La régression polynomiale (1/3)

- La régression Polynomiale est une extension de la régression linéaire.
- En général, les données sont complexes : difficile de trouver une droite linéaire qui approche au mieux les variables → **la solution** est d'utiliser régression polynomiale.
- La nième puissance est le degré du polynôme .

La régression polynomiale appartient à la régression linéaire car la relation entre ses paramètres de poids w est toujours linéaire tandis que sa non-linéarité se reflète dans la dimension des attributs

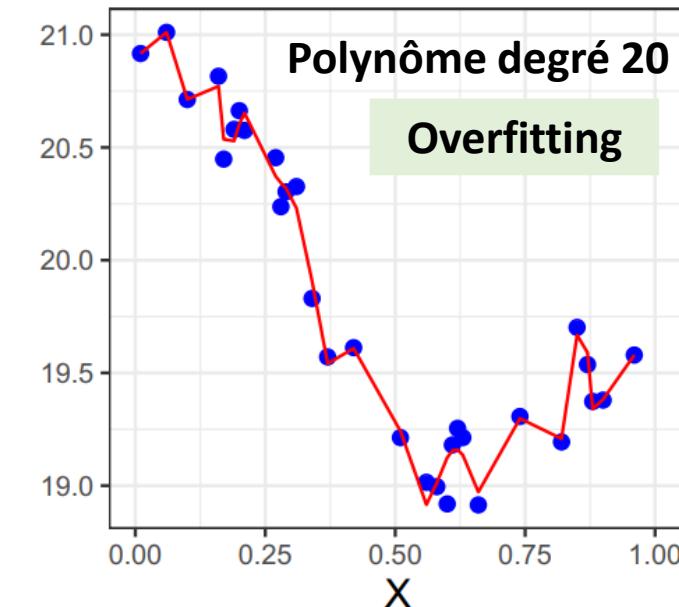
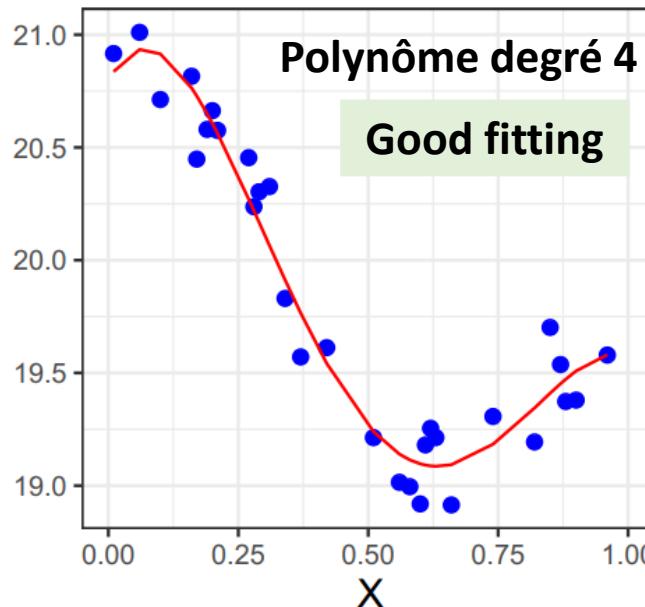
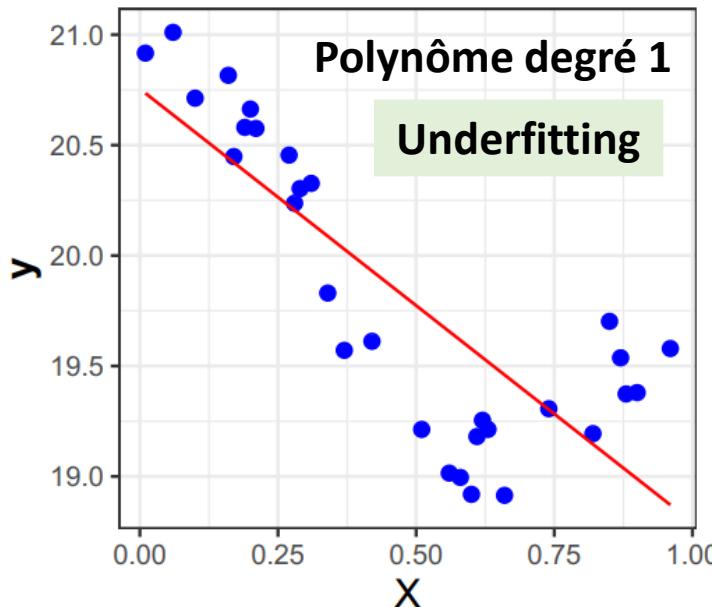


La régression polynomiale (2/3)



La régression polynomiale (3/3): quel degré n choisir ?

- Généralisation : indique comment le modèle réagit devant des données non vues :
 - **Overfitting** : le modèle apprend bien des données d'apprentissage mais ne parvient pas à généraliser sur les données de test
 - **Underfitting** : modèle ne peut ni apprendre parfaitement des données d'entraînement ni généraliser sur de nouvelles données de test
- Exemple : polynômes de degré 1, 4 et 20 avec données d'apprentissage et de test



Exercice 1

Modèle (a) : Biais élevé, variance élevé → Sous-apprentissage du modèle

Modèle (b) : Biais faible, variance faible → modèle bien généralisé

Modèle (c) : Biais faible et variance importante → sur-apprentissage du modèle

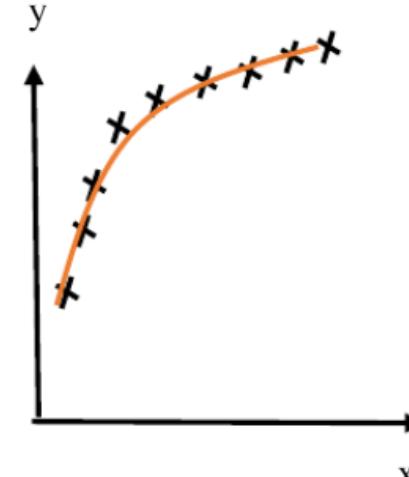
- Soient les graphes suivants qui illustrent le résultat de la prédiction y par le modèle de régression en fonction des échantillons d'apprentissage x .



(a)



(b)



(c)

Discuter les résultats obtenus pour les modèles

Exercice 2

- Soit le tableau suivant qui résume les résultats d'application de trois modèles de classification sur un jeu de données :

	Modèle 01	Modèle 02	Modèle 03
Erreur d'apprentissage	1%	25%	0.5%
Erreur de test	20%	26%	1%

Discuter les résultats obtenus pour les 3 modèles

Modèle 01 : Biais faible et variance importante → Sur-apprentissage du modèle

Modèle 02 : Biais important et variance élevé → sous-apprentissage du modèle

Modèle 03 : Biais et variance faible → Modèle bien généralisé (performant)

Merci pour votre attention

AI



Des questions ?



Chapitre 3:

Apprentissage supervisé

La classification



Plan

- Motivation
- La classification
- Évaluation des performances d'un classifieurs
- Algorithme KNN de classification
- Algorithme arbre de décision
- Algorithme neuronal

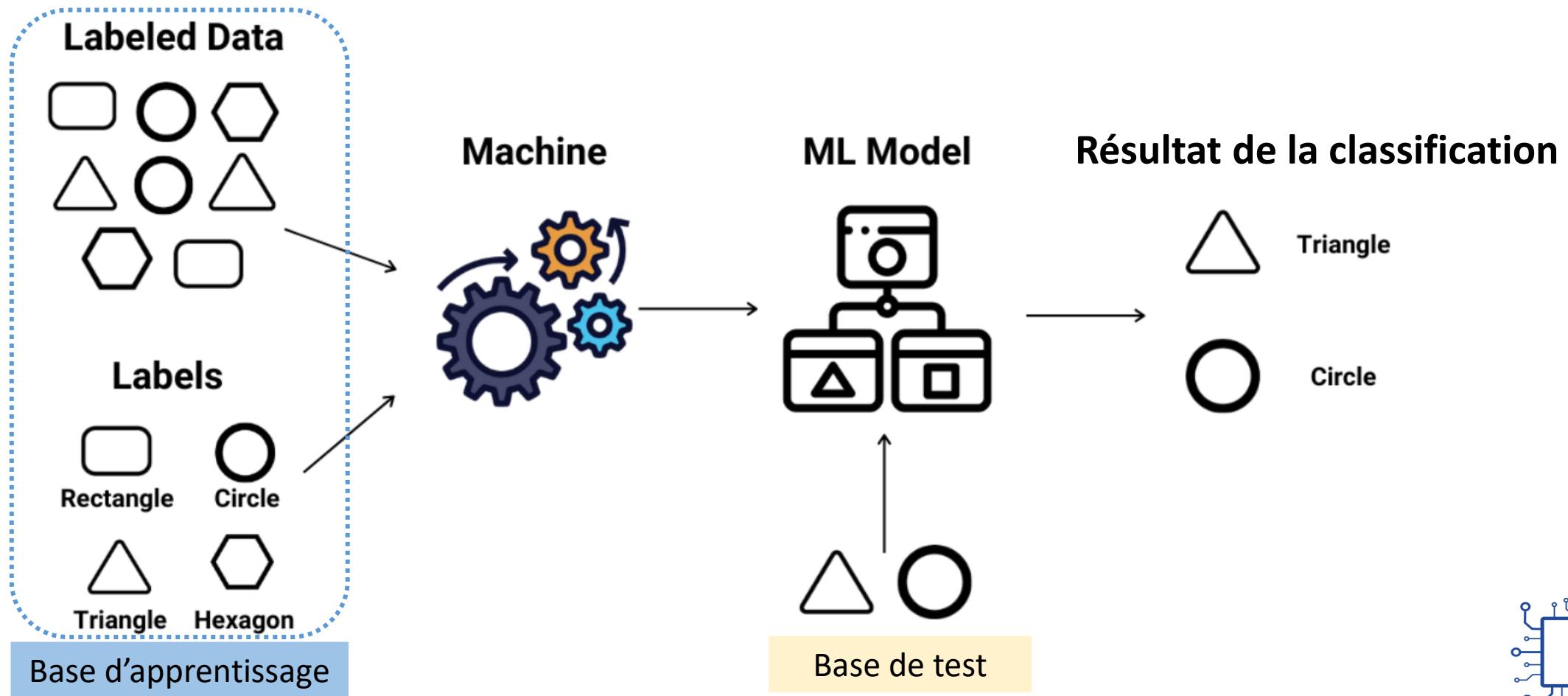


Objectifs

- Connaitre les critères d'évaluation d'un modèle d'apprentissage supervisé
- Comprendre la performance en généralisation d'un modèle d'apprentissage supervisé.
- Maitriser les algorithmes de classification supervisée :
 - KNN
 - Arbre de décision
 - Perceptron

La classification

La **classification supervisée** est la catégorisation d'individus (ou objets). Elle consiste à attribuer une classe ou catégorie à chaque individu à classer, en se fondant sur leurs attributs (en général statistiques)



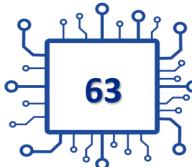
Évaluation des performances (1/3)

- La matrice de confusion est un outil de mesure de la performance des modèles de classification à 2 classes ou plus.
- Dans le cas binaire (à deux classes) la matrice de confusion est un tableau à 4 valeurs représentant les différentes combinaisons de valeurs réelles et valeurs prédites :

Estimated amount		yes	no
Actual amount	yes	TP	FN
no	FP	TN	

- TP (true positive) = nombre de cas positifs correctement classés par le classifieur,
- TN (true negative) = nombre de cas négatifs correctement classés par le classifieur,
- FP (false positive) = nombre de cas négatifs incorrectement classés par le classifieur,
- FN (false negative) = nombre de cas positifs incorrectement classés par le classifieur.

La matrice de confusion indique le nombre de cas qui appartiennent réellement à la classe i mais qui ont été attribués à la classe j

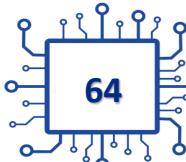


Évaluation des performances (2/3)

Measurement	Ratio
Accuracy and recognition rate proportion de bonnes prédictions par rapport à toutes les prédictions	$\frac{TP + TN}{P + N}$
Error rate and misclassification rate proportion de mauvaises prédictions par rapport à toutes les prédictions	$\frac{FP + FN}{P + N}$
Sensitivity, true positive rate, and recall (proportion d'exemples positifs correctement identifiés)	$\frac{TP}{P}$
Specificity and true negative rate (proportion d'exemples négatifs correctement identifiés)	$\frac{TN}{N}$
Precision (proportion de prédictions correctes parmi les prédictions positives)	$\frac{TP}{TP + FP}$
F_1 , harmonic mean of the recall rate and precision la moyenne harmonique de la précision et du rappel	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
F_β , where β is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

Actual amount	Estimated amount		Total
	yes	no	
yes	TP	FN	P
no	FP	TN	N
Total	P'	N'	$P + N$

Confusion matrix



Exemple

- Considérons la matrice de confusion suivante, correspondante à un classifieur binaire :

	Predicted Yes	Predicted No
Actual Yes	95	5
Actual No	5	45

- Calculer :
 - Taux de reconnaissance ou Accuracy
 - Taux d'erreur ou Error rate
 - La sensibilité ou Recall
 - La spécificité
 - La précision
 - F1 score

Solution

	Predicted Yes	Predicted No
Actual Yes	95	5
Actual No	5	45

- Accuracy : overall : how often is the classifier correct ? = $(45+95)/150$
- Error rate : overall : how often is it wrong? = $(5+5)/150$
- Recall : When it is actually yes, how often does he predict yes? = $95/100$
- Specificity : When it is actually no, how often does he predict no? = $45/50$
- Precision : when it predicts yes, how often he is correct ? = $95/100$

Exercice 1

- Considérons le problème de classification à 2 classe suivant, qui consiste à prédire si la personne dans une photo est une femme ou un homme. Le tableau suivant donne les étiquettes prédites et actuelles pour une base de test à 10 personnes

1. Donner la matrice de confusion

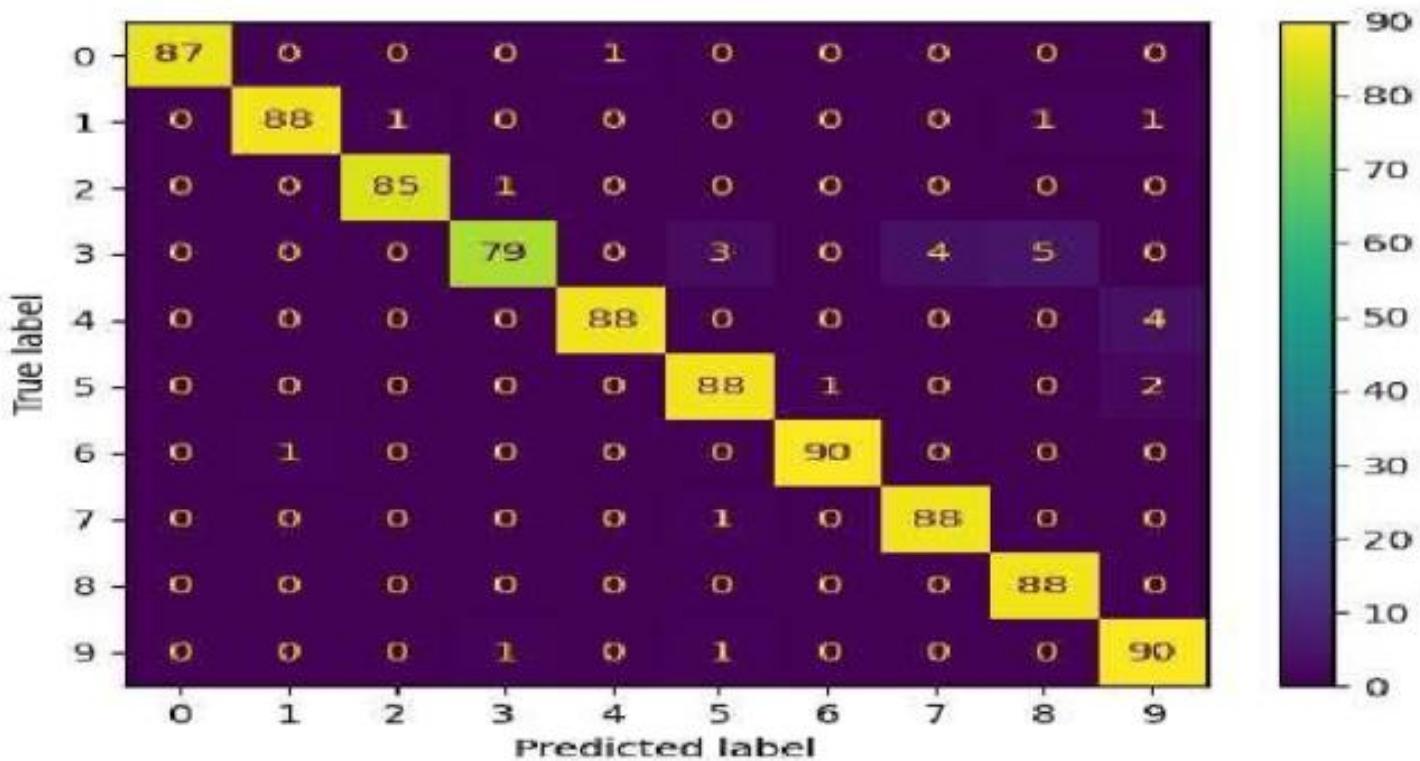
2. Calculer :

- Taux de reconnaissance ou Accuracy
- Taux d'erreur ou Error rate
- La sensibilité ou Recall
- La spécificité
- La précision
- F1 score

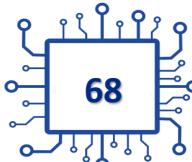
	Expected	Predicted
1	man	woman
2	man	man
3	woman	woman
4	man	man
5	woman	man
6	woman	woman
7	woman	woman
8	man	man
9	man	woman
10	woman	woman

Exercice 2

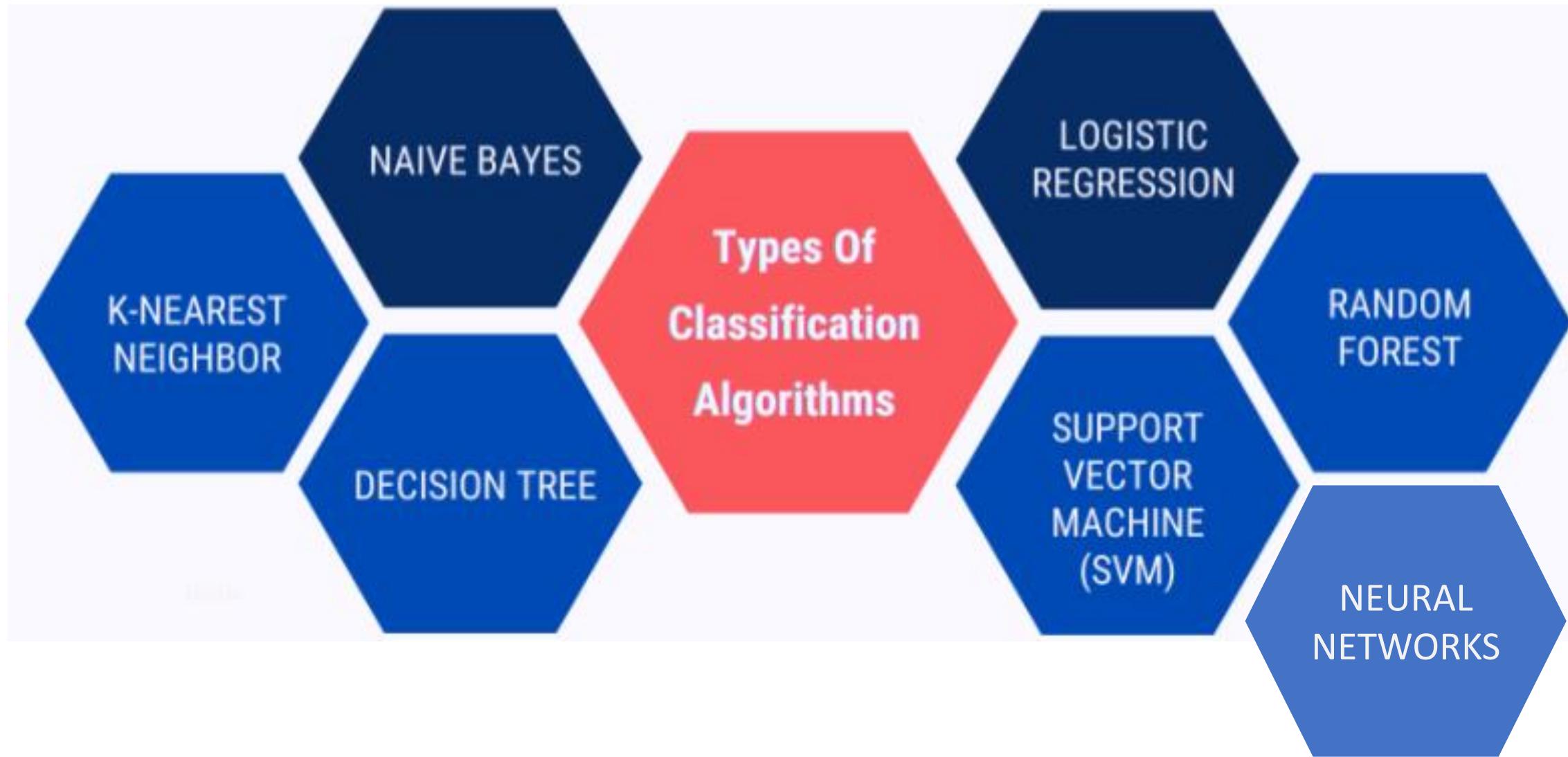
Soit la matrice de confusion suivante d'un modèle de classification donné :



1. Que représente cette matrice ?
2. Suggérer sur quelle application cette matrice a été calculée
3. Interpréter la matrice
4. Supposant que les données sont équilibrées, évaluer les résultats de classification du modèle.
5. Que peut-on dire de ce modèle ?



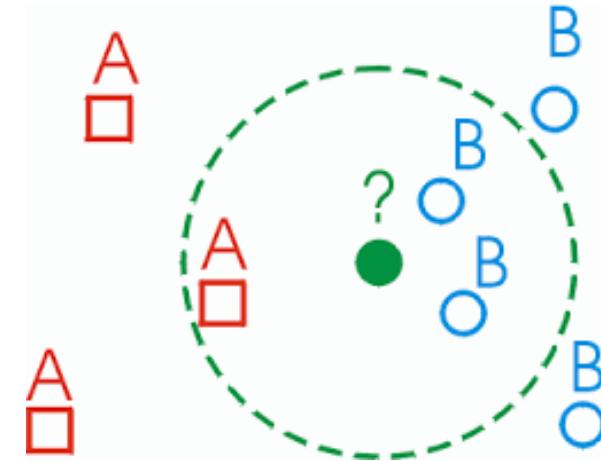
Exemples d'algorithmes de classification



Algorithme KNN de classification (1/5)

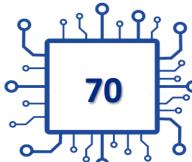
K plus proche voisins = K-Nearest Neighbors (KNN)

- “Dis-moi qui sont tes amis, et je te dirais qui tu es”
- Un algorithme de classification simple mais très puissant
- Classification basée sur une mesure de similarité
- Non paramétrique (sauf K)



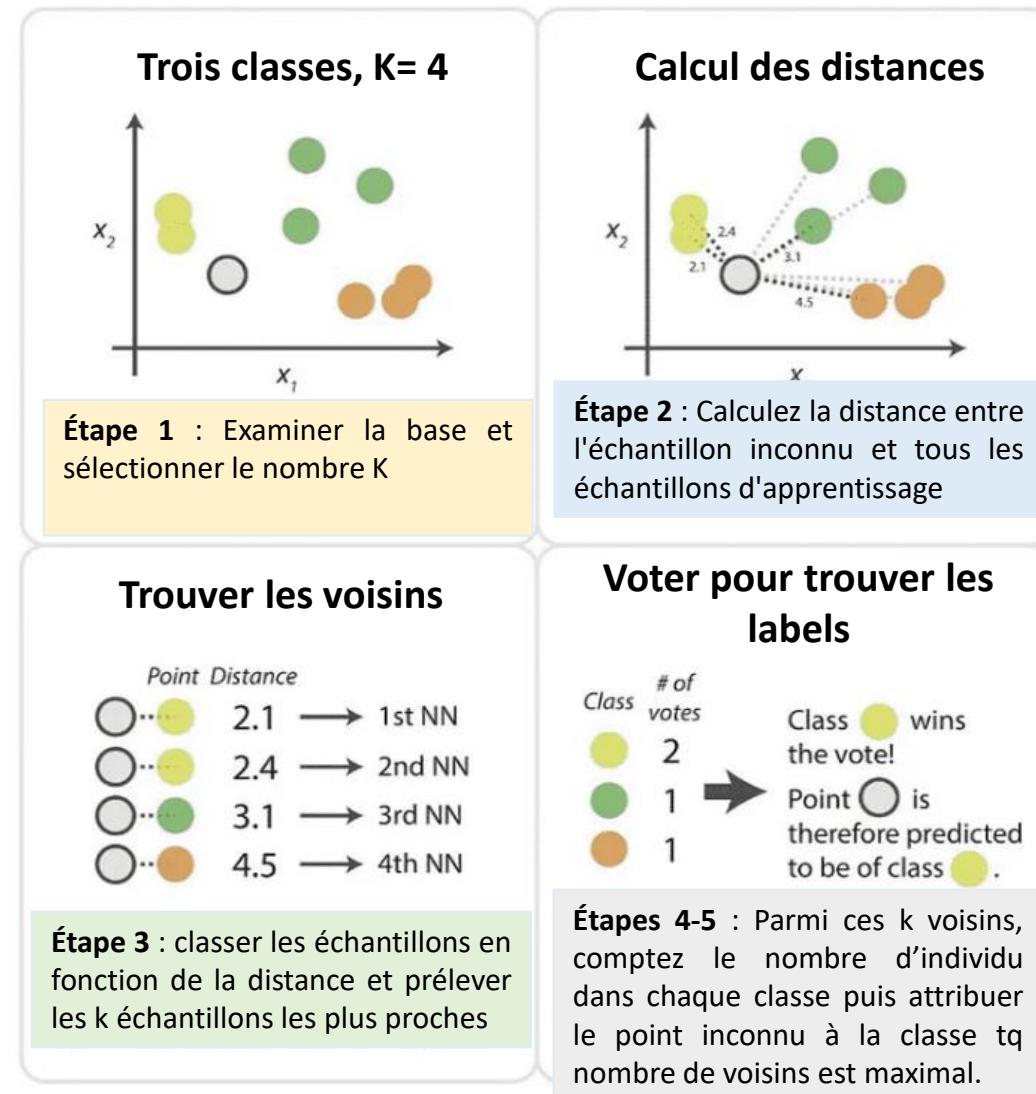
Principe : vote majoritaire

- Chaque fois que nous avons une nouvelle donnée à classer, nous trouvons ses K voisins les plus proches à partir des données d'apprentissage



Algorithme KNN de classification (2/5)

- Étape 1 : Sélectionnez le nombre K de voisins
- Étape 2 : Calculez la distance entre l'échantillon inconnu et tous les échantillons d'apprentissage
- Étape 3 : classer les échantillons en fonction de la distance et prélever les k échantillons les plus proches
- Étape 4 : Parmi ces k voisins, comptez le nombre d'individu dans chaque classe.
- Étape 5 : attribuer le nouveau point inconnu à la classe pour laquelle le nombre de voisins est maximal.



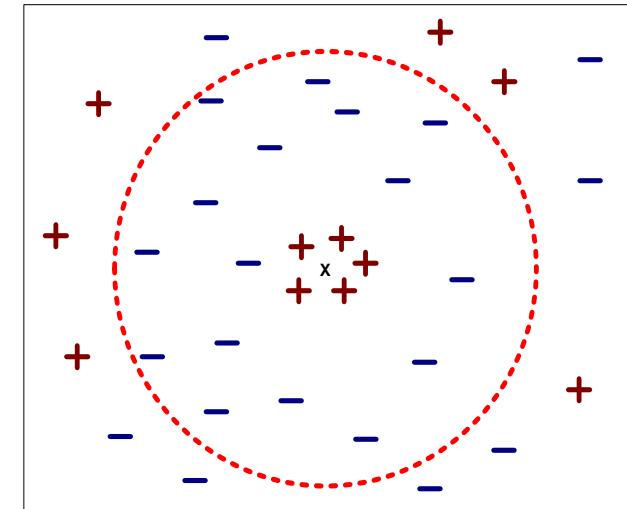
Algorithme KNN de classification (3/5)

- Le succès de l'algorithme KNN va dépendre de trois facteurs :

1 La quantité de données d'entraînement : plus la base d'apprentissage est grande, plus la performance est meilleure

2 Le choix du nombre de voisins K :

- Si K est très petit, le model sera sensible au bruit
- Si K est très grand, le voisinage peut inclure des points d'autres classes



Algorithme KNN de classification (4/5)

③ La mesure de distance : en pratique, on utilise souvent la distance Euclidienne:

Minkowsky:

$$D(x, y) = \left(\sum_{i=1}^m |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(x, y) = \sqrt{\sum_{i=1}^m (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(x, y) = \sum_{i=1}^m |x_i - y_i|$$

Camberra:

$$D(x, y) = \sum_{i=1}^m \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(x, y) = \max_{i=1}^m |x_i - y_i|$$

x et y sont des vecteurs
de m attributs

- Remarque : les attributs peuvent être dans des échelles différentes → le résultat du KNN peut être affecté

Normalisation Min-Max

$$x_i := \frac{x_i - \min}{\max - \min}$$

Normaliser dans [0, 1]

Normalisation Mean

$$x_i := \frac{x_i - \mu_i}{\max - \min}$$

Normaliser dans [-1, 1]

Normalisation Z-score

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

Normaliser dans
[-d, d]

all features will have a mean of
0 and a standard deviation of 1



Algorithme KNN de classification (5/5)

KNN

- Sensible à la variation de K
- Lent et coûteux en termes de mémoire
- Très sensible aux attributs non pertinents
- Ne convient pas si les données d'apprentissage ont peu d'exemples



Exercice

KNN

- Quelle sera la classe de X_6 ayant comme attributs : $<3,12,4,7,8>$?

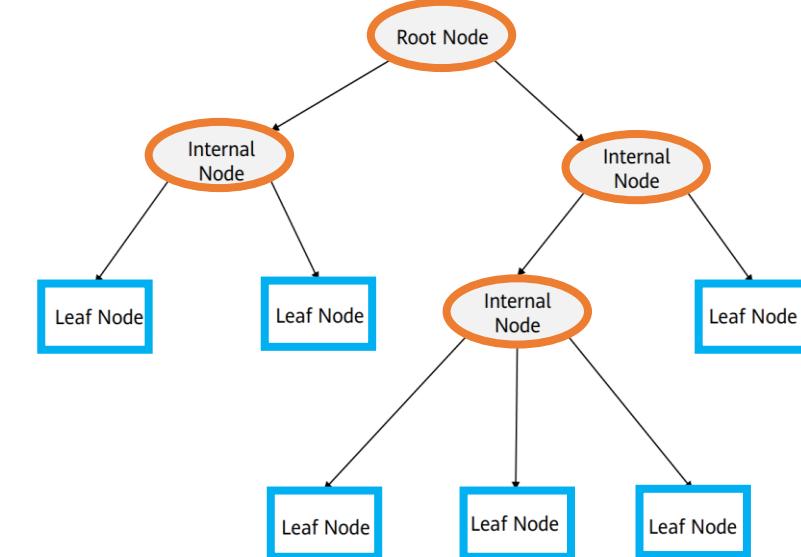
- Appliquer l'algorithme de classification K-NN en faisant varier le paramètre K (avec $K=1$ puis $K=3$) et en faisant varier la distance (distance euclidienne puis distance de Manhattan).
 - discuter les résultats

- Base d'apprentissage :

Instances	A1	A2	A3	A4	A5	Classe
X_1	3	5	4	6	1	1
X_2	4	6	10	3	2	2
X_3	8	3	4	2	6	3
X_4	2	1	4	3	6	3
X_5	2	5	1	4	8	2

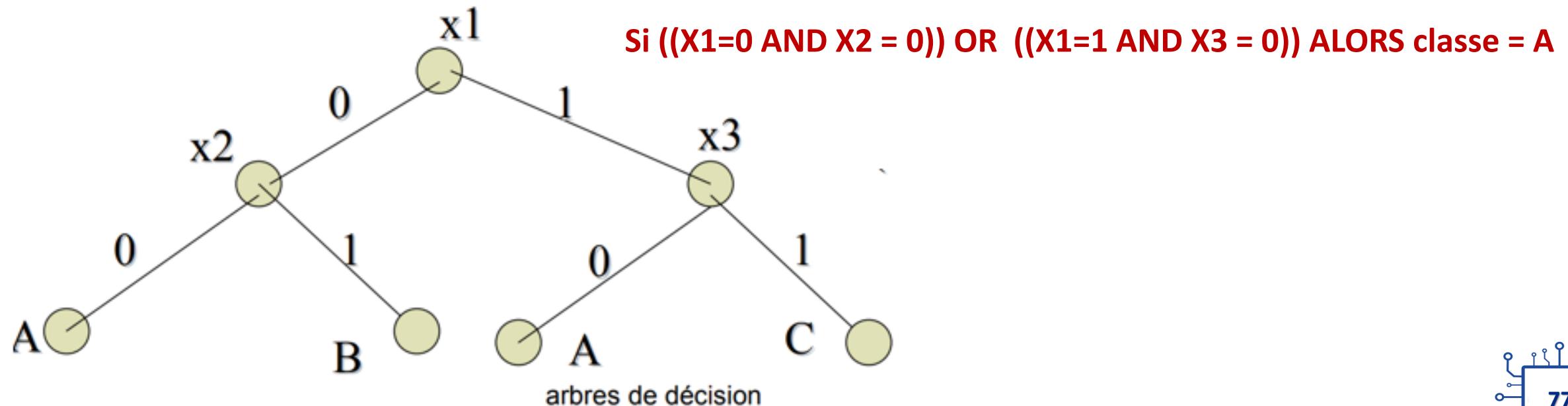
Algorithme arbre de décision (1/5)

- Les arbres de décision représentent une méthode très efficace de classification supervisée
- Il s'agit de partitionner un ensemble de données en des groupes les plus homogènes possible du point de vue. On prend en entrée un ensemble de données ayant des attributs différents et on fournit en sortie un **arbre** où :
 - chaque nœud final (feuille ou leaf) représente une décision (une classe)
 - chaque feuille représente la décision d'appartenance à une classe des données vérifiant tous les tests du chemin menant de la racine à cette feuille
 - chaque nœud non final (interne) représente un test sur un attribut.



Algorithme arbre de décision

- Les arbres de décision adoptent une représentation DNF : Disjunctive Normal Form.
 - Pour une classe donnée, chaque branche déployée de la racine vers une feuille de la dite classe, est une conjonction de valeurs d'attributs.
 - Les différentes branches se terminant vers cette classe forment une disjonction.



Algorithme arbre de décision (1/5)

- Pour créer un arbre de décision, nous devons sélectionner des attributs et déterminer la structure arborescente entre les attributs :
- L'idée clé de la construction d'un arbre de décision est de :
 - 1 Choisir le meilleur attribut,
 - 2 Construire des branches de l'arbre en rajoutant une nouvelle branche pour chaque valeur de l'attribut sélectionné
 - 3 Répartir les exemples d'apprentissage sur les feuilles de l'arbre
 - 4 Si tous les exemples d'apprentissages sont bien classés : Arrêter la construction de l'arbre, sinon, Répéter les étapes 1-4 pour les autres attributs
- Le choix du meilleur attribut se fait en fonction de la pureté de celui-ci
- Les algorithmes d'arbre de décision les plus utilisés : ID3, C4.5 et CART

Algorithme arbre de décision (1/5)

- Les mesures permettant de quantifier la « pureté » incluent l'entropie et l'indice GINI :

$$H(X) = -\sum_{k=1}^K p_k \log_2(p_k)$$

$$Gini = 1 - \sum_{k=1}^K p_k^2$$

où p_k indique la probabilité que l'exemple appartient à la classe k (il y a K classe au total).

- Étant donné l'entropie comme mesure de l'impureté dans une base d'apprentissage, nous pouvons maintenant définir une mesure de l'importance d'un attribut : **gain d'information**

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

où $Values(A)$ est l'ensemble de toutes les valeurs possibles de l'attribut A, et S_v est le sous-ensemble de S pour lequel l'attribut A a la valeur v

- Le gain d'information est la réduction attendue de l'entropie provoquée par le partitionnement des exemples selon l'attribut A.

Algorithme arbre de décision (1/5)

L'algorithme ID3

- L'algorithme ID3 (Iterative Dichotomiser 3) produit un arbre de décision qui servira à classer les nouveaux échantillons.
- Il construit l'arbre en commençant par chercher l'**attribut qui doit être sélectionné à la racine de l'arbre**, pour le savoir, il faut :
 1. Calculer l'entropie du nœud racine (Entropie(S) de la racine)
 2. Pour chaque attribut A dans S, calculez le gain d'information en utilisant ses différentes valeurs v
 3. Sélectionnez l'attribut ayant fourni le gain le plus élevé : il formera le nœud suivant dans l'arborescence.
 4. Supprimer l'attribut sélectionné de la liste des attributs pour créer un ensemble de données réduit
 5. Répétez les étapes 1 à 4 jusqu'à ce que tous les attributs soient sélectionnés ou que un nœud feuille soit atteint (où tous les exemples sont soit positifs soit négatifs)

Exemple : arbre de décision avec l'algorithme ID3 (1/10)

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Base d'apprentissage pour l'exemple Play Tennis, source : Textbook

Exemple : arbre de décision avec l'algorithme ID3 (2/10)

Étape 1 : Entropie (S) ?

Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

$$\begin{aligned}Entropy([9+, 5-]) &= -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) \\&= 0.940\end{aligned}$$

Exemple : arbre de décision avec l'algorithme ID3 (3/10)

Étape 2 : information gain pour chaque attribut

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Gain(S, Outlook) =$$

$$Gain(S, Humidity) =$$

$$Gain(S, Wind) =$$

$$Gain(S, Temperature) =$$



Exemple : arbre de décision avec l'algorithme ID3 (4/10)

Étape 2 : information gain pour chaque attribut

Attribut = outlook

v = Sunny, Overcast, Rain

$$S_{Sunny} \leftarrow [2+, 3-]$$

$$\text{Entropy}(S_{Sunny}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$

$$S_{Overcast} \leftarrow [4+, 0-]$$

$$\text{Entropy}(S_{Overcast}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$$

$$S_{Rain} \leftarrow [3+, 2-]$$

$$\text{Entropy}(S_{Rain}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

$$\text{Gain}(S, \text{Outlook}) = \text{Entropy}(S) - \sum_{v \in \{\text{Sunny}, \text{Overcast}, \text{Rain}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$\text{Gain}(S, \text{Outlook})$

$$= \text{Entropy}(S) - \frac{5}{14} \text{Entropy}(S_{Sunny}) - \frac{4}{14} \text{Entropy}(S_{Overcast})$$

$$-\frac{5}{14} \text{Entropy}(S_{Rain})$$

→ Gain(S, Outlook) = 0.246

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Exemple : arbre de décision avec l'algorithme ID3 (5/10)

Étape 2 : information gain pour chaque attribut

Attribut = Temperature

v = Hot, Mild, Cool



$$Gain(S, Temperature) = 0.029$$

Attribut = Humidity

v = High, Normal



$$Gain(S, Humidity) = 0.151$$

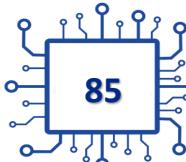
Attribut = Wind

v = Weak, Strong



$$Gain(S, Wind) = 0.048$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Exemple : arbre de décision avec l'algorithme ID3 (6/10)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

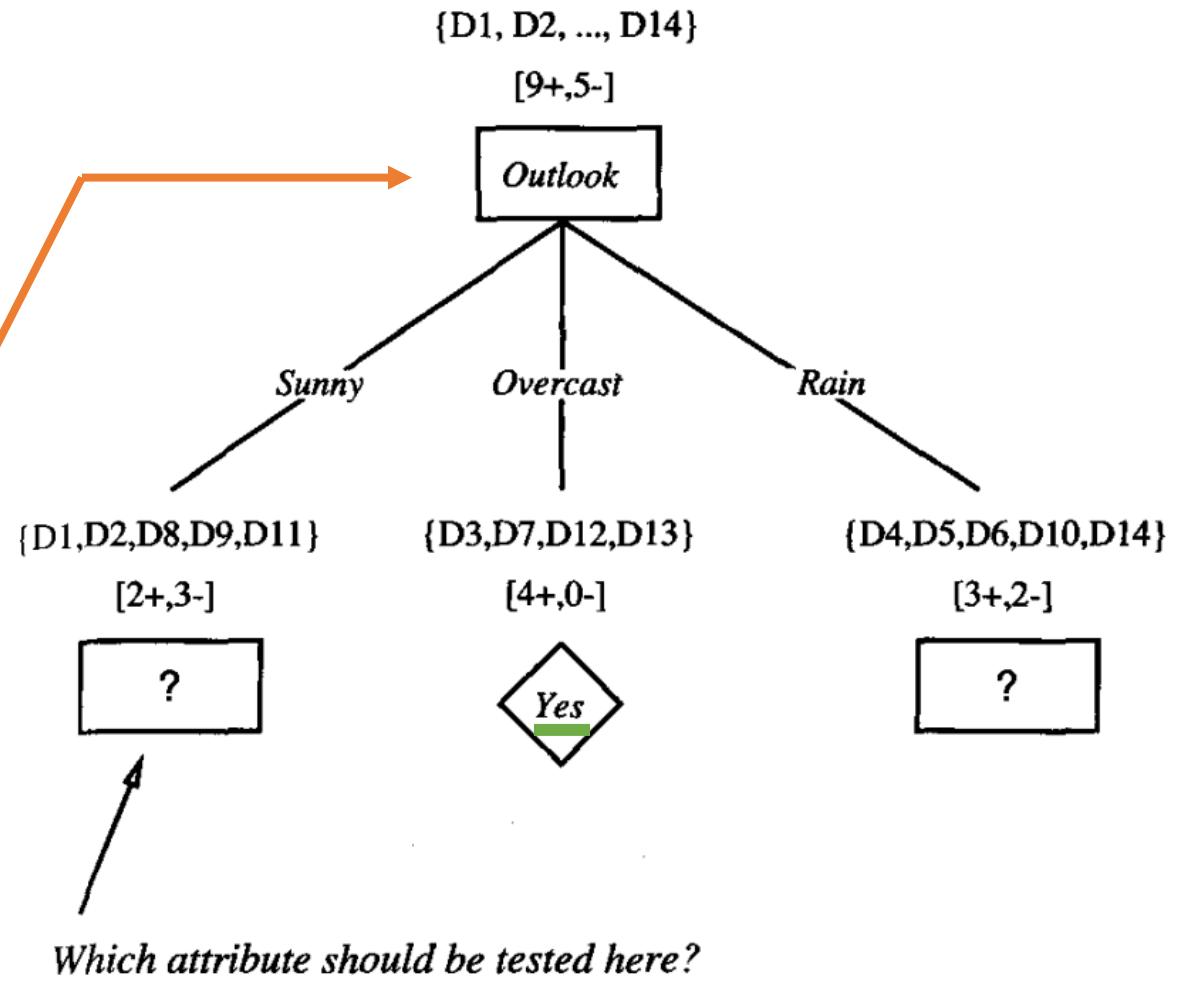
$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

Étape 3 : Sélectionner attribut ayant le gain le plus élevé



Exemple : arbre de décision avec l'algorithme ID3 (7/10)

Répéter 1 - 4

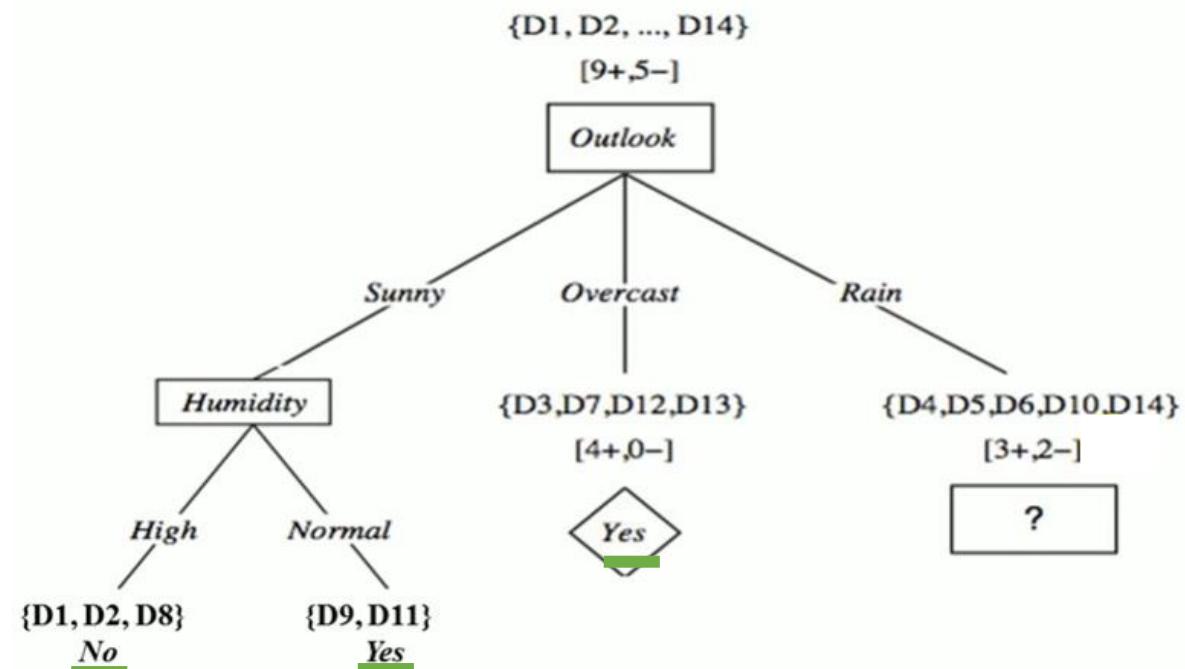
Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$$S_{sunny} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{sunny}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{sunny}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{sunny}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$



Exemple : arbre de décision avec l'algorithme ID3 (8/10)

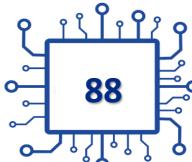
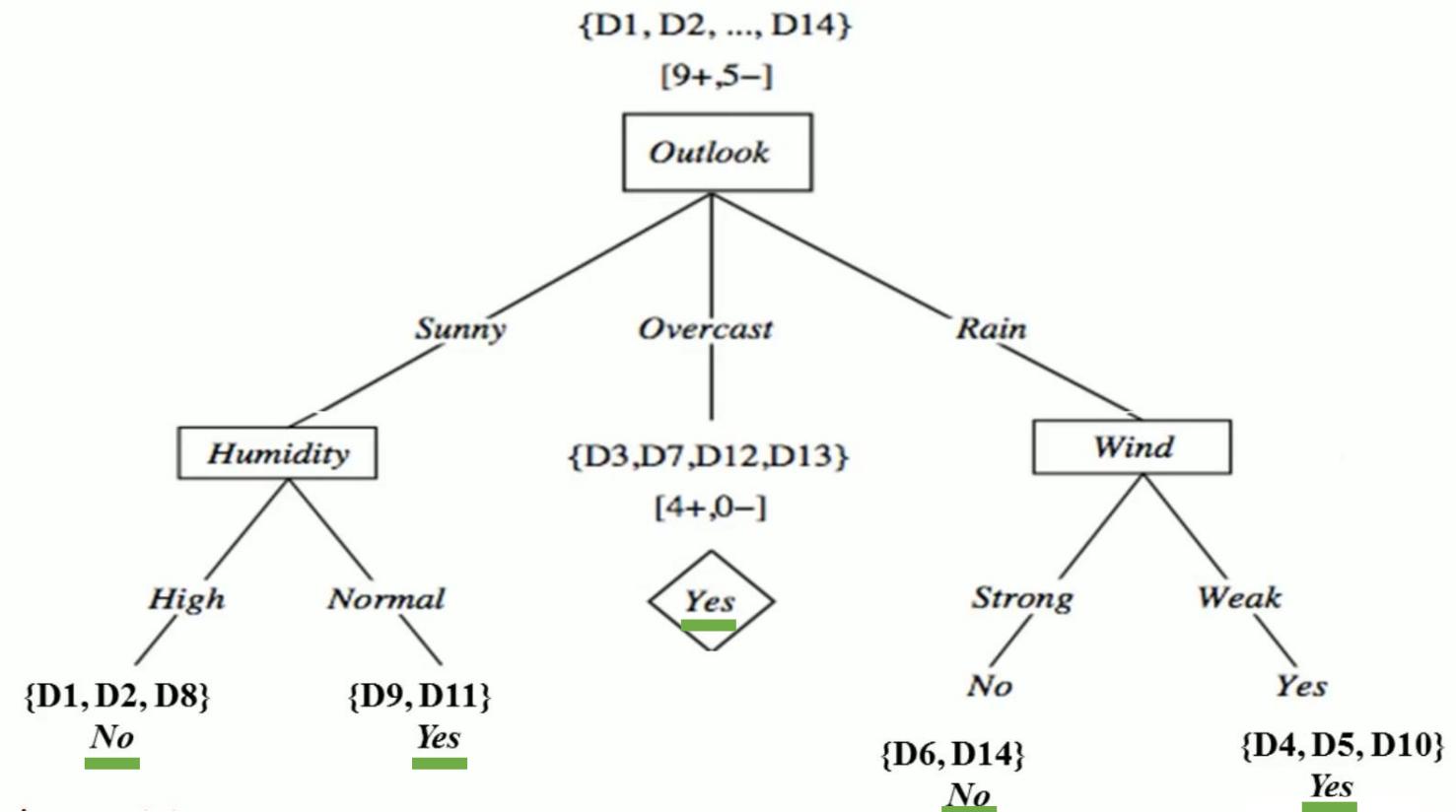
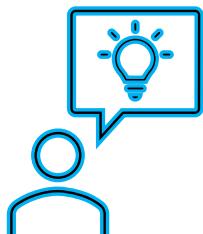
Répéter 1 - 4

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

$$Gain(S_{Rain}, Temp) = 0.0192$$

$$Gain(S_{Rain}, Humidity) = 0.0192$$

$$Gain(S_{Rain}, Wind) = 0.97$$



Exemple : arbre de décision avec l'algorithme ID3 (9/10)

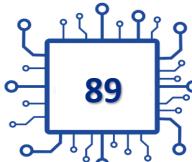
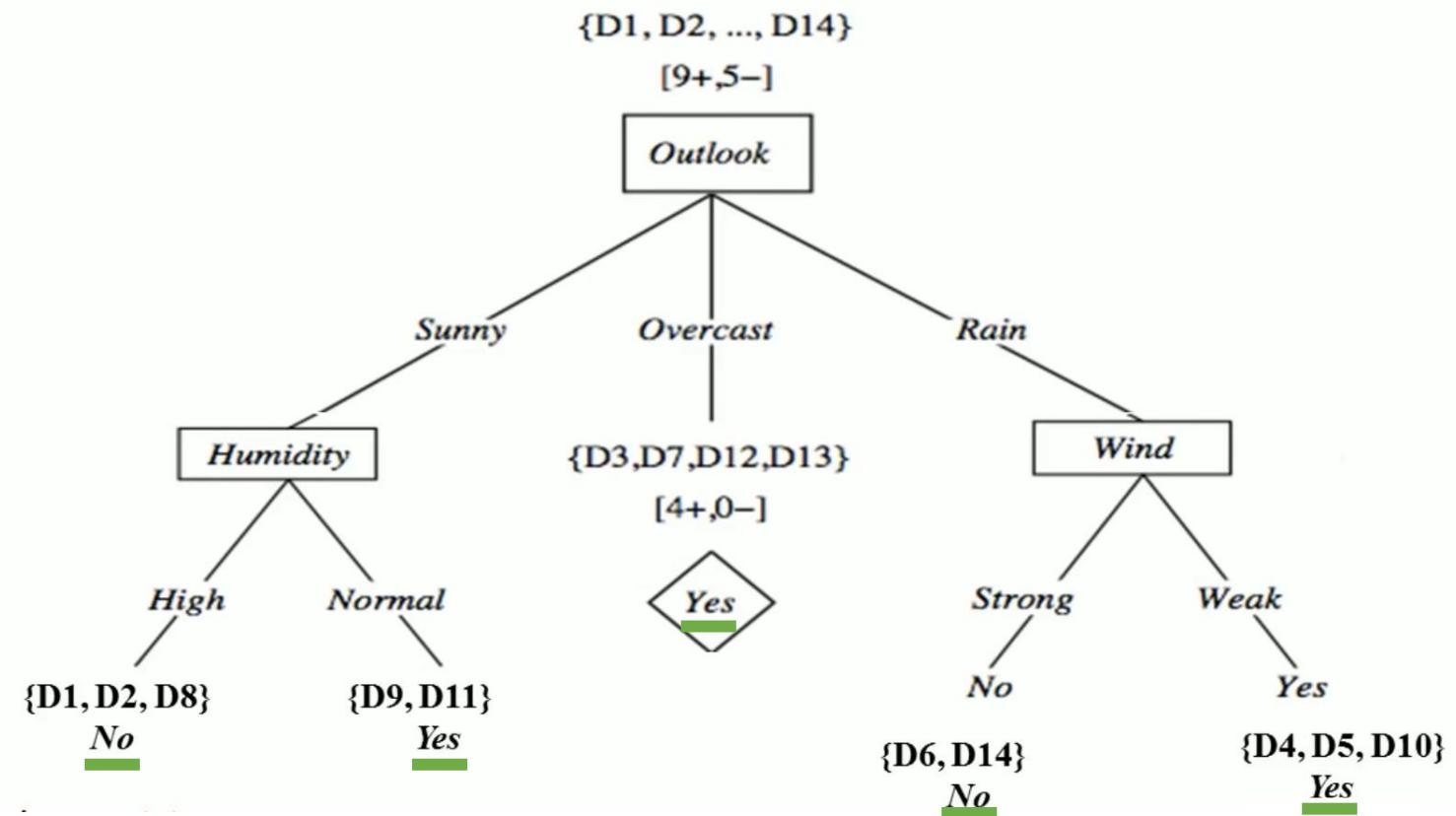
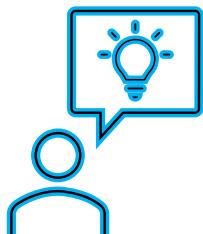
Répéter 1 - 4

Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

$$Gain(S_{Rain}, Temp) = 0.0192$$

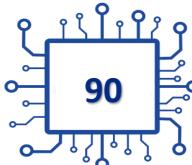
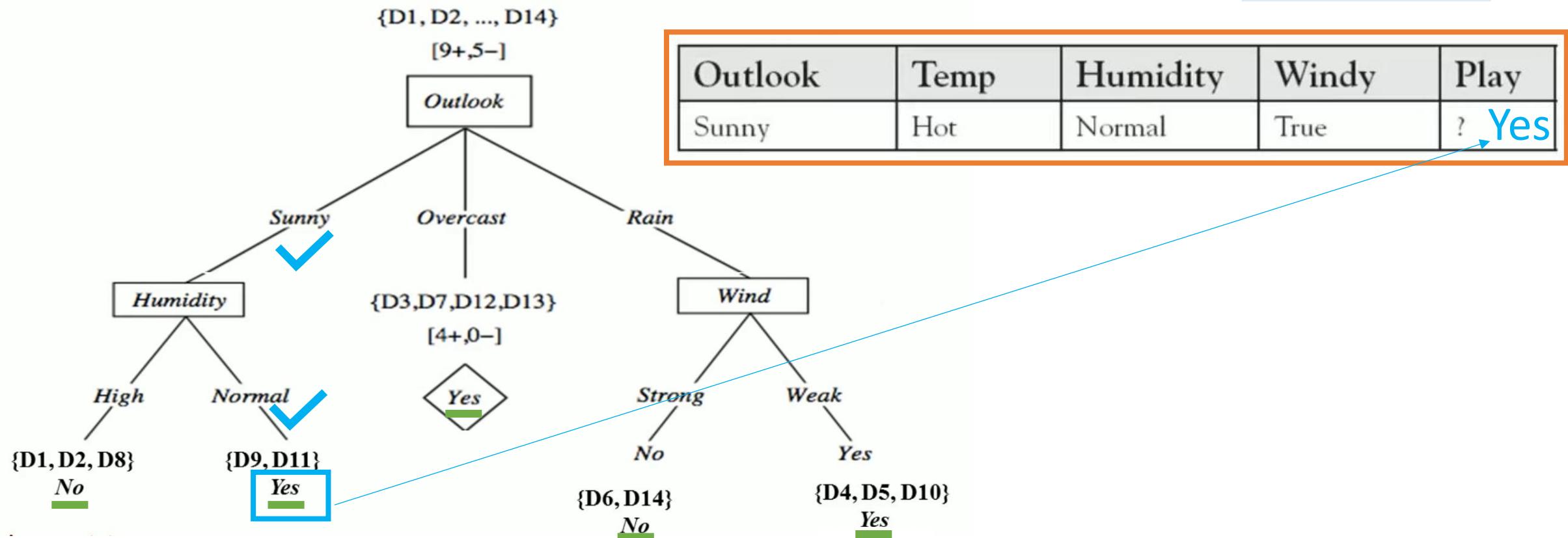
$$Gain(S_{Rain}, Humidity) = 0.0192$$

$$Gain(S_{Rain}, Wind) = 0.97$$



Exemple : arbre de décision avec l'algorithme ID3 (10/10)

Étape de test



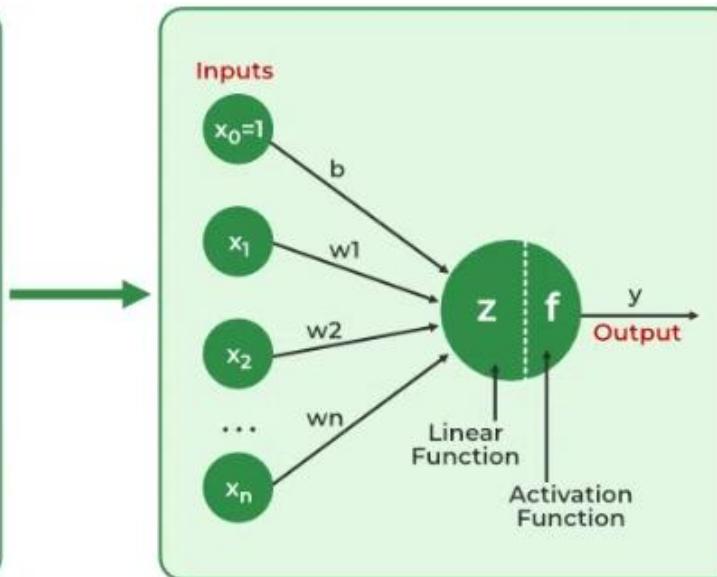
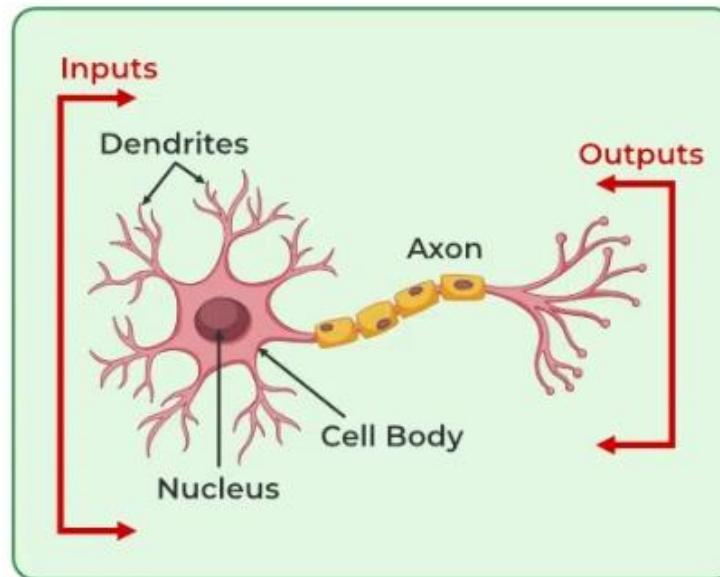
Motivation

- Les réseaux de neurones sont une série d'algorithmes qui **imitent** les opérations du cerveau humain pour reconnaître les relations entre de grandes quantités de données.
→ **ils imitent les connexions entre les neurones et les synapses du cerveau humain**
- Les réseaux de neurones comportant plusieurs couches sont appelés réseaux « profonds » et sont utilisés dans les algorithmes d'apprentissage profond
- Ils sont utilisés dans des applications diverses :
 - forecasting and marketing research, fraud detection and risk assessment, pattern recognition, text translation, paper summarization, industrial controls, energy prediction, ...

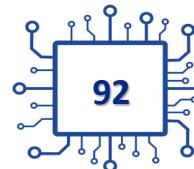


Neurones Artificiels vs neurone biologique

- La structure des réseaux de neurones artificiels est inspirée des neurones biologiques:
 - Un neurone biologique possède un corps cellulaire ou soma pour traiter les impulsions, des dendrites pour les recevoir et un axone qui les transfère aux autres neurones.
 - chaque neurone artificiel reçoit un certain nombre de signaux d'entrées provenant des neurones en amont. À chacune de ces entrées, est associé un poids qui représente la force de la connexion entre les deux neurones. Chaque neurone est doté d'une sortie unique.



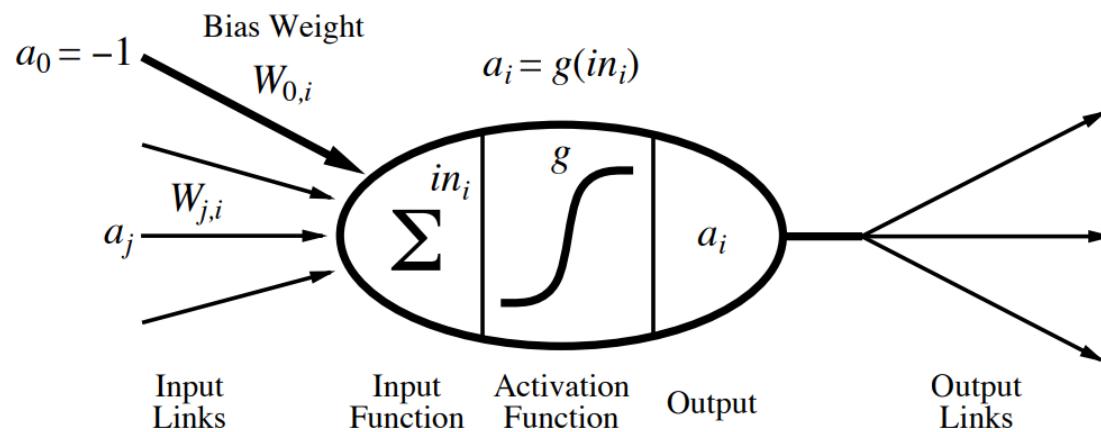
Biological Neuron	Artificial Neuron
Dendrite	Inputs
Cell nucleus or Soma	Nodes
Synapses	Weights
Axon	Output



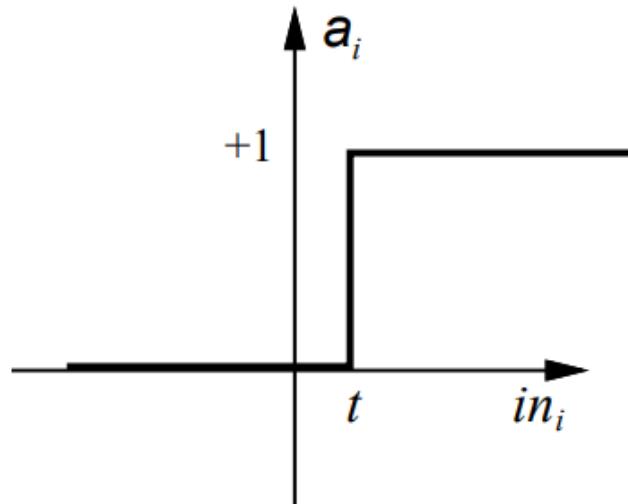
Unité dans un réseau de neurones

- Un réseau de neurones artificiels est un graphe composé de cellules ou unités, connectés par des liens :
 - Chaque lien est associé à un poids (weight)
 - Généralement, chaque nœud i génère un nombre réel, qui est transmis en entrée aux nœuds connectés à i .
 - La sortie d'un nœud est fonction de la somme pondérée des entrées du nœud \approx fonction des combinaisons linéaires d'entrées

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

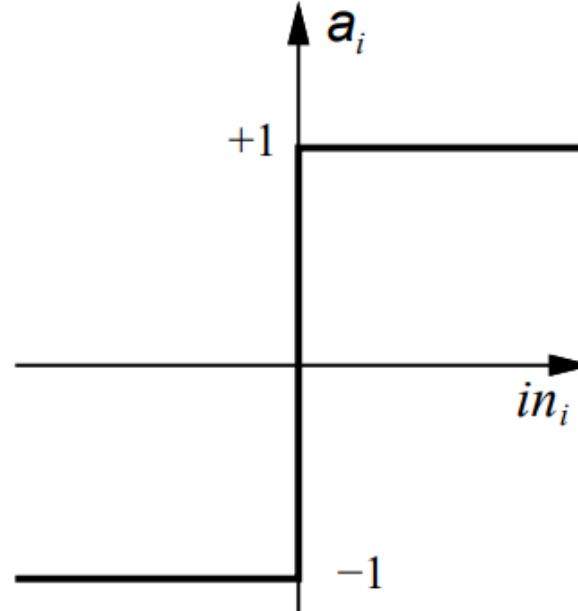


Exemples de fonctions d'activation



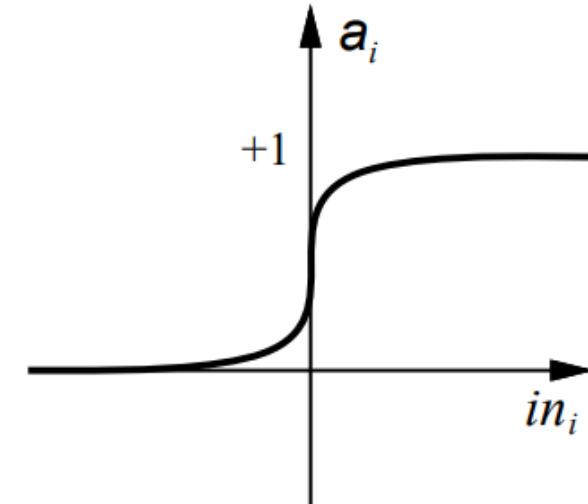
(a) Step function

$$step_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases}$$



(b) Sign function

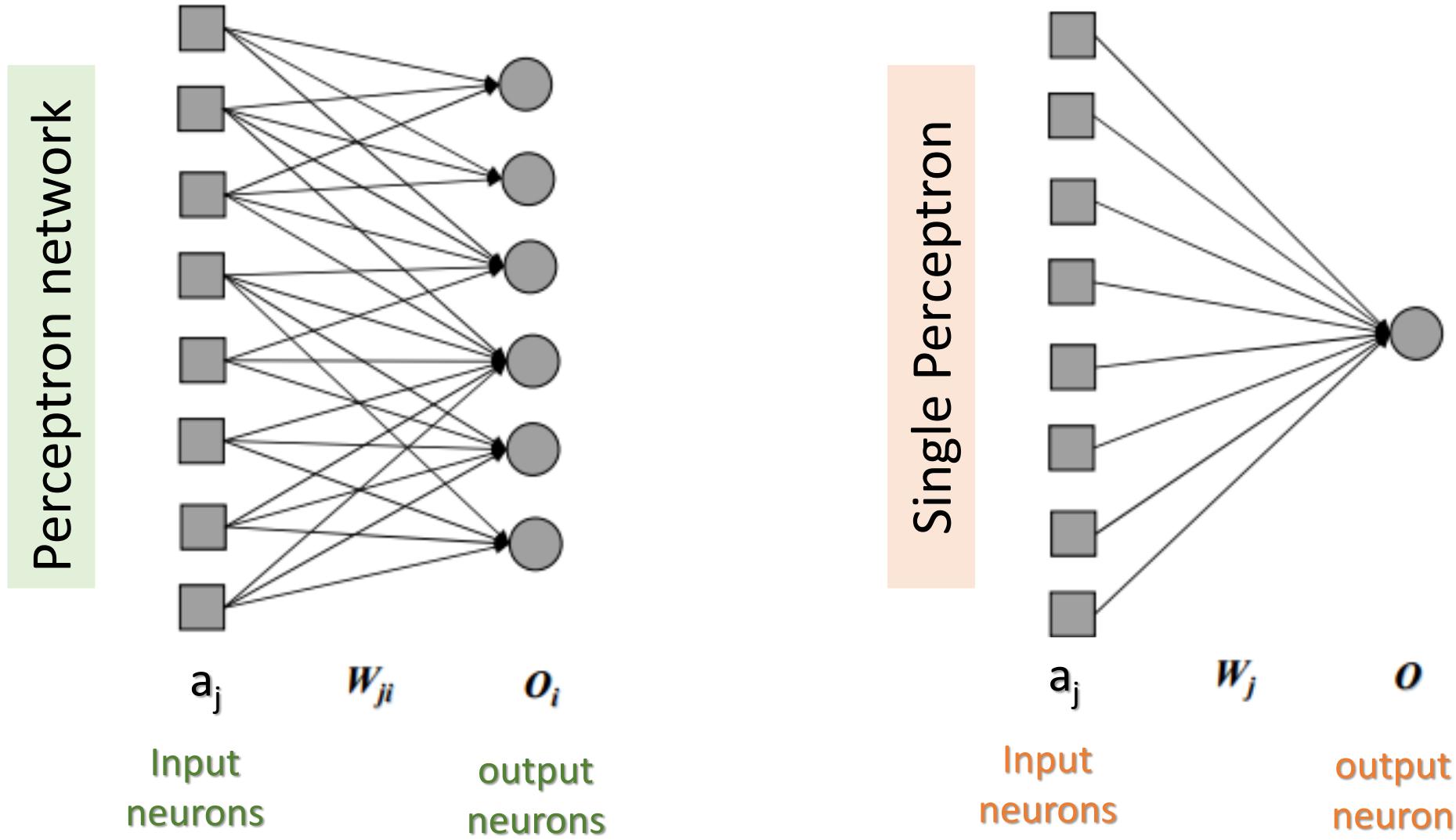
$$sign(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$$



(c) Sigmoid function

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

Perceptron network vs single layer Perceptron



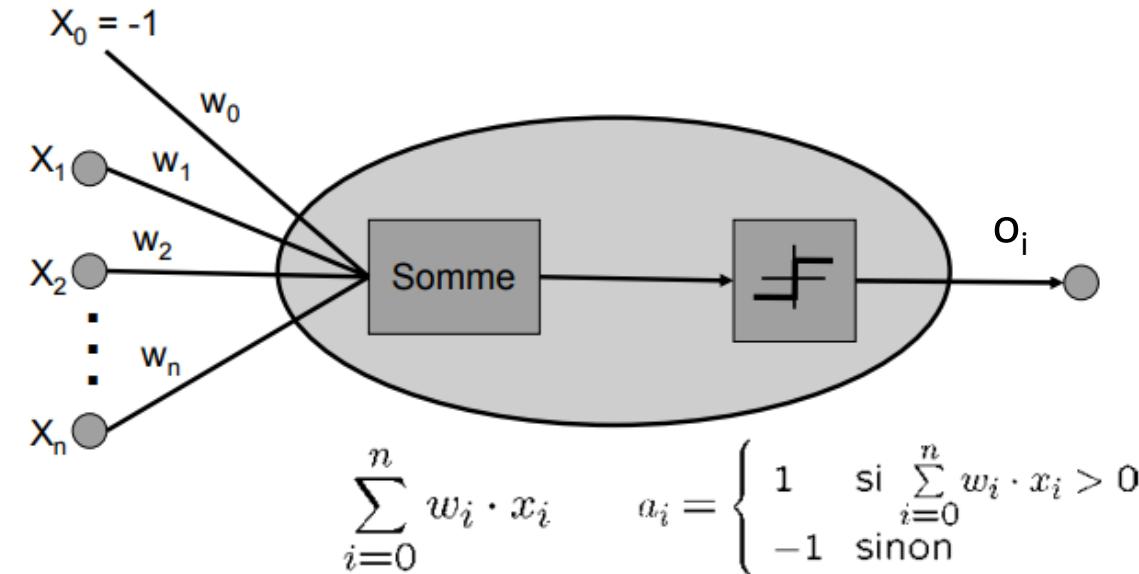
Single-layer neural network : Perceptron

- C'est quoi un perceptron ?

- Le perceptron est le type de réseau de neurones le plus simple : réseau monocouche à une seule sortie.
- Le perceptron est un algorithme d'apprentissage supervisé de classifieurs binaire.
- Sert à modéliser la décision à l'aide d'une fonction linéaire, suivi d'un seuil

- On considère un perceptron comme un neurone à deux entrées

Comment calculer et mettre à jour les poids?



$$o_i = \text{sgn}(W \cdot X + b) = \text{sgn}(w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_0)$$

Règle d'apprentissage du perceptron



Règle d'apprentissage : Perceptron learning rule

- Learning rule : démarche pour modifier les poids dans le réseau.

Algorithm: Perceptron learning rule

```
1: Let  $D = \{(X_i, t_i) | i = 1, 2, \dots, n\}$  be the set of training instances.  
2: Initialize the weight vector with random values,  $\omega$   
3: repeat  
4:   for each training instance  $(x_i, t_i) \in D$   
5:     Compute the predicted output  $o_i$   
6:     for each weight  $\omega_j$   
7:       Update the weight,  $\omega_j = \omega_j + \eta(t_i - o_i)x_{ij}$   
8:     end for  
9:   end for  
10: until stopping condition is met
```

$$\eta(t_i - o_i)x_{ij}$$

learning rate target value perceptron output input value

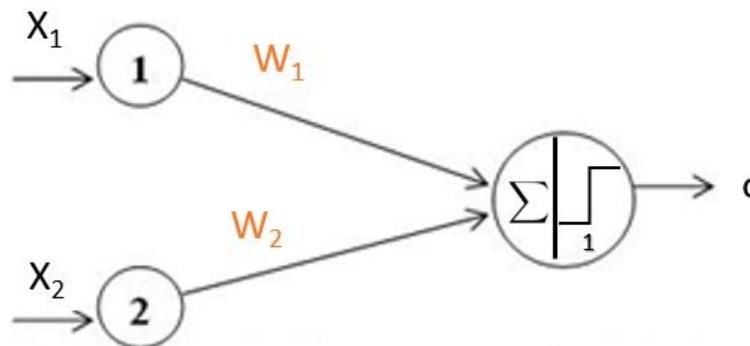


Le changement de poids est égal à la différence entre la sortie réelle et la sortie prédictive multipliée par l'entrée et le taux d'apprentissage.

Exemple 1 : AND gate (1/3)

- Comment concevoir AND gate en utilisant la règle d'apprentissage du perceptron (sans biais) avec :

- poids initiaux: $w_1 = 1.2$, $w_2 = 0.6$
- Threshold = 1
- learning rate = 0.5



X_1	X_2	t
0	0	0
0	1	0
1	0	0
1	1	1

- Learning rule :

- i=1 : $X_1 = 0, X_2 = 0$ and $t_1 = 0$
 - $w_1 \cdot X_1 + w_2 \cdot X_2 = 1.2 \times 0 + 0.6 \times 0 = 0$ not greater than 1 $\rightarrow o_1 = 0$
 - $W_1\text{new} = w_1 + \eta(t_1 - o_1)X_1 = 1.2 + 0.5 \times (0 - 0) \times 0 = 1.2$ no update.
 - $W_2\text{new} = w_2 + \eta(t_1 - o_1)X_2 = 0.6 + 0.5 \times (0 - 0) \times 0 = 0.6$ no update.

- i=2 : $X_1 = 0, X_2 = 1$ and $t_2 = 0$
 - $w_1 \cdot X_1 + w_2 \cdot X_2 = 1.2 \times 0 + 0.6 \times 1 = 0.6$ not greater than 1 $\rightarrow o_2 = 0$
 - $W_1\text{new} = w_1 + \eta(t_2 - o_2)X_1 = 1.2 + 0.5 \times (0 - 0) \times 0 = 1.2$ no update.
 - $W_2\text{new} = w_2 + \eta(t_2 - o_2)X_2 = 0.6 + 0.5 \times (0 - 0) \times 1 = 0.6$ no update.

Exemple 1 : AND gate (2/3)

□ i=3 : $X_1 = 1, X_2 = 0$ and $t_3 = 0$

- $w_1 \cdot X_1 + w_2 \cdot X_2 = 1.2 \times 1 + 0.6 \times 0 = 1.2$ greater than 1 $\rightarrow o_3 = 1$
- $W_1\text{new} = w_1 + \eta(t_3 - o_3)X_1 = 1.2 + 0.5 \times (0 - 1) \times 1 = 0.7$ updated weight
- $W_2\text{new} = w_2 + \eta(t_3 - o_3)X_2 = 0.6 + 0.5 \times (0 - 1) \times 0 = 0.6$ no update.

□ i=1 : $X_1 = 0, X_2 = 0$ and $t_1 = 0$

- $w_1 \cdot X_1 + w_2 \cdot X_2 = 0.7 \times 0 + 0.6 \times 0 = 0$ not greater than 1 $\rightarrow o_1 = 0$
- $W_1\text{new} = w_1 + \eta(t_1 - o_1)X_1 = 0.7 + 0.5 \times (0 - 0) \times 0 = 0.7$ no update.
- $W_2\text{new} = w_2 + \eta(t_1 - o_1)X_2 = 0.6 + 0.5 \times (0 - 0) \times 0 = 0.6$ no update.

□ i=2 : $X_1 = 0, X_2 = 1$ and $t_2 = 0$

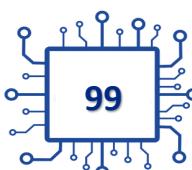
- $w_1 \cdot X_1 + w_2 \cdot X_2 = 0.7 \times 0 + 0.6 \times 1 = 0.6$ not greater than 1 $\rightarrow o_2 = 0$
- $W_1\text{new} = w_1 + \eta(t_2 - o_2)X_1 = 0.7 + 0.5 \times (0 - 0) \times 0 = 0.7$ no update.
- $W_2\text{new} = w_2 + \eta(t_2 - o_2)X_2 = 0.6 + 0.5 \times (0 - 0) \times 1 = 0.6$ no update.

□ i=3 : $X_1 = 1, X_2 = 0$ and $t_3 = 0$

- $w_1 \cdot X_1 + w_2 \cdot X_2 = 0.7 \times 1 + 0.6 \times 0 = 0.7$ not greater than 1 $\rightarrow o_3 = 0$
- $W_1\text{new} = w_1 + \eta(t_3 - o_3)X_1 = 0.7 + 0.5 \times (0 - 0) \times 1 = 0.7$ no update.
- $W_2\text{new} = w_2 + \eta(t_3 - o_3)X_2 = 0.6 + 0.5 \times (0 - 0) \times 0 = 0.6$ no update.



- Poids initiaux :
 - $w_1 = 0.7,$
 - $w_2 = 0.6,$
- Threshold = 1
- learning rate = 0.5



Exemple 1 : AND gate (3/3)

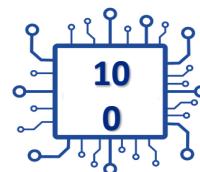
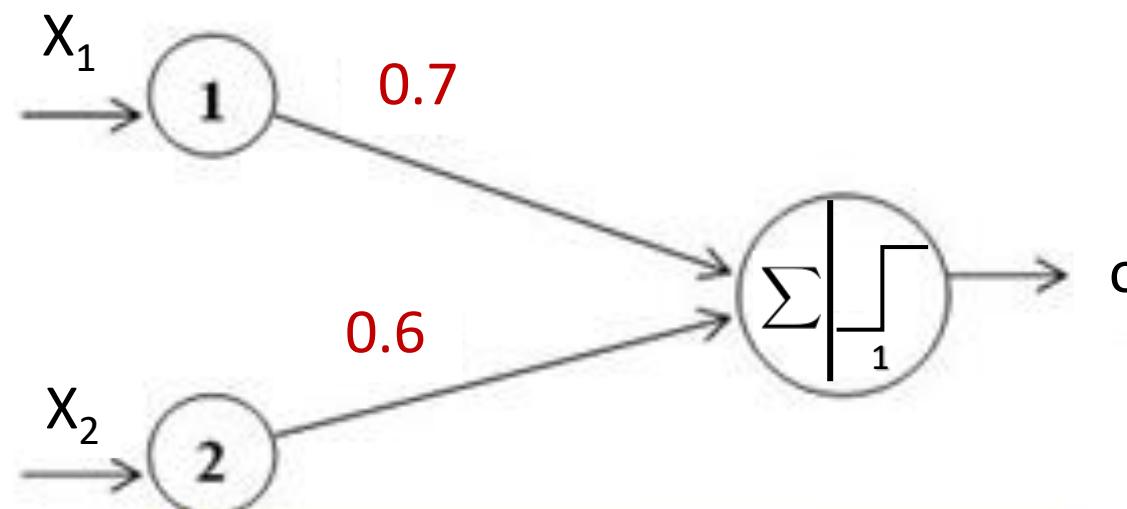
□ i=4 : $X_1 = 1$, $X_2 = 1$ and $t_4 = 1$

$$w_1 \cdot X_1 + w_2 \cdot X_2 = 0.7 \times 1 + 0.6 \times 1 = 1.3 \text{ greater than } 1 \rightarrow o_4 = 1$$

$$W_{1\text{new}} = w_1 + \eta(t_4 - o_4)X_1 = 0.7 + 0.5 \times (1 - 1) \times 1 = 0.7 \text{ no update.}$$

$$W_{2\text{new}} = w_2 + \eta(t_4 - o_4)X_2 = 0.6 + 0.5 \times (1 - 1) \times 1 = 0.6 \text{ no update.}$$

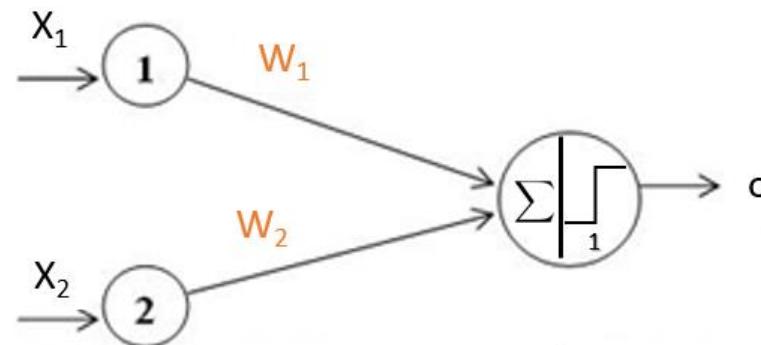
Poids finaux après
l'apprentissage de la
fonction AND :



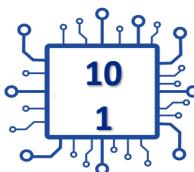
Exemple 2 : OR gate

- Comment concevoir OR gate en utilisant la règle d'apprentissage du perceptron (sans biais) avec :

- Poids initiaux : $w_1 = 0.6$, $w_2 = 0.6$
- Threshold = 1
- learning rate = 0.5



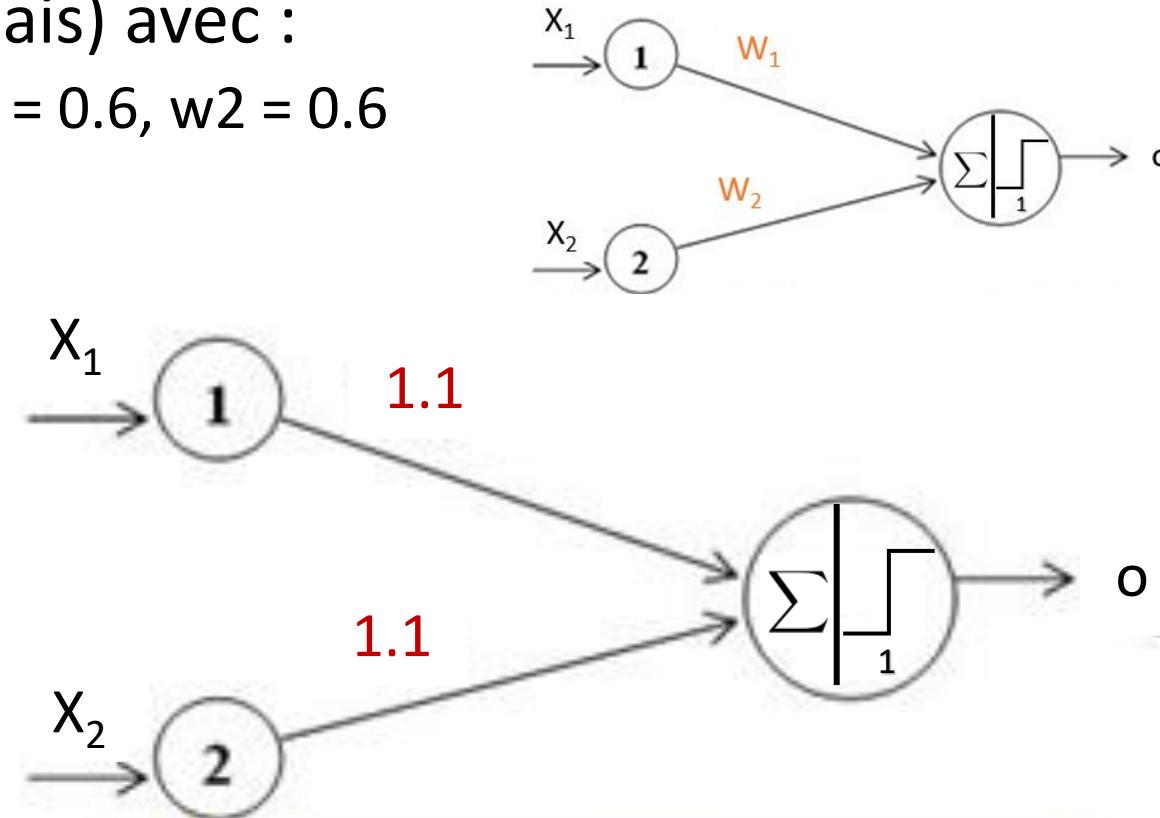
x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	1



Solution : OR gate

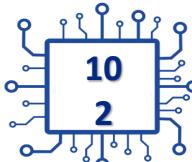
- Comment concevoir OR gate en utilisant la règle d'apprentissage du perceptron (sans biais) avec :

- Poids initiaux : $w_1 = 0.6$, $w_2 = 0.6$
- Threshold = 1
- learning rate = 0.5



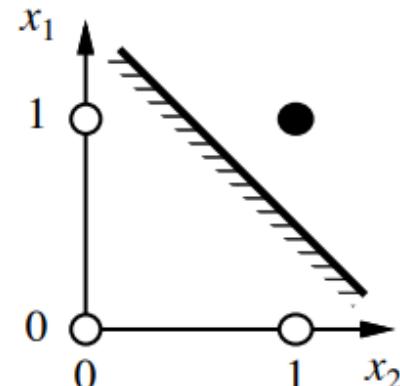
Poids finaux après
l'apprentissage de la
fonction OR :

X_1	X_2	t
0	0	0
0	1	1
1	0	1
1	1	1

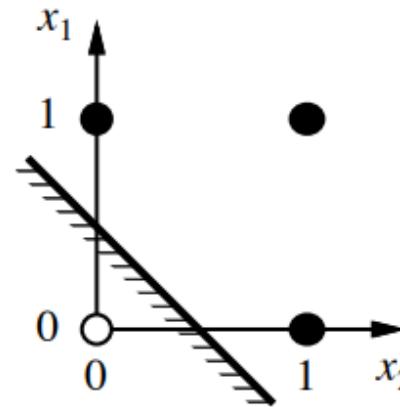


Problème avec les perceptrons

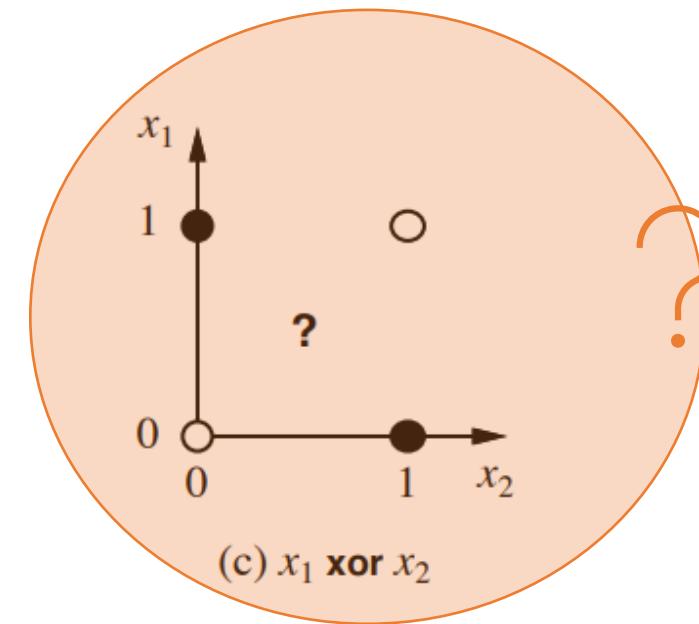
- Les perceptrons peuvent représenter des **fonctions linéairement séparables**



(a) x_1 **and** x_2

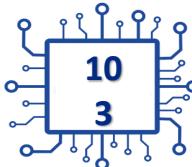


(b) x_1 **or** x_2



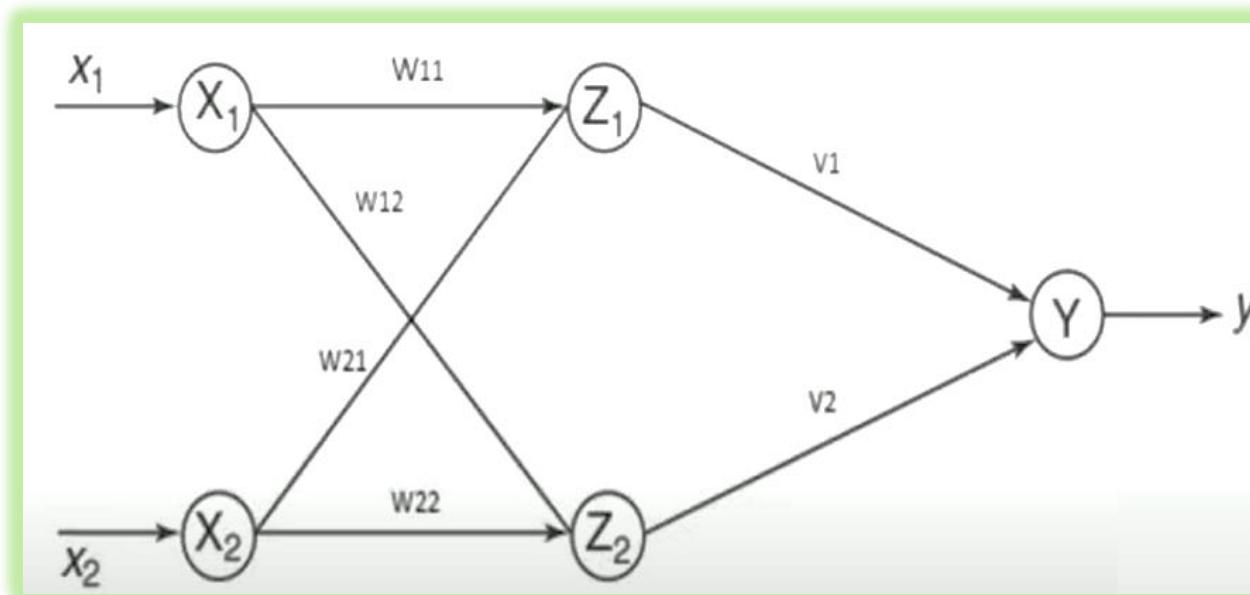
(c) x_1 **xor** x_2

- Problèmes : Toutes les fonctions ne sont pas forcément linéaires.
- Solution : Multilayer perceptrons

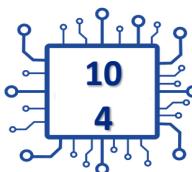


Exemple : XOR gate

- Comment concevoir XOR gate en utilisant la règle d'apprentissage du perceptron (sans biais) ?
 - XOR ne peut pas être modélisée par un seul perceptron
 - XOR n'est pas linéairement séparable donc le perceptron ne peut pas l'apprendre
 - $y = x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 = z_1 + z_2$



X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



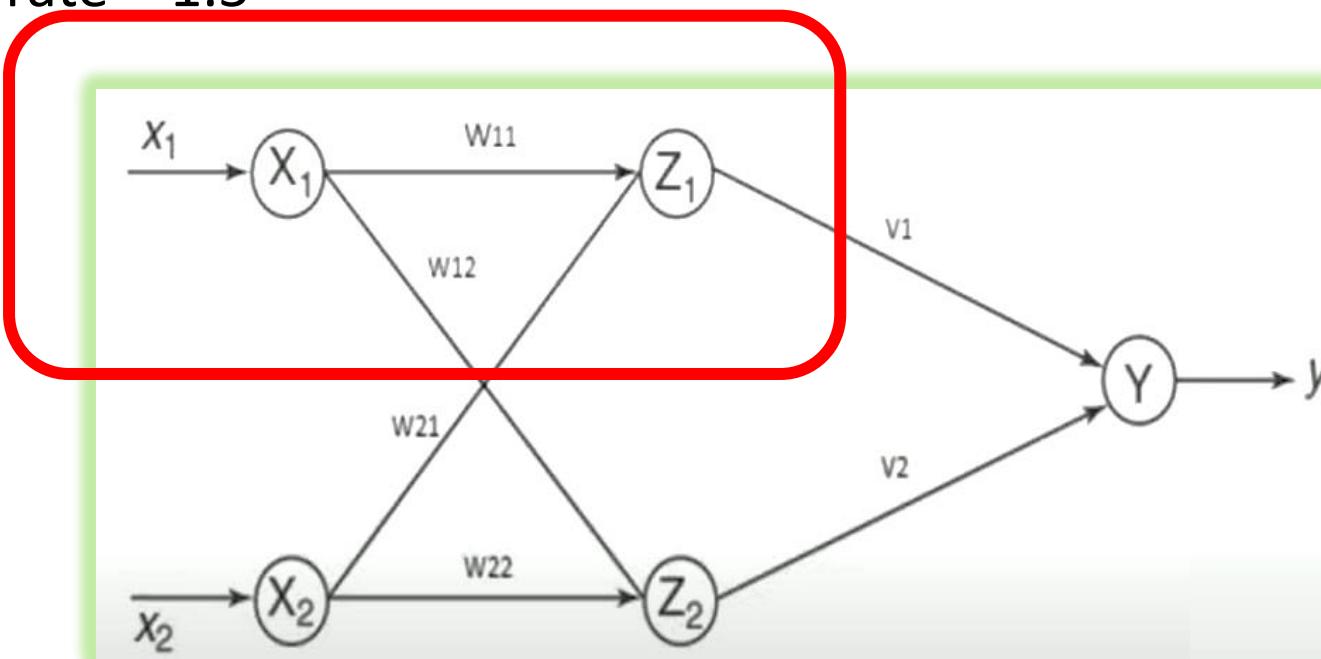
Solution : XOR gate (1/6)

- First function $Z_1 = x_1 \cdot \bar{x}_2$

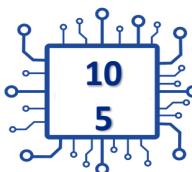
- initial weight $w_{11}=w_{21}=1$

- Threshold = 1

- learning rate = 1.5



x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

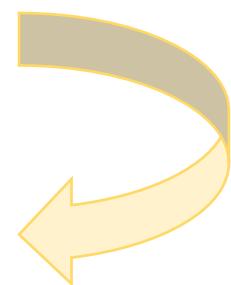


Solution : XOR gate (2/6)

- $i_1(0,0) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 0 + 1 \times 0 = 0$ (out =0 : z1t=0: no update)
- $i_2(0,1) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 0 + 1 \times 1 = 1$ (out =1 : z1t=0: updated weight)

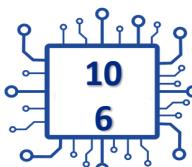
❖ $W_{11\text{new}} = w_{11} + \eta(t_2 - o_2)X_1 = 1 + 1.5 \times (0 - 1) \times 0 = 1$

❖ $W_{21\text{new}} = w_{21} + \eta(t_2 - o_2)X_2 = 1 + 1.5 \times (0 - 1) \times 1 = -0.5$



- initial weights :
 - $w_{11} = 1$,
 - $w_{21} = -0.5$,
- Threshold = 1
- learning rate = 1.5

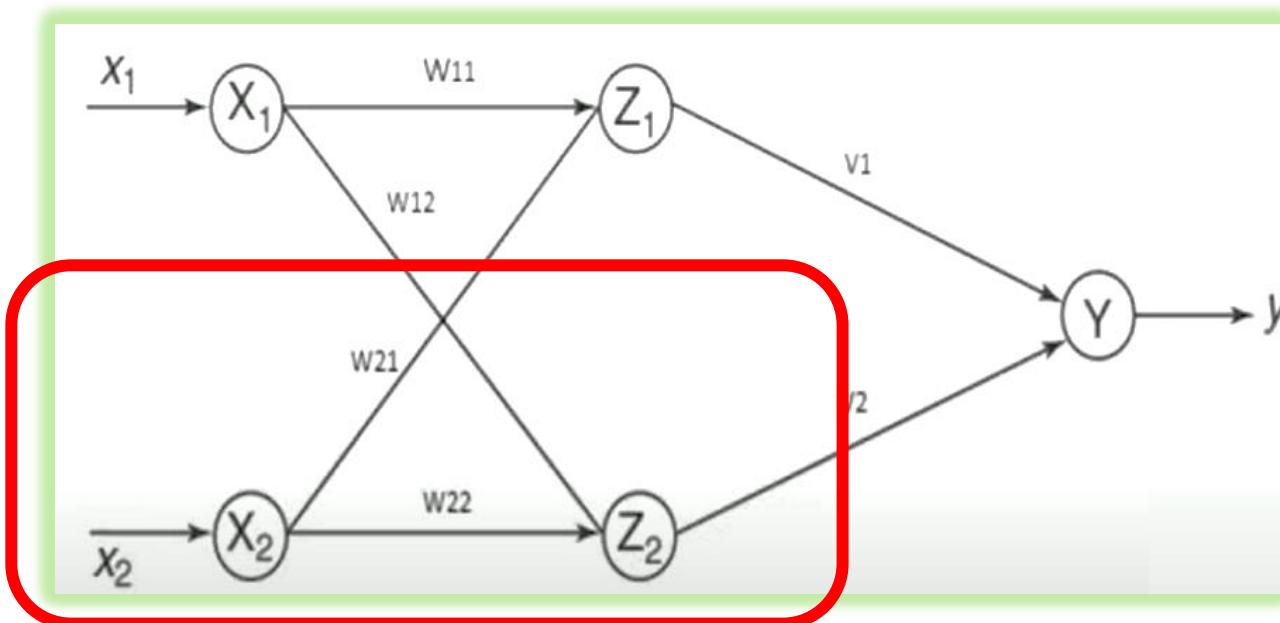
- $i_1(0,0) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 0 + (-0.5) \times 0 = 0$ (out =0 : z1t=0: no update)
- $i_2(0,1) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 0 + (-0.5) \times 1 = -0.5$ (out =0 : z1t=0: no update)
- $i_3(1,0) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 1 + (-0.5) \times 0 = 1$ (out =1 : z1t=1: no update)
- $i_4(1,1) : Z1 = w_{11} \cdot X_1 + w_{21} \cdot X_2 = 1 \times 1 + (-0.5) \times 1 = 0.5$ (out =0 : z1t=0: no update)



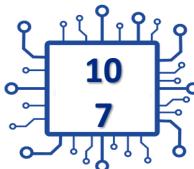
Solution : XOR gate (3/6)

- Second function $Z_2 = \bar{x}_1 \cdot x_2$

- initial weight $w_{12}=w_{22}=1$
- Threshold = 1
- learning rate = 1.5



x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0



Solution : XOR gate (4/6)

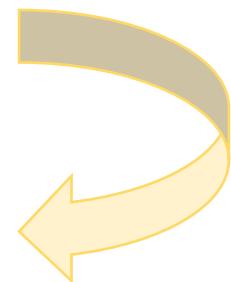
□ $i_1(0,0) : Z2 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = 1 \times 0 + 1 \times 0 = 0$ (out =0 : z2t=0: no update)

□ $i_2(0,1) : Z2 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = 1 \times 0 + 1 \times 1 = 1$ (out =1 : z2t=1: no update)

□ $i_3(0,1) : Z2 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = 1 \times 1 + 1 \times 0 = 1$ (out =1 : z2t=0: updated weight)

❖ $w_{12}\text{new} = w_{12} + \eta(t_3 - o_3)X_1 = 1 + 1.5 \times (0 - 1) \times 1 = -0.5$

❖ $w_{22}\text{new} = w_{22} + \eta(t_3 - o_3)X_2 = 1 + 1.5 \times (0 - 1) \times 0 = 1$



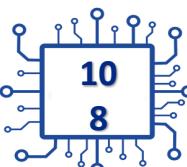
- initial weights :
 - $w_{12} = -0.5$,
 - $w_{22} = 1$,
- Threshold = 1
- learning rate = 1.5

□ $i_1(0,0) : Z1 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = (-0.5) \times 0 + 0 \times 0 = 0$ (out =0 : z2t=0: no update)

□ $i_2(0,1) : Z1 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = (-0.5) \times 0 + 1 \times 1 = 1$ (out =1 : z2t=1: no update)

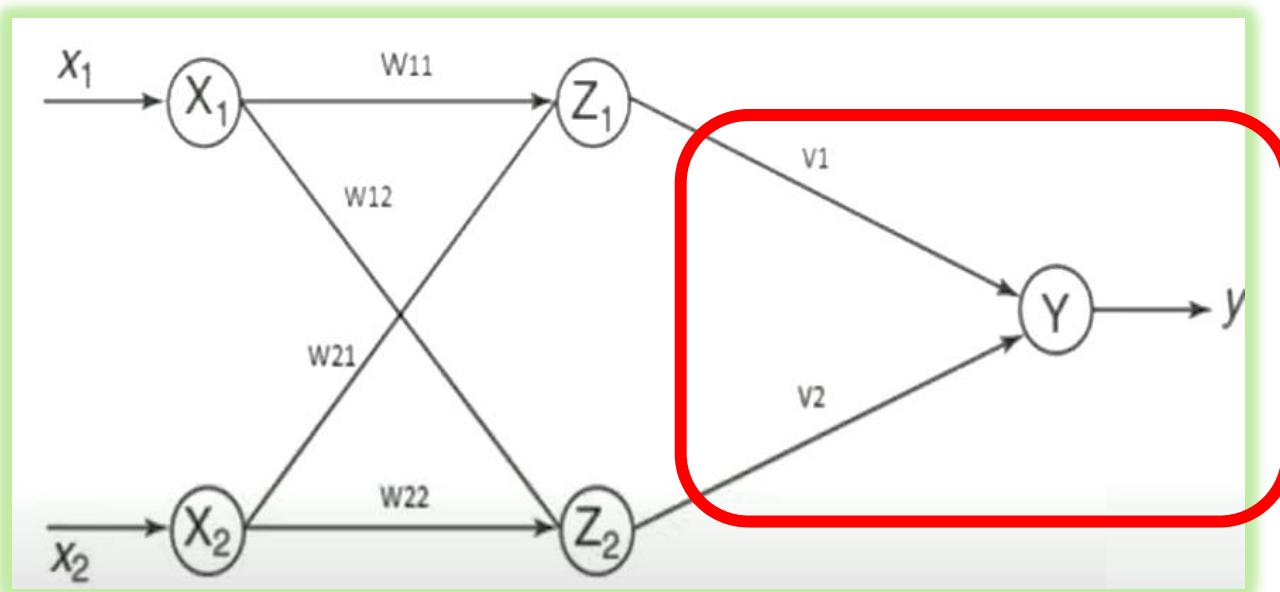
□ $i_3(1,0) : Z1 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = (-0.5) \times 1 + 1 \times 0 = -0.5$ (out =0 : z2t=0: no update)

□ $i_4(1,1) : Z1 = w_{12} \cdot X_1 + w_{22} \cdot X_2 = (-0.5) \times 1 + 1 \times 1 = 0.5$ (out =0 : z2t=0: no update)

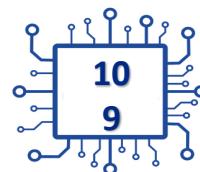


Solution : XOR gate (5/6)

- Third function $y = Z_1 + Z_2$
 - initial weight $v_1=v_2=1$
 - Threshold = 1
 - learning rate = 1.5

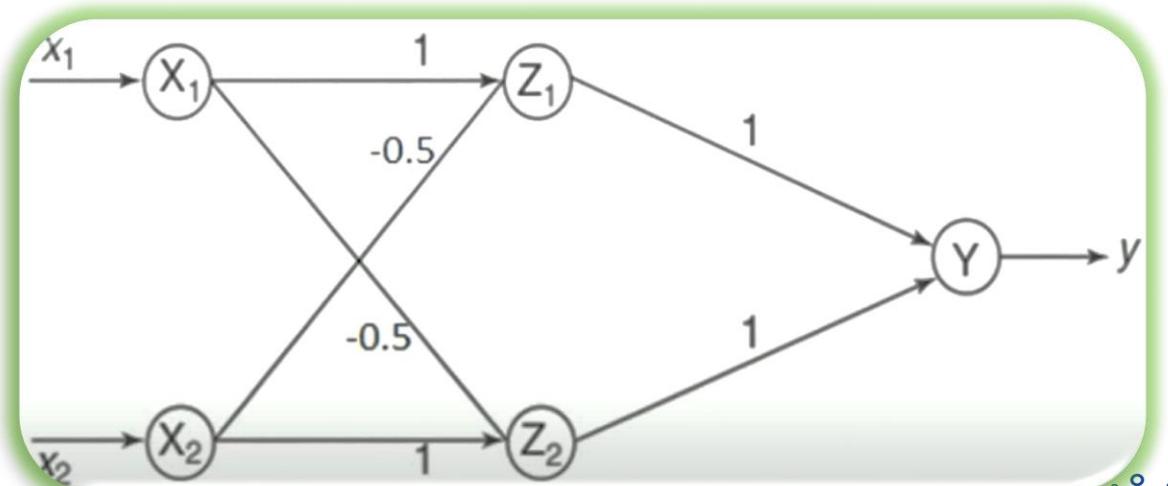
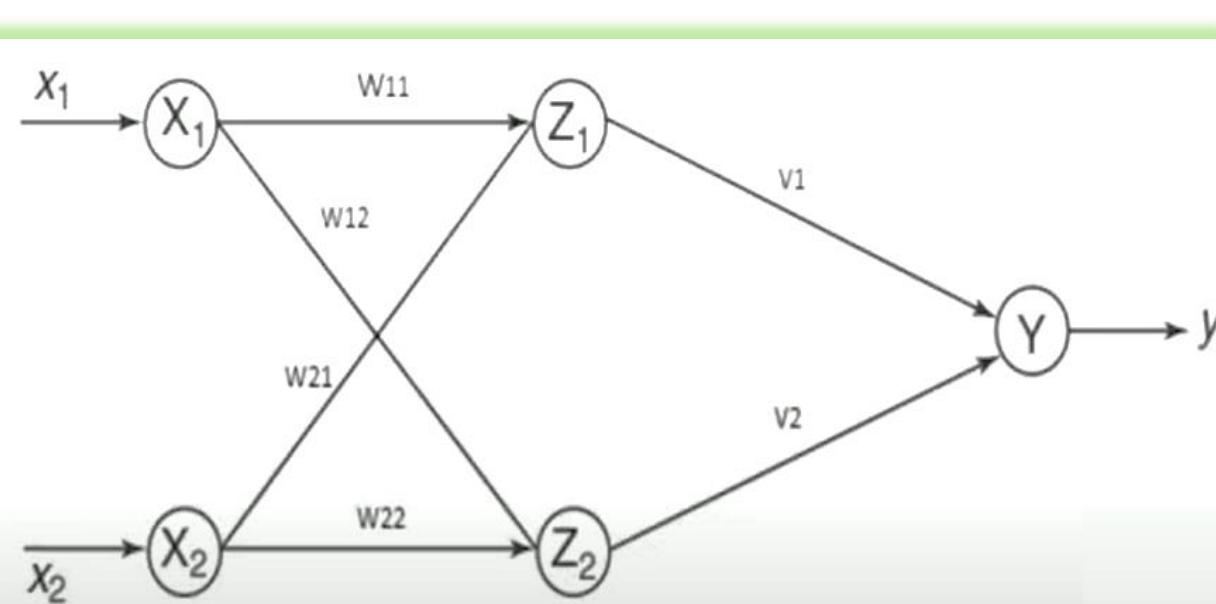


x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



Solution : XOR gate (6/6)

- $i_1(0,0) : y = V_1 \cdot X_1 + V_2 \cdot X_2 = 1 \times 0 + 1 \times 0 = 0$ (out =0 : yt=0: no update)
- $i_2(0,1) : y = V_1 \cdot X_1 + V_2 \cdot X_2 = 1 \times 0 + 1 \times 1 = 1$ (out =1 : yt=1: no update)
- $i_3(1,0) : y = V_1 \cdot X_1 + V_2 \cdot X_2 = 1 \times 1 + 1 \times 0 = 1$ (out =1 : yt=1: no update)
- $i_4(1,1) : y = V_1 \cdot X_1 + V_2 \cdot X_2 = 1 \times 0 + 1 \times 0 = 0$ (out =0 : yt=0: no update)

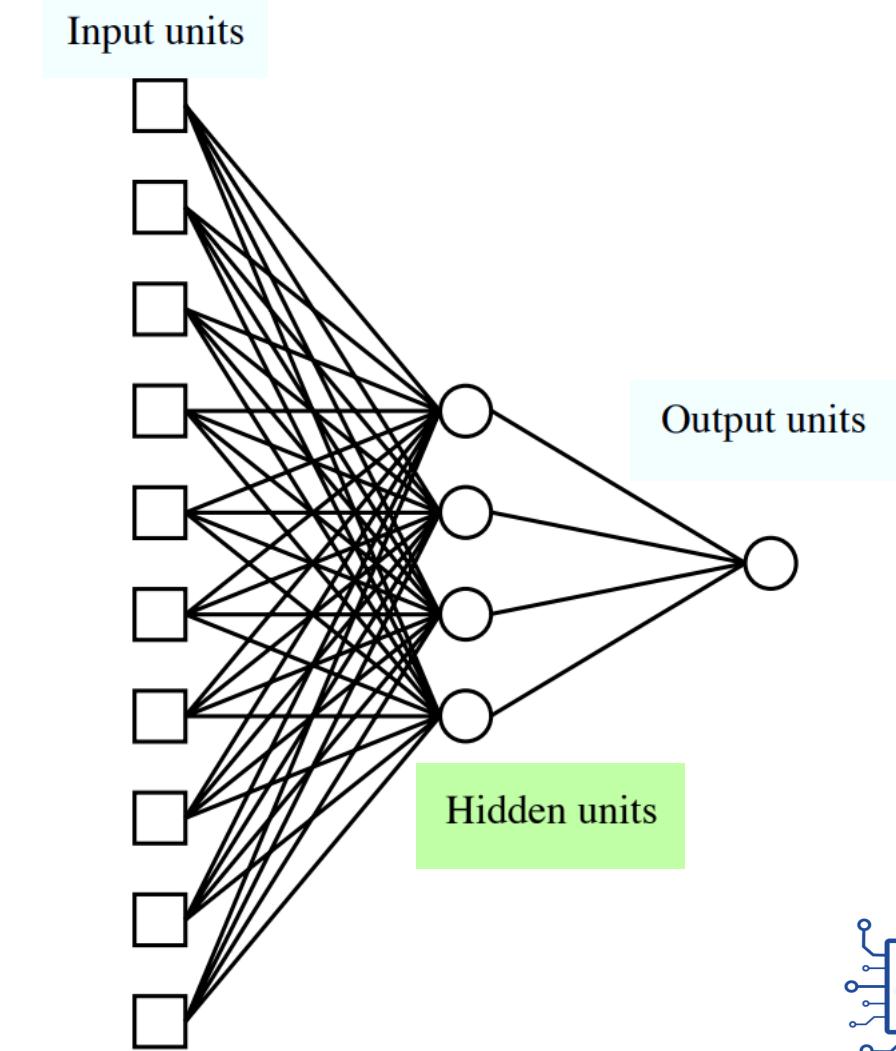


Perceptrons multicouches

- Les couches sont entièrement connectées,
- Nombre de couches caches généralement choisi par tests

Comment calculer et mettre jour les poids?

Back Propagation
algorithm



Merci pour votre attention

AI



Des questions ?



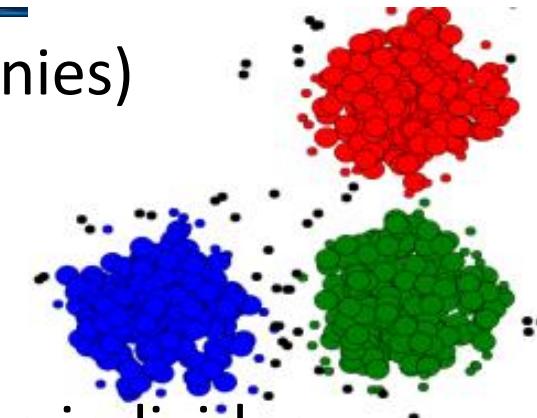
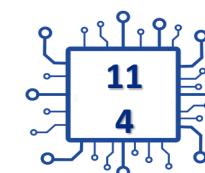
Chapitre 4:

Apprentissage non supervisé

Le regroupement (clustering)



Le regroupement

- Clustering = classification non supervisée (pas de labels prédefinies)
 - Clustering : vise à construire une collection d'individus :
 - Similaires au sein d'un même groupe
 - Non similaires dans des groupes différents
 - Il s'agit de construire une partition à k clusters d'une base D de n individus en se basant sur **la distance**
 - En clustering, la décision par la distance constitue une des approches les plus simples et les plus intuitives : **Il est naturel de considérer qu'un élément \in à une classe s'il est plus proche de cette classe que toutes les autres.**
 - Principales méthodes :
 - K-means : K centres mobiles
 - K-means flou
 - ..
- 
- 
- 

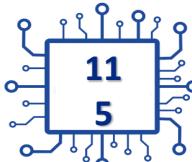
Rappel sur la notion de distance

- Si E est l'ensemble des vecteurs, soient $X = \{x_i\}$ et $Y = \{y_i\}$ avec $i=1,\dots,n$ deux éléments de E , on peut alors définir un certain nombre de distances classiques :

Distance de Hamming: $d_1(X, Y) = \sum_{i=1}^n |x_i - y_i|$

Distance euclidienne: $d_2(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$

Distance du maximum : $d_\infty(X, Y) = \max_{i=1,n} |x_i - y_i|$



K-means : algorithme

Algorithme K-Means

Entrée : k le nombre de groupes recherchés

DEBUT

Choisir aléatoirement les centres des groupes

REPETER

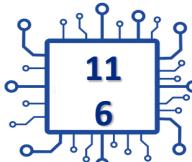
- i. Affecter chaque cas au groupe dont il est le plus proche au son centre
- ii. Recalculer le centre de chaque groupe

JUSQU'A (stabilisation des **centres**)

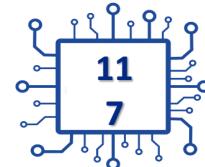
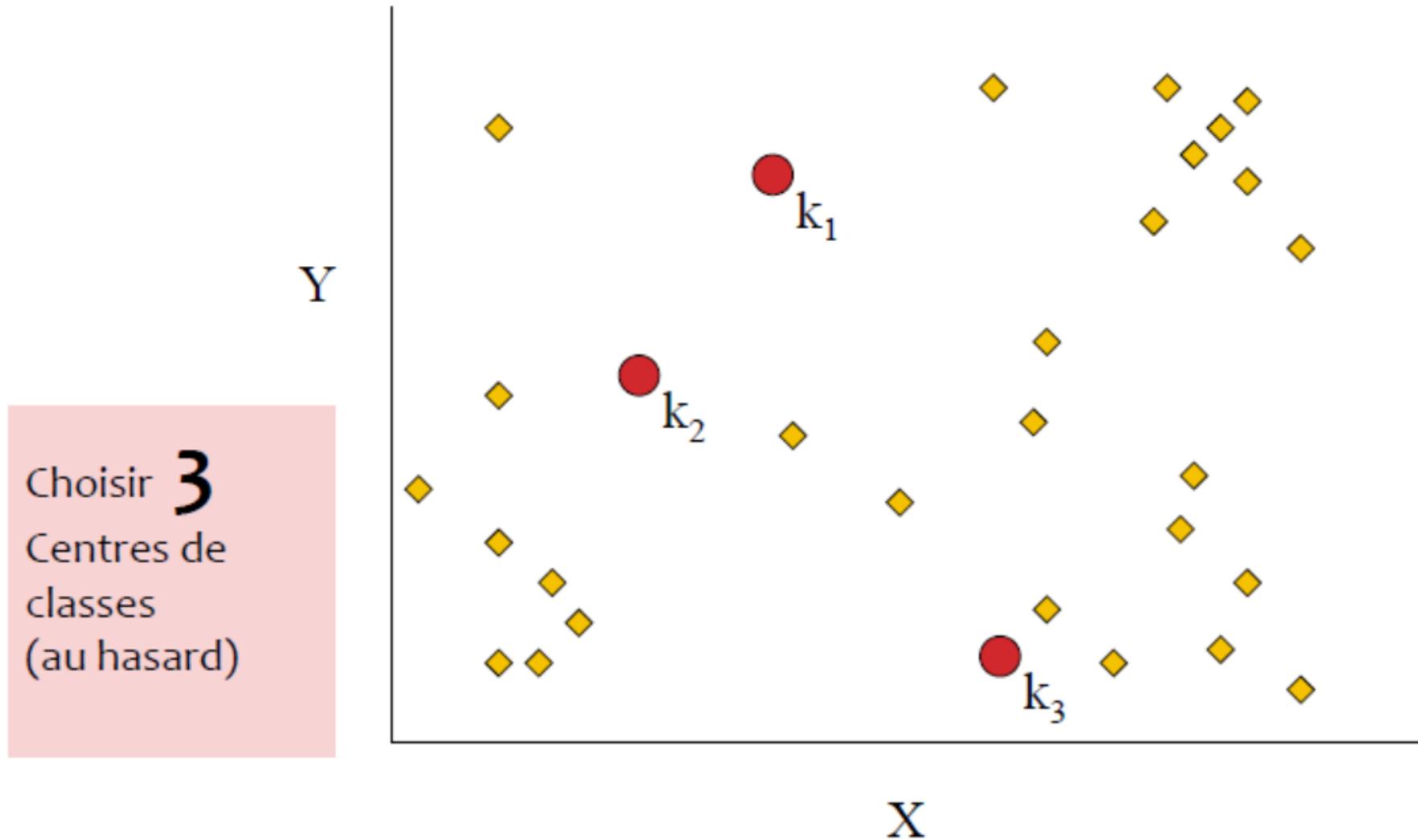
OU (nombre d'itérations = **t**)

OU (stabilisation de **l'inertie totale** de la population)

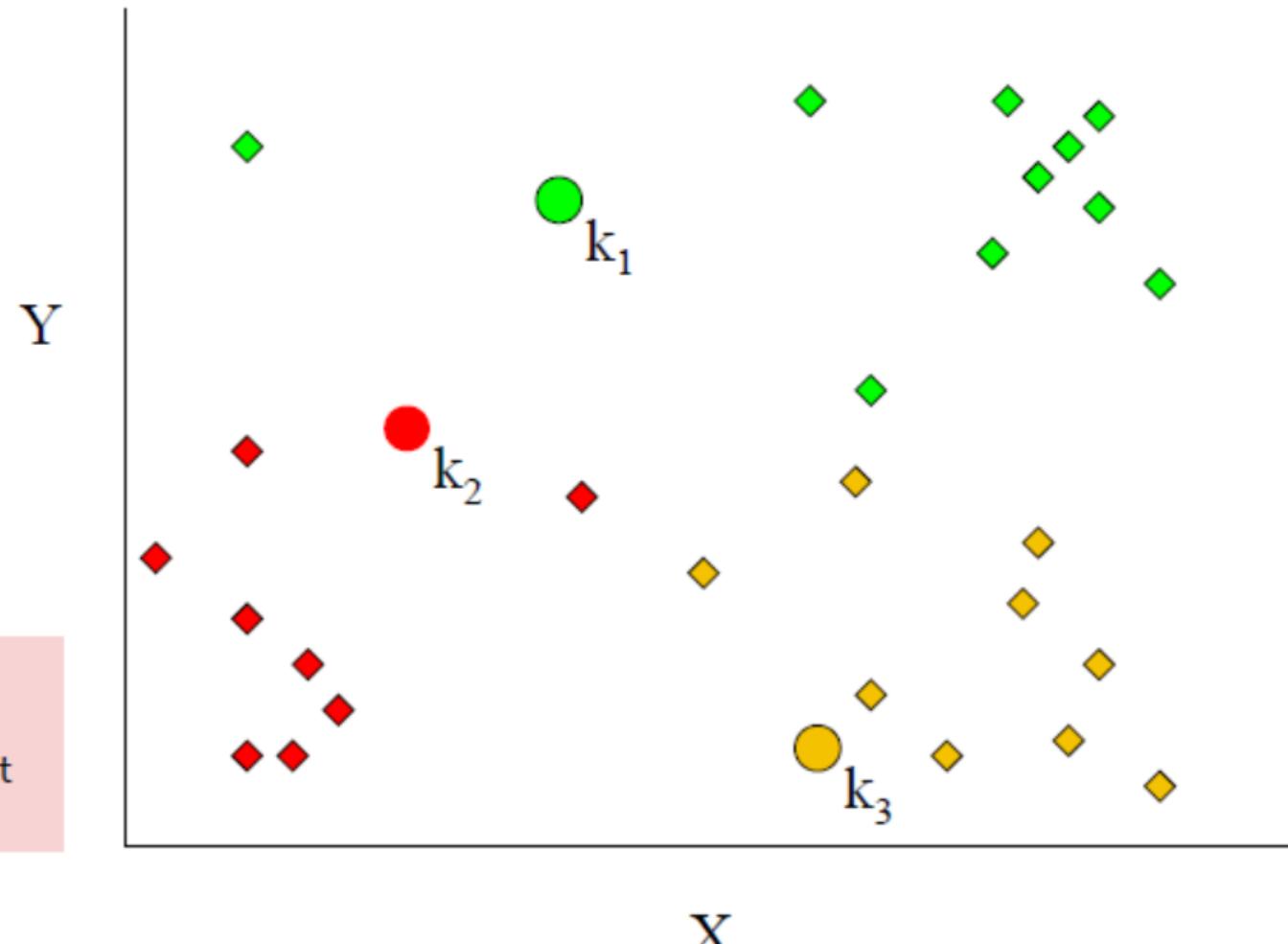
FIN



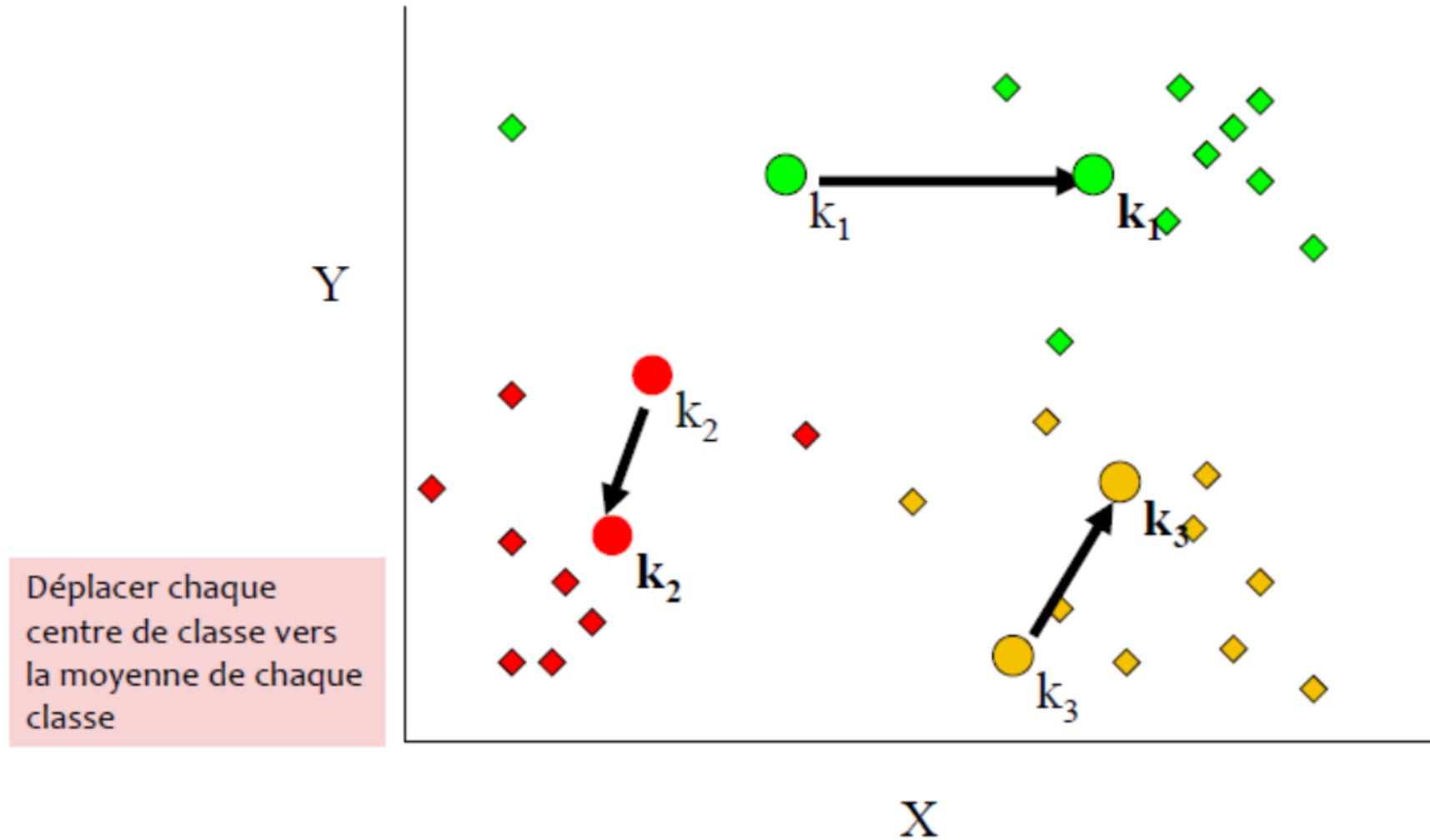
K-means : simulation (1/7)



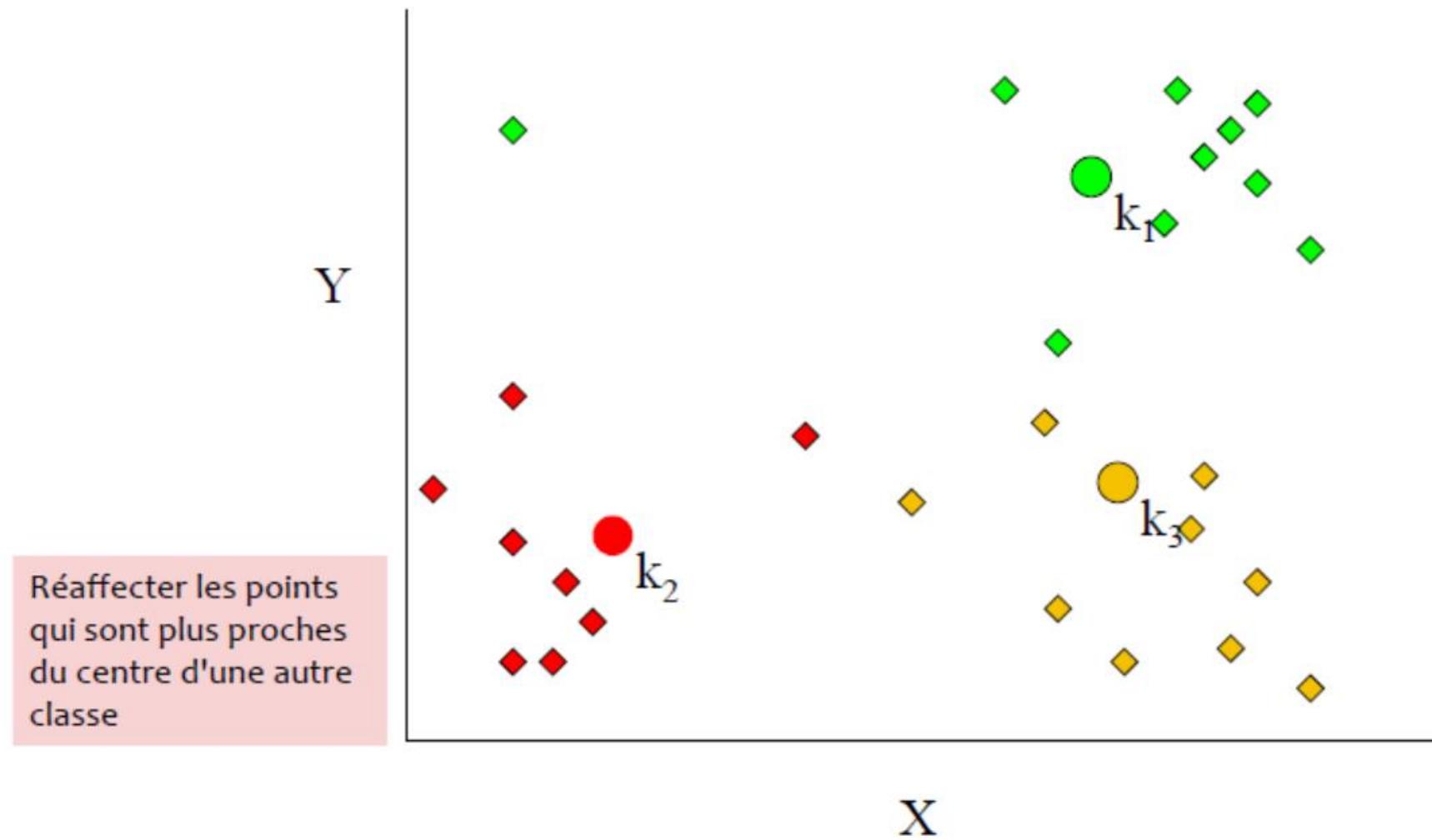
K-means : simulation (2/7)



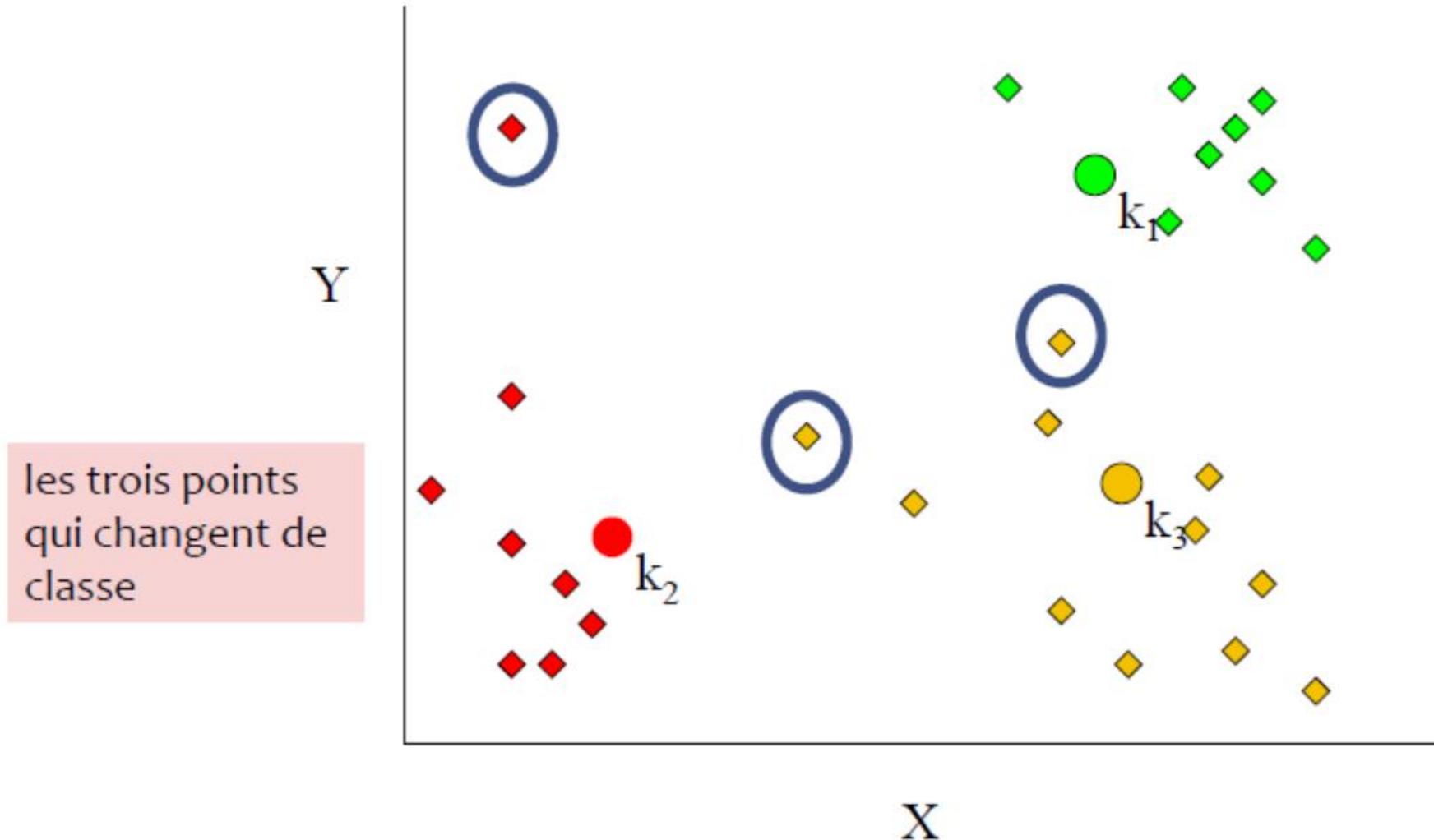
K-means : simulation (3/7)



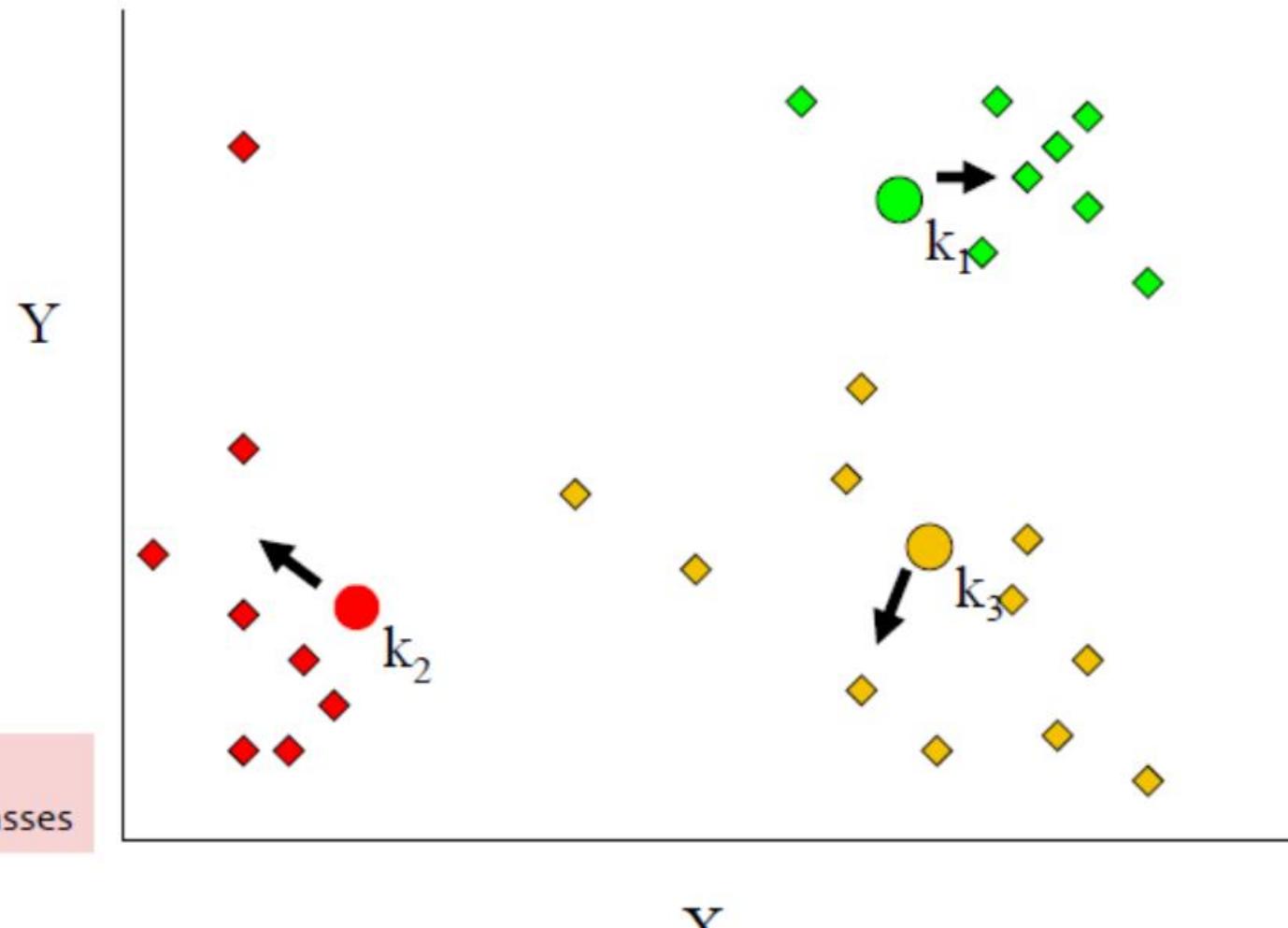
K-means : simulation (4/7)



K-means : simulation (5/7)

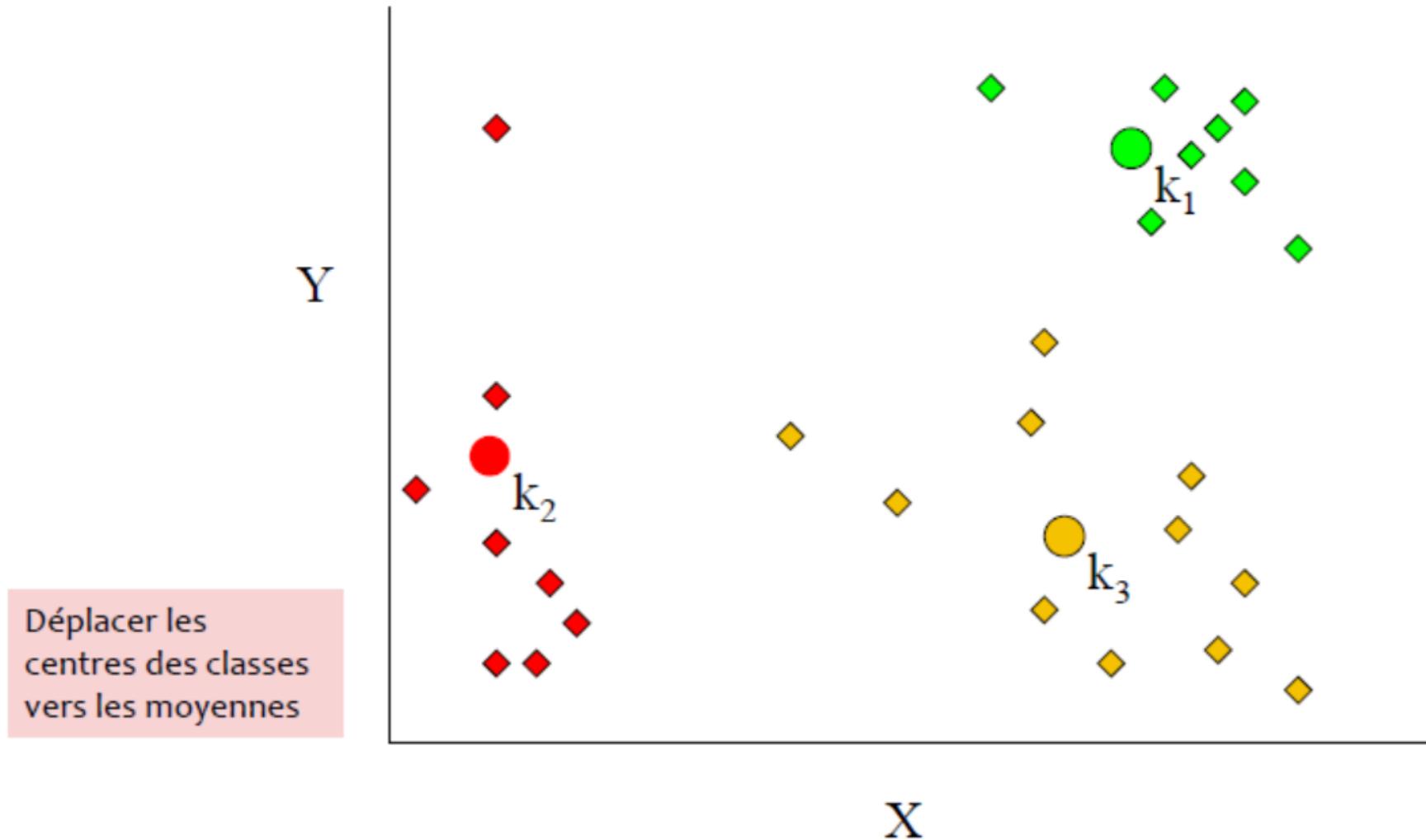


K-means : simulation (6/7)



Re-calculer les
moyennes des classes

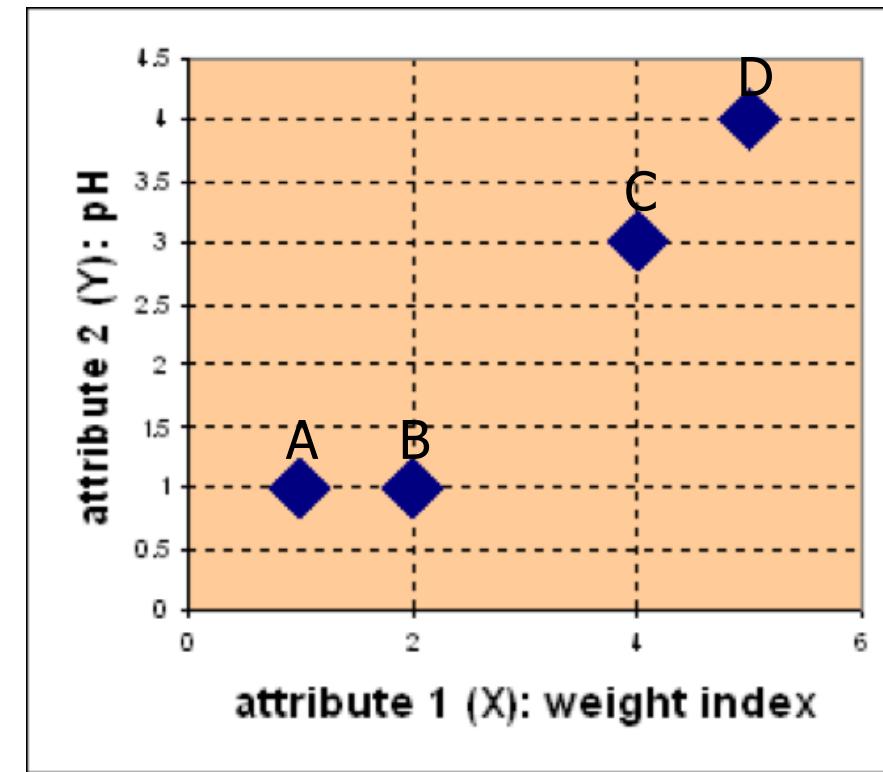
K-means : simulation (7/7)



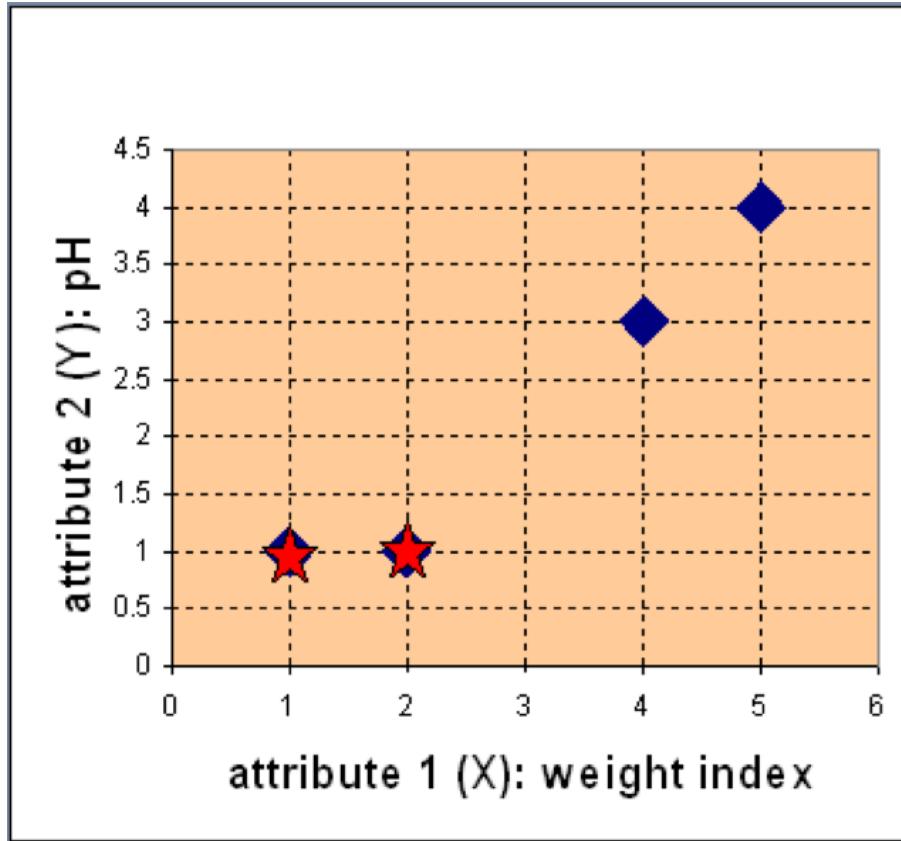
Exemple :

- Soient les 4 médicaments suivants, caractérisé chacun par deux caractéristiques weight index et ph
- On souhaite grouper ces médicaments en 2 groupements

Médicament	Weight index	pH
A	1	1
B	2	1
C	4	3
D	5	4



Exemple



$$c_1 = A, c_2 = B$$

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}$$

$A \quad B \quad C \quad D$

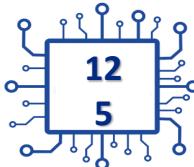
$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix}$$

$X \quad Y$

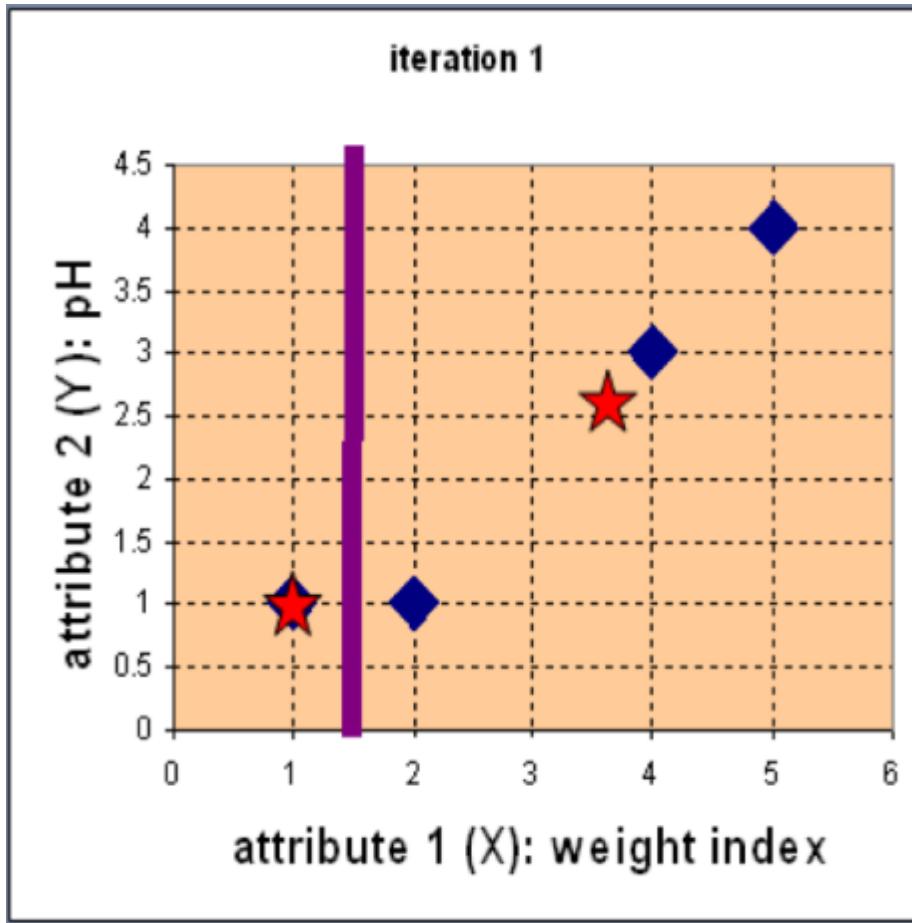
Euclidean distance

$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

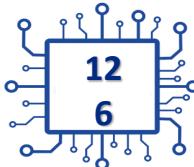


Exemple

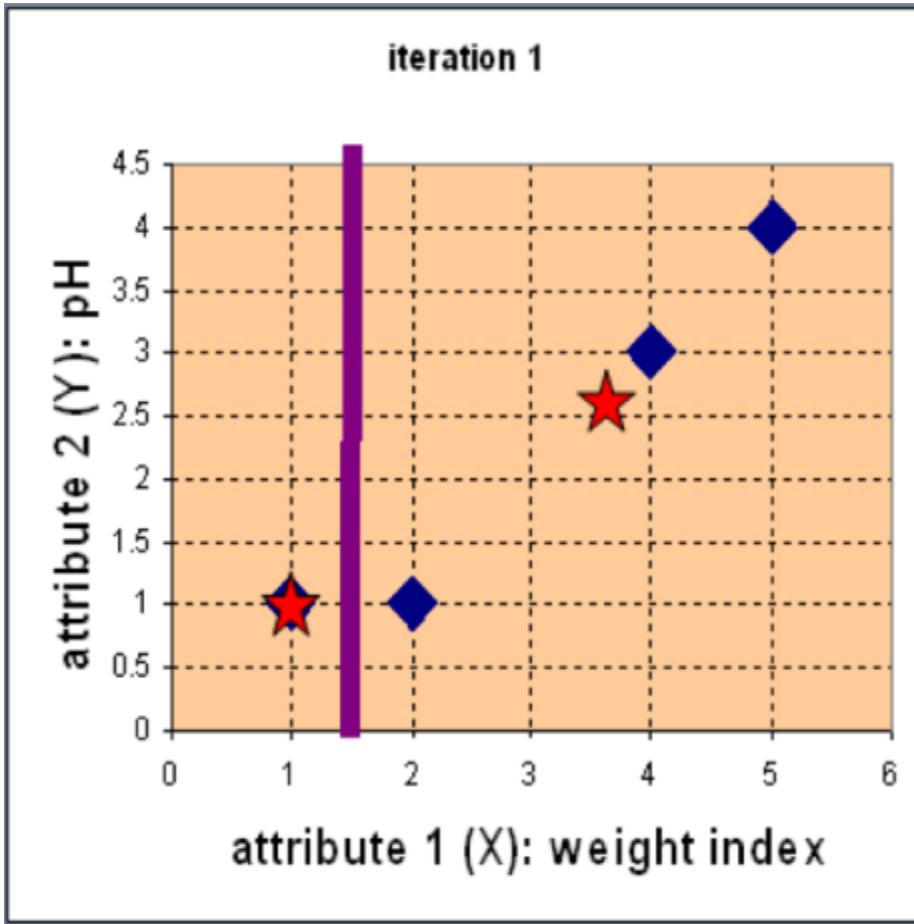


$$c_1 = (1, 1)$$

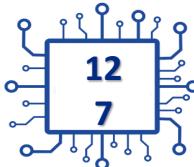
$$\begin{aligned} c_2 &= \left(\frac{2+4+5}{3}, \frac{1+3+4}{3} \right) \\ &= \left(\frac{11}{3}, \frac{8}{3} \right) \end{aligned}$$



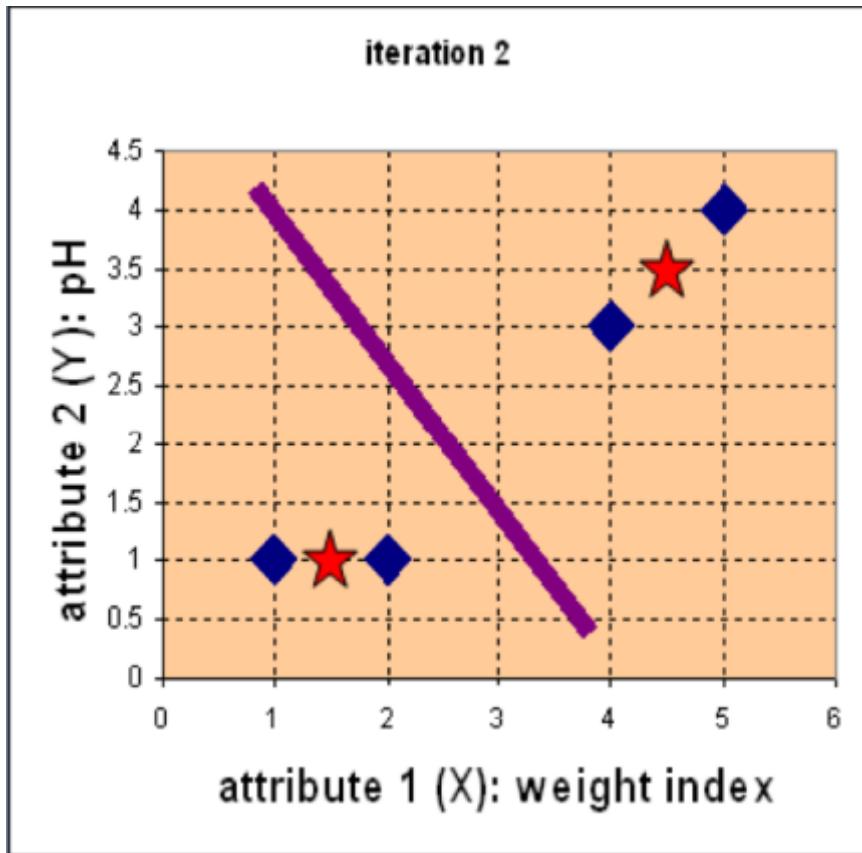
Exemple



$$\mathbf{D}^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \mathbf{c}_1 = (1,1) \quad \text{group -1}$$
$$\mathbf{c}_2 = \left(\frac{11}{3}, \frac{8}{3}\right) \quad \text{group -2}$$
$$A \quad B \quad C \quad D$$
$$\begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix} \quad X$$
$$\begin{bmatrix} 1 & 1 & 3 & 4 \end{bmatrix} \quad Y$$



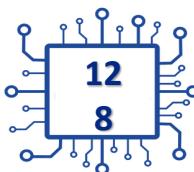
Exemple



$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2} \right) = \left(1\frac{1}{2}, 1 \right)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2} \right) = \left(4\frac{1}{2}, 3\frac{1}{2} \right)$$

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \mathbf{c}_1 = \left(1\frac{1}{2}, 1 \right) \text{ group -1}$$
$$\mathbf{c}_2 = \left(4\frac{1}{2}, 3\frac{1}{2} \right) \text{ group -2}$$
$$\begin{array}{cccc} A & B & C & D \end{array} \quad X$$
$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad Y$$

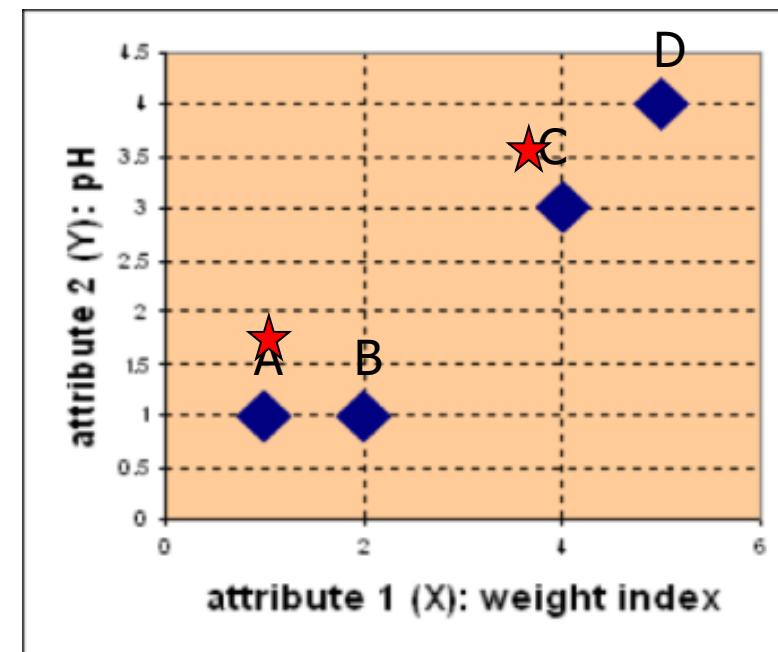


Exercice

- Refaire le clustering avec la distance de Manhattan
 - K=2
 - C₁ = A and C₂ = C

$$\text{1-distance} \quad \sum_{i=1}^n |x_i - y_i|$$

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



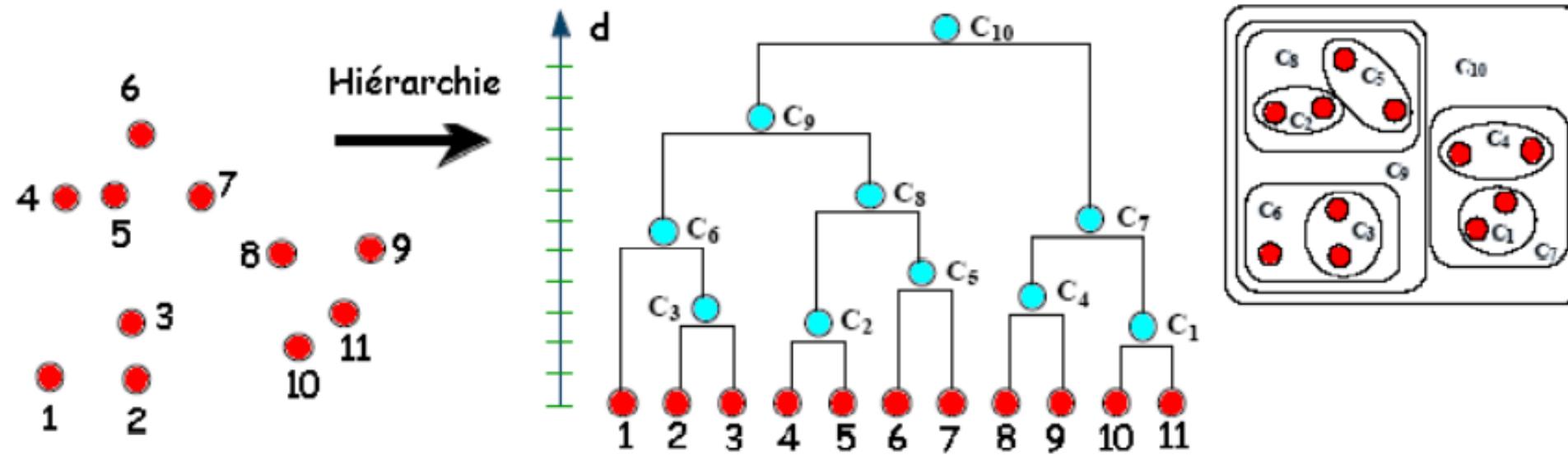
Points faibles du K-means



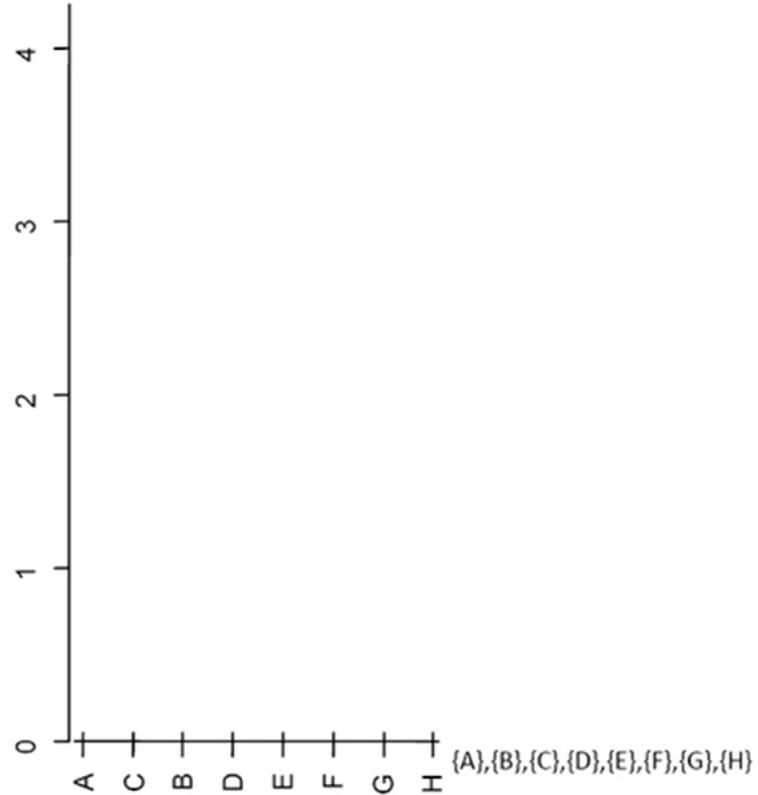
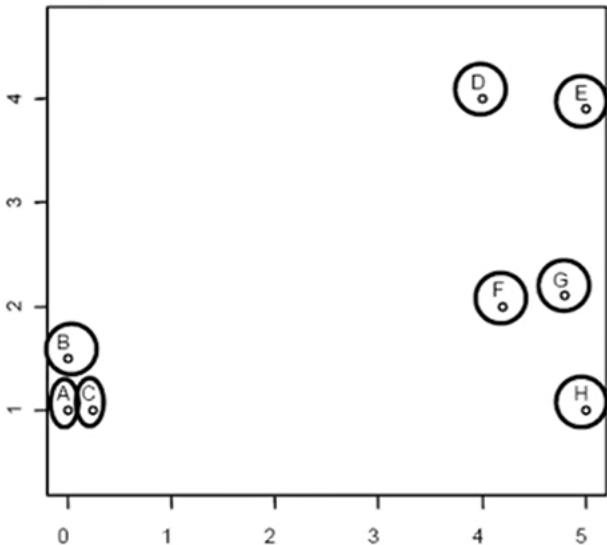
- Le choix du nombre de groupes est subjectif dans le cas où le nombre de classes est inconnu au sein de l'échantillon.
- L'algorithme du K-Means ne trouve pas nécessairement la configuration la plus optimale correspondant à la fonction objective minimale.
- Les résultats de l'algorithme du K-Means sont sensibles à l'initialisation aléatoires des centres.

Le groupement hiérarchique

- Connue aussi sous le nom Classification Ascendante Hiérarchique (ou CAH)
- Objectifs : production d'une structure (arbre) permettant :
 - La mise en évidence de liens hiérarchiques entre individus ou groupe d'individu
 - La détection d'un nombre de classe **naturel** au sein de la population



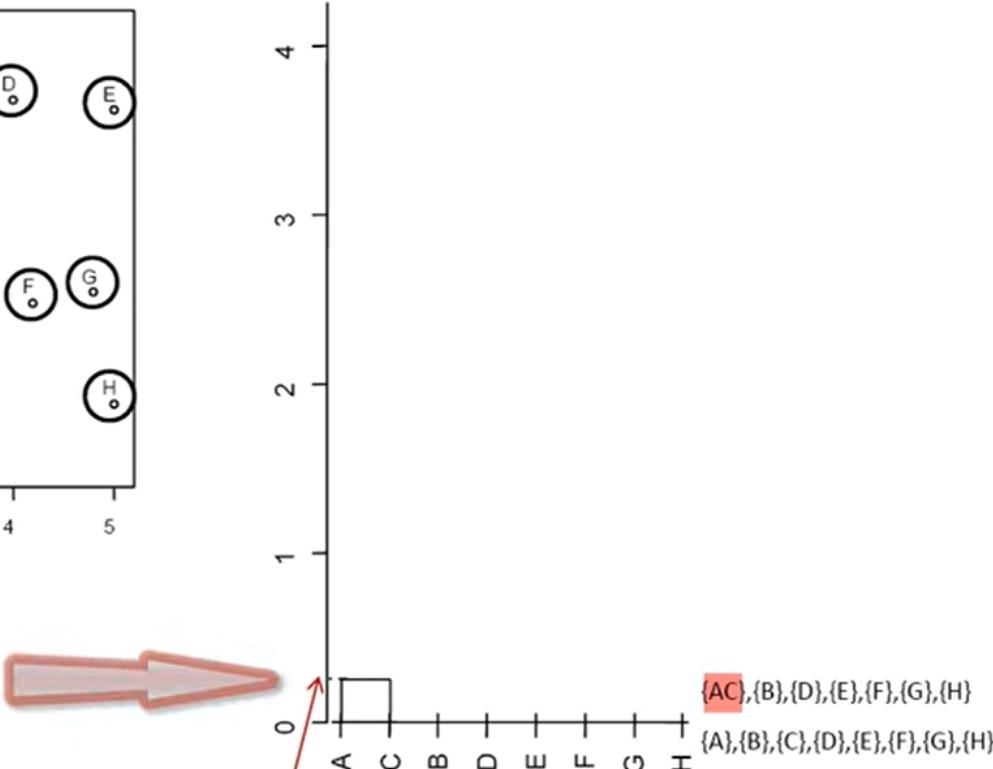
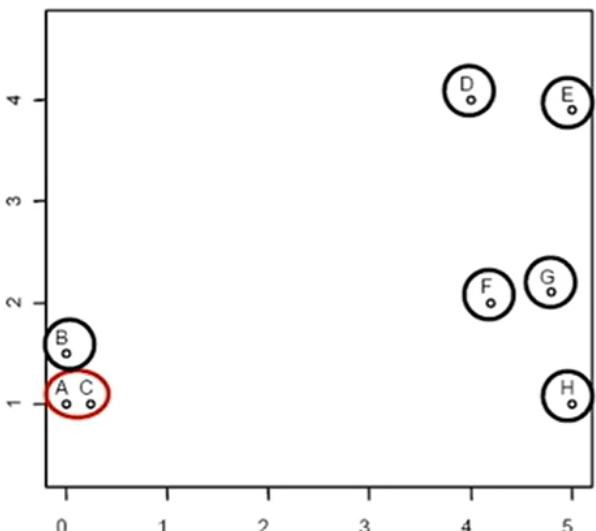
CAH : Exemple



(source : https://youtu.be/SE_4dLh5vXY)

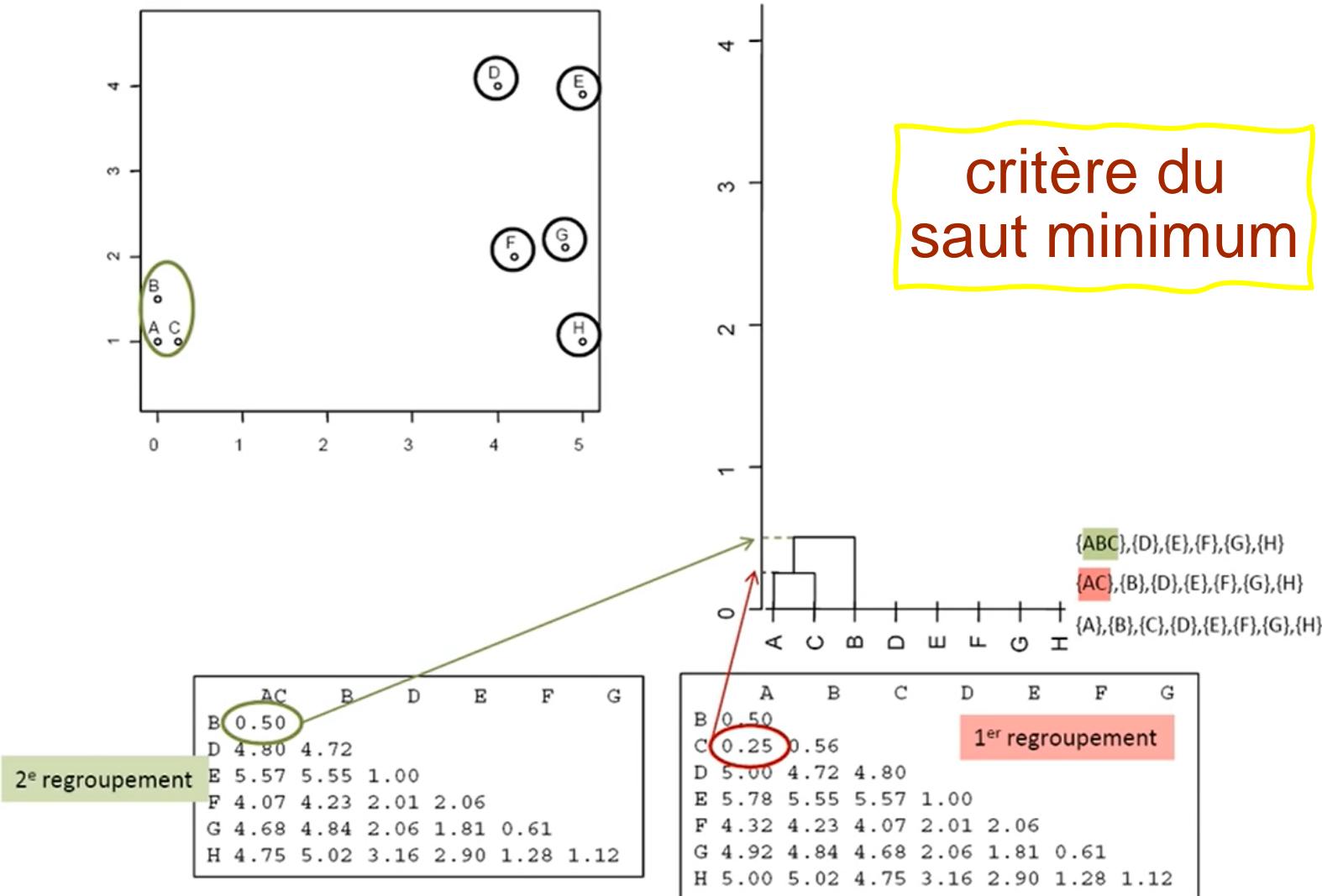
	A	B	C	D	E	F	G
B	0.50						
C	0.25	0.56					
D	5.00	4.72	4.80				
E	5.78	5.55	5.57	1.00			
F	4.32	4.23	4.07	2.01	2.06		
G	4.92	4.84	4.68	2.06	1.81	0.61	
H	5.00	5.02	4.75	3.16	2.90	1.28	1.12

CAH : Exemple

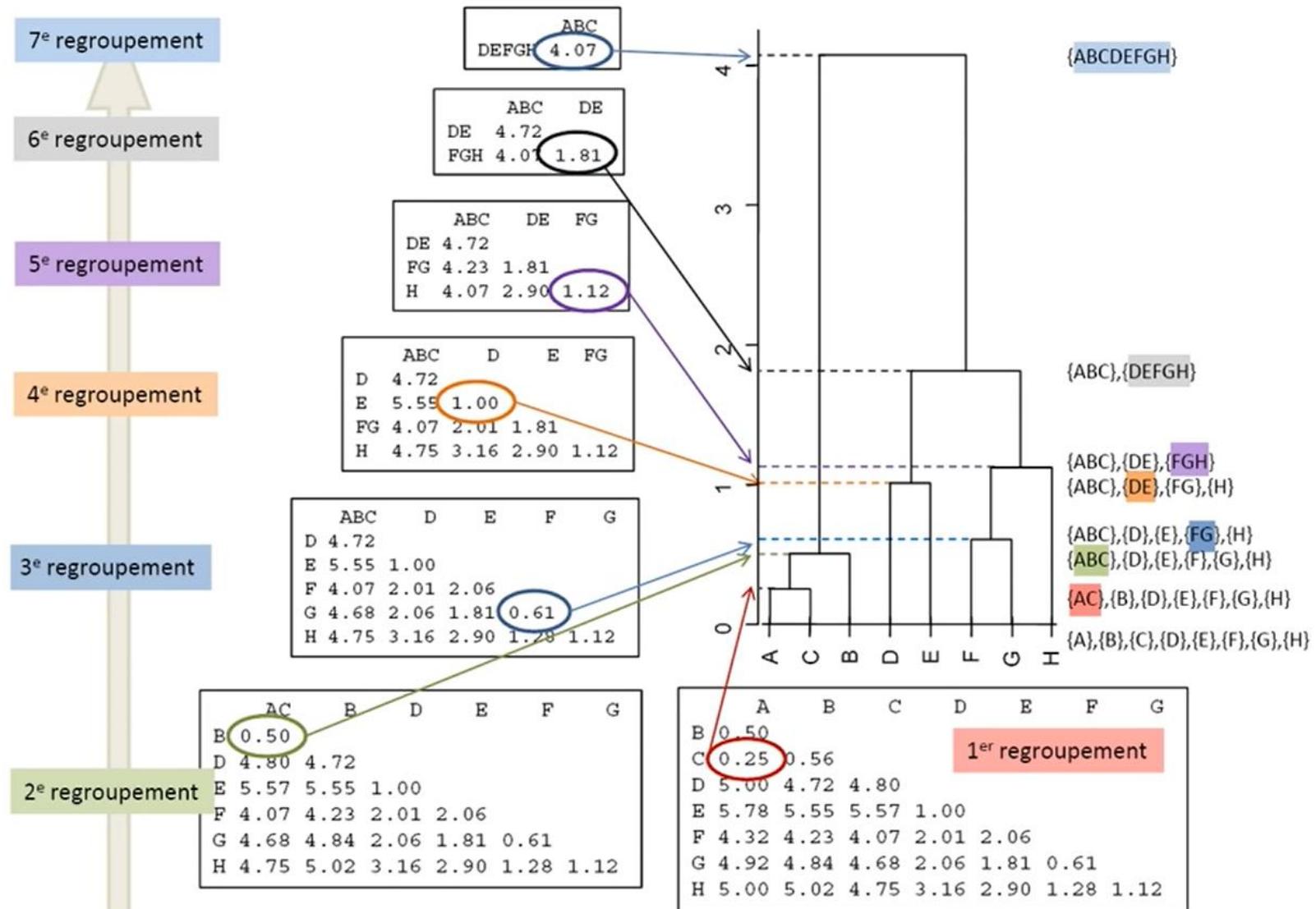


	A	B	C	D	E	F	G
B	0.50						
C	0.25	0.56					
D	5.00	4.72	4.80				
E	5.78	5.55	5.57	1.00			
F	4.32	4.23	4.07	2.01	2.06		
G	4.92	4.84	4.68	2.06	1.81	0.61	
H	5.00	5.02	4.75	3.16	2.90	1.28	1.12

CAH : Exemple

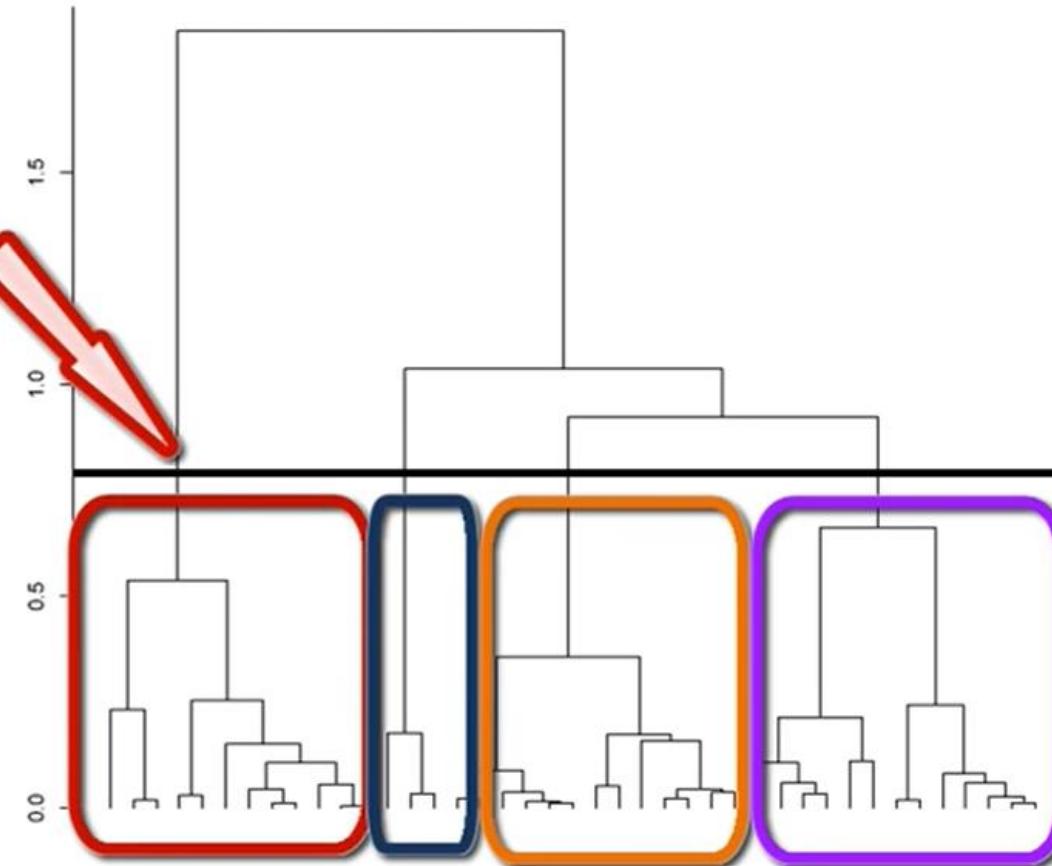


CAH : Exemple



CAH : Exemple

En définissant un niveau de coupure, on construit une partition



Points faibles de la CAH



- Le choix du critère de distance entre les groupes influe énormément sur les groupements formés.
- On ne peut pas décider sur la coupure optimale de l'arbre
- Devient complexe avec temps de calcul énorme avec de grandes données

Solution : on peut coupler la CAH au k-means :
C'est ce qu'on appelle la méthode mixte.



Merci pour votre attention

AI



Des questions ?

