

DATA MINING

séance 1,
25/10/2021

Introduction générale

Auparavant, la machine m'a jamais pris la décision à la place de l'homme. Depuis l'histoire de l'humanité (invention de l'électricité, la voiture, machine à vapeur...), c'est l'homme qui décideait.

Maintenant avec data mining, data science, big data l'ordinateur est en train de prendre le relais

exemple: * Auparavant, si quelqu'un demandait un crédit auprès d'une banque, c'était des humains qui se basaient sur son dossier, voyaient un certain nombre d'attributs du client, quel son salaire, sa situation familiale puis arrêtaient d'après ça est ce qu'il va lui donner le crédit ou pas, ça c'est ce qu'il était capable de rembourser le crédit ou pas.

* Maintenant, la machine fait ça et c'est elle qui décide. Et très souvent elle arrive à décider mieux que l'homme.

DATA: données

MINING: exploiter, creuser les données, aller jusqu'au fond pour chercher les pépites d'or à l'intérieur qui nous permettent de prendre les bonnes décisions

Opérationnelle (de production) (transactionnelle)

Informatique

Décisionnelle (stratégique)

exemple: * Faire une réservation d'avion ça prend des données qui sont stockées

Informatiques dans une certaine table d'une BD.

Opérationnelle

de production

* Lorsque vous retirez de l'argent près d'un guichet automatique, les transaction que vous avez faites ça crée une information qui sera enregistrée dans une table d'une BD.

⇒ l'informatique opérationnelle de production fait des programmes qui produisent des données qui permettent de les stocker

* l'informatique décisionnelle va utiliser ces données qui ont été produites par l'informatique de production, elle va les analyser, et va en tirer des informations pertinentes, c'est de la valeur (de l'argent) et prendre des décisions.

exemple: * Quand vous achetez qq chose sur Amazon. Tant de suite on vous dit que

~~Info décisionnelle~~ 90% des ceux qui ont acheté ça, ont aussi acheté ça. Donc on vous propose à acheter plus et donc ça fait gagner l'entreprise.

⇒ On dit que les ordinateurs avec le Big data nous commencent à nous qui on connaît bien. Ils découvrent des choses en nous même que nous même on ne connaît pas.

⇒ Informatique opérationnelle (de production):

Automatisation des tâches répétitives ⇒ c'est qu'il fait rapidement les choses qu'on fait

→ gestion de paie, → ramène les choses automatiques et rapides

→ gestion de stock,

→ comptabilité,

→ gestion de commandes, etc.

⇒ Toutes ces opérations là ça créent des données stockées qq part.

⇒ Informatique décisionnelle (stratégique):

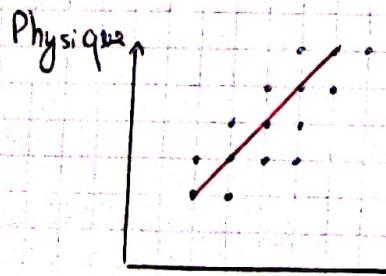
→ Extraction de connaissances à partir des BD,

→ visualisation de données multidimensionnelles,

→ modélisation,

→ prédition, etc.

exemples de la visualisation:



* on trace un graphique et on remarque des choses

(visualisation des données). On remarque

que en général l'étudiant a une tendance

positive c'est à dire lorsqu'on est bien en math en

Maths général on est bien en physique.

chaque point a comme coordonnées la note de math et de la physique d'un certain étudiant.

⇒ on visualise les données pour détecter quelque chose, un pattern (un format particulier) qu'on ne connaît pas par la suite. Donc on voit que la physique est liée aux maths positivement.

⇒ Puisque on remarque ça, on se pose la question suivante : Est ce que sur cet ensemble de points on peut passer une courbe ?

* dans l'exemple précédent c'est une droite d'équation $P = aM + b$ droite de regression, une fois on a le a et le b ça va me servir pour faire de la **prédiction**, par exemple si on connaît un élément n'existant pas dans l'ensemble des données, en connaissant sa matrice en maths on peut prévoir sa matrice de physique.

⇒ **modélisation** : c'est créer un modèle, comme on vient de faire $P = aM + b$ qui nous permet de faire de la **prédiction**. (est ce que un client va rembourser son crédit ou pas, est ce que ce type là va avoir une attaque cardiaque, est ce que cette femme va accoucher avec des complications ou pas, est ce que ce client va faire beau...)

⇒ **enrichissement de choses qu'on peut faire en utilisant le data mining** -

(est ce que ce client à partir de sa façon de se balader dans les sites web est un bon acheteur ou non (à travers son historique))

⇒ **DATA Warehouse** (entreposé de données) :

* ensemble de données historisées et orientées sujet.

exemple :
→ des clients lorsqu'ils vont acheter de Mayane,

ils ont maintenant une carte de fidélité, et ils ont

l'historique de tout ce qu'ils ont acheté,

donc ça c'est un ensemble historisé et

orienté sujet.

→ La Banque a un data warehouse de tous ses clients, c'est des données historisées sur tous les clients.

* Ensuite lorsqu'on stocke les données sur un data warehouse on va l'utiliser pour faire la suite du data mining.



DATA Warehouse : définition

* entrepôt de données.

⇒ ~~différents temps~~ Ensemble de données historisées avec leur variation dans le temps, organisées par sujet, stockées dans une BD unique, gérée dans un environnement de stockage partiel.

Le aidant à la prise de décision dans l'entreprise.

* Trois fonctions essentielles :

- collecte de données de base existantes et chargement.
- gestion des données dans l'entrepôt.
- analyse de données pour la prise de décision.

(car les données peuvent être dispersées pour chaque vendeur BD) (ensuite on va combiner toutes ces données dans un data warehouse)

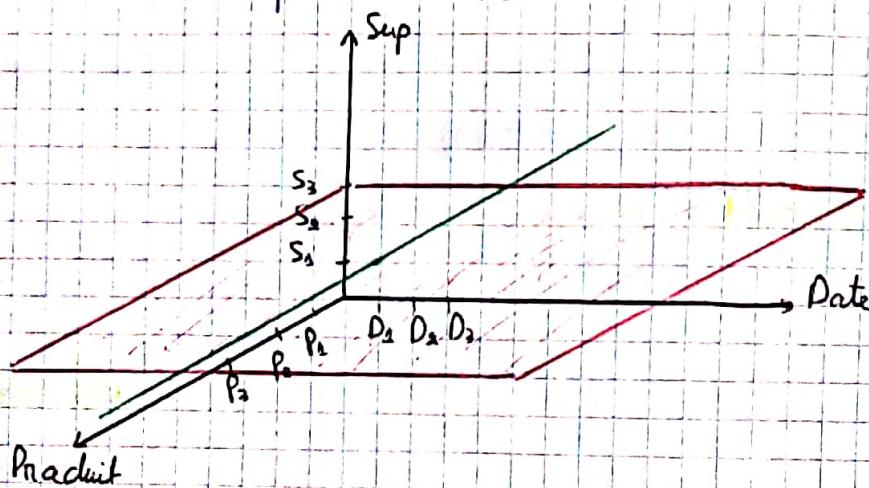
Exemple :

→ Visualisation.

on a des supermarchés, des produits et des dates.

⇒ un produit a été vendu à une certaine date dans un supermarché.

on représente ceci dans un espace tridimensionnel.



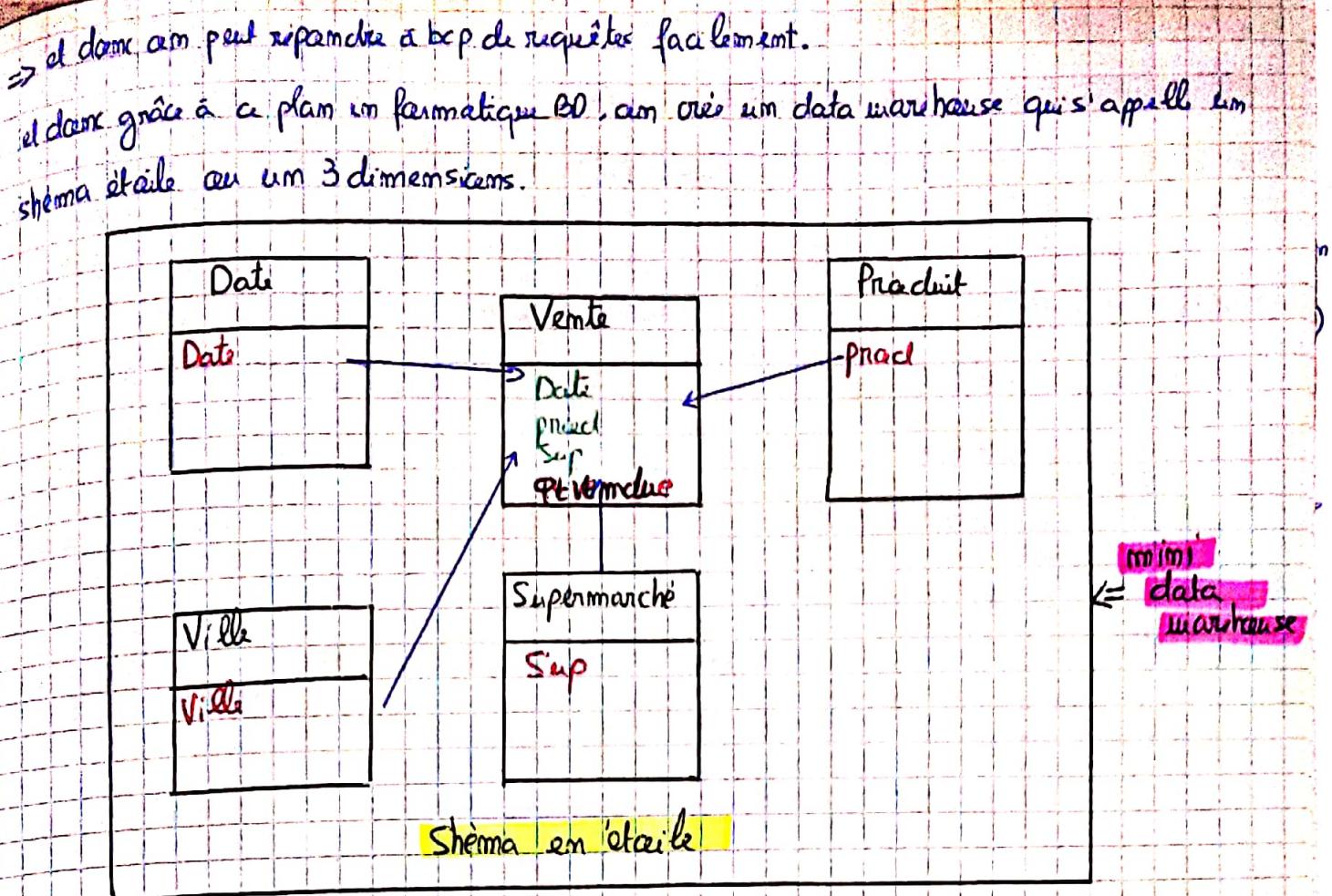
→ si on fixe une date donnée, un produit donné et un supermarché donné ⇒ on obtient à une date donnée pour un produit donné dans un supermarché donné la quantité vendue (on donne un point dans l'espace à 3 dim).

→ chaque fois qu'on fixe une valeur de date, une valeur de produit et de supermarché on a 1 point dans l'espace.

⇒ Utilité : créer un data warehouse qui répond aux questions suivantes:

* total des ventes d'un supermarché pour tous les produits à une date déterminée.

* le chiffre d'affaires d'un supermarché pour tous les produits sur toute l'année.



* 3 dimensions \Rightarrow la dimension date, produit, supermarché (des entités)

en rouge : clé primaire

* date, produit, supermarché \Rightarrow tables de dimension

* vente \Rightarrow tabl. de fait

en vert : clé étrangère

qt vendue : dans une date précise, pour un produit précis dans un supermarché déterminé. Voici la quantité qu'on a vendue.

D On peut ajouter une 4^{me} dimension Ville

\hookrightarrow dans un certain supermarché d'une certaine Ville dans cette date pour ce produit voici la quantité vendue.

NB : on peut ajouter autant dimensions qu'on veut mais on a 1 seul tableau de fait.

\Rightarrow On vient de construire un mini data warehouse qui permet de faire des requêtes très faciles très simples

par exemple : si on veut le total des produits dans une date donnée pour une ville

donnée : on fixe la date et la ville et on obtient le total, on fait le Σ

on fait le total du Σ sur les 2 autres en les variant

Schema récapitulatif (ver) Architecture type

Sources de données \Rightarrow extraire les données \Rightarrow Data warehouse

dont j'ai besoin et je vais les nettoyer (corriger les erreurs et supprimer les données aberrantes, données absentes qu'on va imputer) et on va les transformer

Extract
Nettoyer
Transformer
Charger
Rafraîchir

(par ex on a un à partir de 2 variables une seule variable par ex on a le revenu de chaque famille et le membre de membres de chaque famille \Rightarrow on a une 3ème colonne correspondant au revenu par tête en divisant le revenu total par le nombre de personnes dans la famille)

et ensuite charges les données
Il y a un logiciel qui fait ça ETL

Extract Transform Load

\Rightarrow on extrait les données, on les transforme et on les charge dans la BD.

les logiciels ETL

\rightarrow TALEND

1^{er} tier : créer des serveurs data warehouse

2^{eme} tier : créer des matières des OLAP (en ligne analytic)

3^{eme} tier : faire des GUI pour répondre aux questions

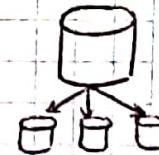
\Rightarrow Architecture générale de l'information décisionnelle.

contient un certain nombre de sujet par ex la comptabilité, gestion du personnel, gestion de stock

\hookrightarrow l'entrée peut être des données contenant des magasins de données

= Data marts

" petit data warehouse qui on peut brasser dans un super data warehouse



\Rightarrow on crée à partir des data warehouse des cubes de données

OLAP OLAP

cube de données par exemple l'espace 3D qui on a vu (produit, date, supermarché)

\downarrow servir

pour faire des requêtes, des graphiques et pour créer des modèles, c'est pour faire du data mining

1. OLTP et OLAP page 7

- * On a des applications par exemple une application de réservation aérienne dans une compagnie aérienne, à partir de ces réservations on va créer des BD de production (BD transactionnelle) **OLTP** = **on ligne transactionnel processing**, donc on est en train de mettre les données produites par une application dans une BD on dit qu'on est en train de faire des transactions et en fait des OLTP (des traitements transactionnels).

- * Sur les BD transactionnelles on utilise un ETL (extract transform load) pour créer un DW (data warehouse) et à partir du DW on va créer des cubes de données OLAP qui permettent de créer des rapports et de faire des analyses et on peut faire du DM (data mining) pour créer des modèles qui vont nous aider à prendre des décisions.
- * **OLAP** = **on ligne analytical processing** (traitement analytique)

→ stocker des données dans la BD => traitement transactionnel.

→ exploiter le data warehouse => traitement analytique

de data mining : page 8

→ Un confluent de la statistique et de l'informatique décisionnelle.

→ DM l'une des 10 technologies émergentes qui changent le monde au 21^e siècle.

Définition : de data mining : p9

* C'est l'application des technologies d'analyse des données et d'intelligence artificielle à l'exploration et à l'analyse de grandes BD en vue d'en extraire des informations pertinentes pour l'entreprise et de les utiliser en particulier dans les systèmes d'aide à la décision.

Data mining : Autres appellations et définitions : p.10

→ KDDB : découverte des connaissances dans les BD.

→ d'extraction d'informations, au travers incertaines, potentiellement utiles à partir de données.

→ la découverte de nouvelles corrélations, tendances et modèles sur le tamisage d'un large volume de données.

→ Terturer l'information disponible jusqu'à ce qu'elle avance.

Plusieurs sources de données : p11

Types : → structuré (sous forme d'une table d'une BD, les lignes sont des enregistrements et les colonnes sont des attributs.)
→ non structuré (texte, image, vidéo, audio...).

Sources : → structuré : domaine Telecom, production, transport, Retail (ventes de détails), Finance => Mostly structured.
→ non structuré : Media, Web, emails => Mostly unstructured.

NB : On peut pas utiliser (faute du DM) les données non structurées avant de les structurer.

↳ données non structurées → données structurées → faire du DM
utiliser les algos du DM

exemple :

- une image c'est une matrice de pixels, quand on rentre une image à l'ordinateur on ne rentre pas l'image comme on la connaît mais on rentre des pixels, même pour les emails on va voir comment les transformer en données structurées.

Data mining et Big Data : p12

→ Des big data « données massives », désignent des ensembles de données qui deviennent tellement volumineuses qu'ils deviennent difficiles à travailler avec des outils classiques de gestion de BD ou de gestion de l'information.

↳ Idée maîtresse : distribuer les données et paralléliser les traitements sur plusieurs processeurs (qui travaillent en parallèle).

Termes :

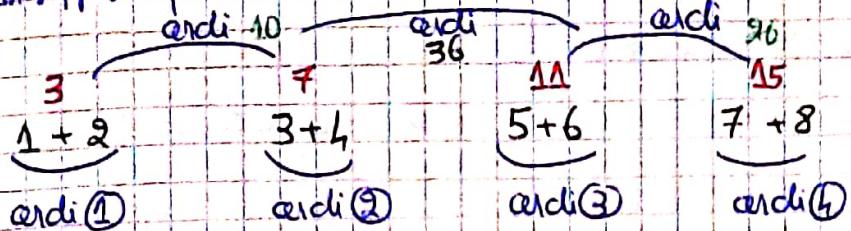
- * Data mining
- * Data warehouse
- * Data Mart (magasin de données)
- * ETL → extraire transformer charger les données
- * OLTP → produire les données
- * OLAP → analyser les données

Exemple : sur distribuer les données et paralléliser les traitements.

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8$$

→ un ordinateur qui l'exécute séquentiellement fera le calcul tout seul.

→ on suppose qu'on a 4 ordinateurs



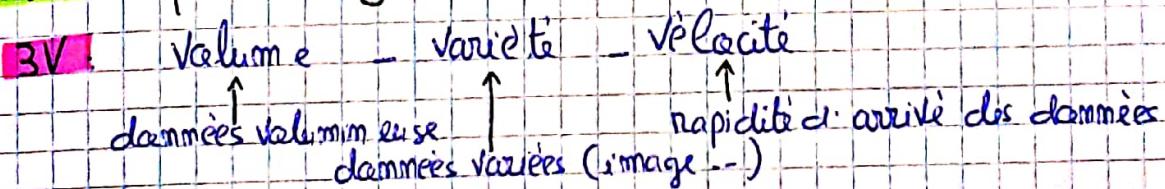
→ l'algorithme devient algorithmique logarithmique au lieu d'être linéaire car on l'a parallélisé.

* les données sont très grandes donc on les parallélise pour aller plus vite.

* et même si on n'a assez de place en mémoire pour stocker les données on aura un prob de temps. C'est pour cela qu'il faut paralléliser les données. ⇒ Big Data

* on peut faire du data mining sur des big data !

Caractéristiques du Big data : p13



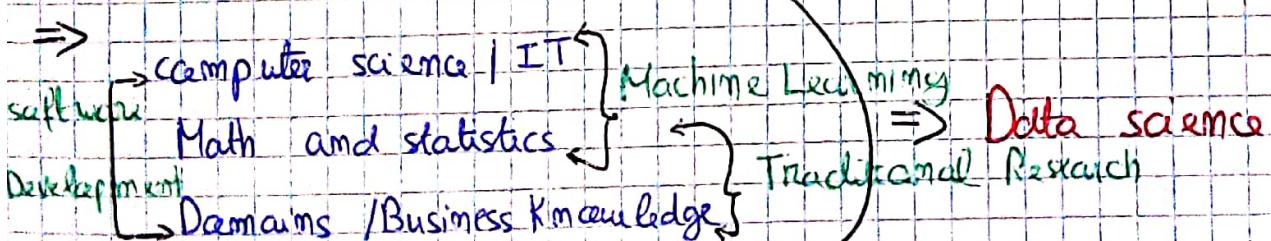
Nouvelle formation : Data science, Profil : p14 - 15

⇒ data science - A la croisée de 3 profils :

* statisticien data mimer

* informaticien

* connaissances métier



Rôles et compétences détaillés : p16

- data scientist
- data engineer
- software engineer

data engineer prépare les données au data scientist.

• data scientist, la parole rare, p 17

⇒ le data scientist est quelqu'un qui sait mieux développer qu'un statisticien et qui connaît mieux les statistiques qu'un développeur.

Facteurs d'émergence du DM : p 18

Pourquoi le DM est fait ?

- production massive des données.
- grande capacité de stockage. (mémoire)
- processeurs plus puissants.
- contexte très concurrentiel.
- disponibilité de logiciels de DM.

Combien d'information produis
chaque jour?

2.5×10^{18}	carradière/jour
10^{18}	quintilliam
10^{15}	quadribiliam
10^{12}	triliam
10^9	Billicam

→ la quantité d'information augmente chaque jour.

Positionnement du DW et du DM p 19:

- * on a un DW qui permet de fournir les données, on utilise ces données pour faire du DM, on obtient des résultats qui on utilise pour prédir et agir et ensuite observer et comparer.

Systèmes décisionnels : suite p 20 à 21

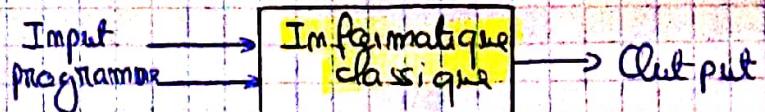
- * on a des BD transactionnelles ; on fait du data mining ça démontre les données et on va les utiliser ensuite pour faire des requêtes et sortir des tableaux et rapports, on peut aussi faire des cubes de données pour sortir des rapports on peut faire du DM pour sortir des modèles.

DATA Mining p 22

deux familles de méthodes :

→ méthodes d'apprentissage supervisé (classement / prévision), caractérisés par l'existence d'une variable privilégiée à prédire

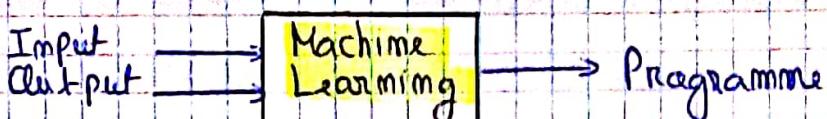
→ méthodes d'apprentissage non supervisé (ou descriptives) où il n'existe pas de variable privilégiée à prédire



Input : par ex données d'un employé (heures de travail, nombre d'enfants...)

- * de data mining utilisent souvent des algos de machine learning.

Supervisé



- * Le machine learning est un sous domaine de l'intelligence artificielle.

exemple :

- * client qui veut un crédit
- * la Banque possède une BD contenant les données de tous ses clients + une variable Y qui concerne le remboursement des crédits des clients (oui, non) on donne à l'ordinateur les données des clients (X) et la réponse (Y) → est ce qu'il a remboursé le crédit ou non et ensuite l'ordinateur va apprendre le programme à utiliser sur de nouveaux clients pour prédire est ce qu'ils vont rembourser leurs crédits ou pas.

BD des clients

$Y \rightarrow$ Remboursement de crédit (oui, non)

—	—	—	
—	—	—	1
—	—	—	0
—	—	—	0
—	—	—	1
—	—	—	;

variable du supervision

X

↑
données des clients (salaires, ...)

⇒ l'ordinateur explore ces données et sait ce qui caractérise ceux qui ont remboursé leurs crédits et ceux qui n'ont pas remboursé le crédit.

⇒ l'ordinateur nous fournit le programme qui va utiliser sur de nouveaux clients.

* sa permet aussi de détecter les fraudes pour les cartes de crédit.

non supervisé :

* on a des étudiants et leurs matos générale, on souhaite à la fin d'année

affecter les étudiants aux départements (on m'a pas le Y, le département affecté)

↳ l'ordinateur fait de la classification et regroupe les étudiants selon des caractéristiques. Par ex on a 20 étudiants qui ont une boîte en C donc on doit les affecter au département im fo 0000

↳ l'ordinateur fait des groupes sans lui donner le Y.

ex supervisé : ex des chats et chiens, voiture autonome, actions de la Banque

Deux phases pour les méthodes d'apprentissage supervisé :

→ phase d'apprentissage.

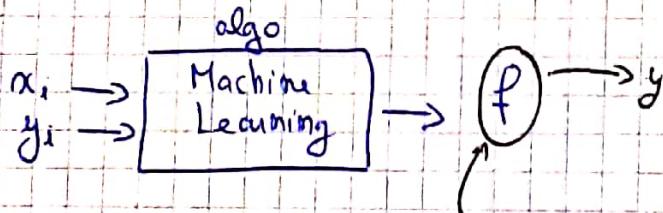
→ phase de prédiction.

Supervisé :

fonction $X \rightarrow Y$

→ dans la phase d'apprentissage on lui donne bcp de X et de Y:

$$D = \{(x_i, y_i)\}_{i=1}^N \xrightarrow{\text{Big data}}$$



on : on donne et la fonction f nous donne le y

ex supervisé :

→ Traduction : on donne à l'ordinateur bcp de phrases en français et en anglais. Ensuite il nous fournit un programme qui permet de traduire les phrases du ph français en anglais

$$X \xrightarrow{F} Y_A$$

→ chats / chiens : Images \rightarrow Identification

Le programme permet de savoir si une image est un chat ou un chien.

→ Avis des clients (Amazon) : permet de savoir si un avis est positif ou négatif.

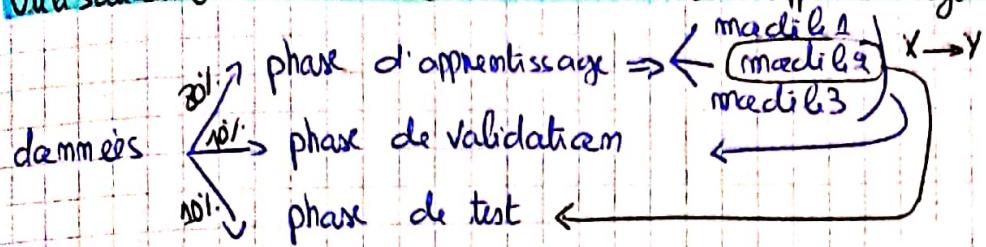
ex main supervisé :

- donner à l'ordinateur plusieurs d'images de chats et de chiens et ensuite lui faire regrouper en 2 : groupe de chat et groupe de chiens.

ex supervisé :

- donner à l'ordinateur bcp d'images de chats et de chiens et lui dire le type de chaque image. Ensuite quand on lui donnera une nouvelle image il connaîtra son type.

Utilisation des données dans les méthodes d'apprentissage supervisé. p. 93



* la phase de validation permet de choisir le meilleur modèle.

* la phase de test permet de tester le meilleur modèle.

Problème de Netflix.

utilisateurs / films	f_1	f_2	\vdots	f_m
U_1	3	1		
U_2				
:				
U_N				

matrice

- l'utilisateur donne une note / 5 aux films qu'il a vus.
- on a des cases vides

- Faire un programme qui permet de proposer des films aux utilisateurs, d'après les films que vous avez vu, voilà ce que je vous propose comme film à voir l'utilisateur à la fin doit aimer le film.

↳ système de recommandation avec du Machine Learning.

↳ en demandant le fichier de validation et celui de test et celui d'apprentissage

Quelques applications du data mining - partie

Applications du DM p 25 : 26

Up selling \Rightarrow pousser le client à acheter qq chose de plus grande valeur que ce qu'il voulait acheter au début (ex: voiture)

Cross selling \Rightarrow pousser le client à acheter d'autres choses.

Related Fields: p 27

Machine Learning

Databases

Visualizations

Statistic

DM and Knowledge Discovery

① Analyse en composantes principales ACP:

\Rightarrow méthode non supervisée

\Rightarrow les colonnes (attributs) doivent être quantitatives. ⚠

\Rightarrow les données \rightarrow

\rightarrow les X

Rappel 8

Attribut (Feature)

quantitative

qualitative

une donnée est dite quantitative si on peut calculer sa moyenne
ex:

- prix d'un produit dans la magasin, caisse faître ...

une donnée est dite qualitative si on ne peut pas calculer sa moyenne
ex:

maternité
sexe

continu

discret

age
- prix
- poids
- taille de quelqu'un

mb de cigarette
mb d'enfants
d'une famille
mb d'habitants
d'une ville
mb d'absence
d'un élève

ordinal

nominal

on peut ordonner les valeurs

ex:

marks (A, B, C)
grades (carré)

ex:

sexe
maternité
mots d'école

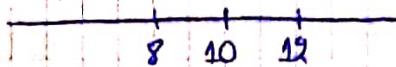
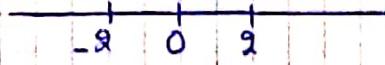
Centrer et réduire les données

x_i	centrier
8	-2
10	0
12	2

$$\text{moyenne } \bar{x} = 10$$

on fait une translation des données

ça devient



centrer \Rightarrow enlever la moyenne de toutes les observations.

$X_{\text{centré}}$

$X_{\text{centré réduit}}$

on divise par l'écart type

8	-2	-1
10	0	0
12	2	1

$$\bar{x} = 10$$

échantillon $\rightarrow n-1$

toute la population $\rightarrow n$

$$\text{variance } (x) = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

estimation de la variance

$$\text{Var} = \frac{(8-10)^2 + (10-10)^2 + (12-10)^2}{2} = 4$$

$$\rightarrow \text{en travailler avec l'écart-type} = \sqrt{\text{variance}} = 2$$

\hookrightarrow pour avoir la même unité de mesure du départ.

Pour centrer et réduire ?

\rightarrow on a les notes des étudiants / 100 \Rightarrow on se débarrasse des données de même sorte Δ .

Centrer Réduire \Rightarrow pour se débarrasser des données de même sorte.
ca d'notes / 100 ou / 100 sa serait la même chose.

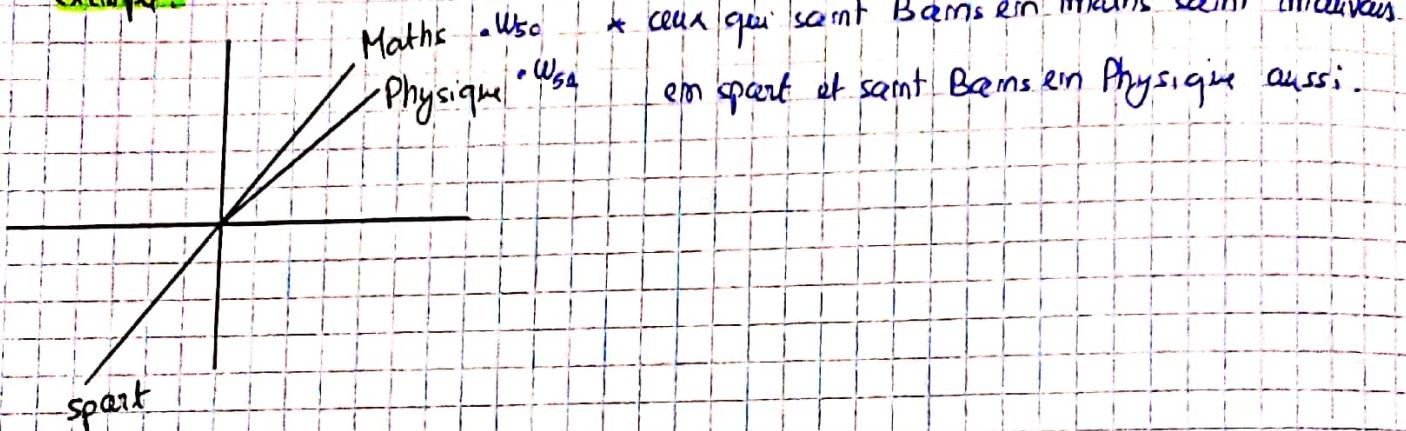
$$\text{moyenne} = 0 \quad \text{en \$ on dit} \rightarrow \text{même chose}$$

$$\text{variance} = 1 \quad \Rightarrow \text{se débarrasser de l'effet taille cm, m (100 x 100)}$$

NB la moyenne et l'incertitude sont toutes les 2 importantes à savoir.

- plus l'angle entre les variables (colonnes) est petit plus les variables sont corrélées entre eux. (corrélées positivement)
- ⇒ Plus je consomme de légumes plus je consomme de fruits et plus je consomme de légumes
- si les variables sont perpendiculaires ⇒ les variables sont indépendantes.
- * l'appartenance des individus ⇒ la ressemblance.
 - * l'appartenance des variables ⇒ corrélation.
- ACP ⇒ prémute le meilleur plan - celui qui permet de perdre moins d'information
- ↳ analyse en composantes principales (Méthode non supervisée)

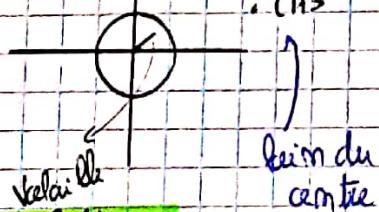
exemple :



* l'appartenance entre une variable et un individu à un sens si l'on sait que l'individu est caractérisé par du centre de gravité du nuage ⇒ on dit dans ce cas que l'individu est caractérisé par la variable !

⇒ les individus consommant beaucoup de

légumes.



Matrice de corrélation

	Pain	Légumes	Fruits
Pain	1		
Légumes		1	
Fruits			1
Viande			0,96

⇒ La corrélation entre Viande et Fruits est très grande 0,96 ...

! Trop qué ? ⇒ les deux sont corrélés
⇒ c'est pas abordable

Objectif de Réduire et Centrer : c'est pourquoi les données doivent être échelle et pour que une variable ne soit pas plus privilégiée qu'une autre.

semaine ② 8 05/04/2022

Rappel :

- * Data mining :
- * Data warehouse : entrepôt de données : c'est là où on prépare les données pour faire du data mining.
visitez avec laquelle les données convergent.
- * Data science :
- * Big data : 3V : Volume, vitesse, variété ← (image, vidéo, texte, table BD...)
- * Data Mart : sous data warehouse sur un sujet particulier = Magasin de données.
- * ETL : extract transform Load.
- * OLAP : on ligne analytical processing / OLTP : on ligne transactionnel processing.
- * Méthodes supervisées : on partage les fichiers en : fichier d'apprentissage, fichier de validation, fichier de test.
- * Méthodes non supervisées :

↳ ACP : analyse en composantes principales.

ETL \Rightarrow logiciel qui extrait et transforme les données et ensuite il les charge dans un data warehouse.

OLTP \Rightarrow la collecte de données (l'opération de retrait d'argent d'un guichet engendre un enregistrement de données dans une BD)

OLAP \Rightarrow analyse des données.

Méthode supervisée : possède une variable privilégiée Y qu'on cherche à prédire

Méthode non supervisée : ne possède pas la variable Y. Y'a rien à prédire

↳ ex : ACP : on veut seulement comprendre les données.

↳ 3 fichiers : * d'apprentissage : apprendre le modèle.

* de validation : choisir le meilleur modèle parmi les modèles qu'on a développé.

* fichier test : évaluer le modèle choisi.

② Analyse Factorielle Discriminante (AFC) visuelle

→ c'est une méthode supervisée

- * 3^e age : vieux / geriatrice : spécialité des vieux.
- * on va contrer l'âge et le poids \Rightarrow ça devrait être la moyenne pour chaque donnée.
- \Rightarrow [pour travailler avec une méthode supervisée on peut ajouter une colonne Z si l'individu est indépendant : 0 s'il est dépendant et 1.]
- * Peut-on prévoir le groupe d'affectation en se basant uniquement sur le poids et l'âge ? non

* mais : si on construit $Y_1 - Y_2$ où

$$Y_1 - Y_2 > 0 \Rightarrow \text{indépendant}$$

$$Y_1 - Y_2 < 0 \Rightarrow \text{dépendant}$$

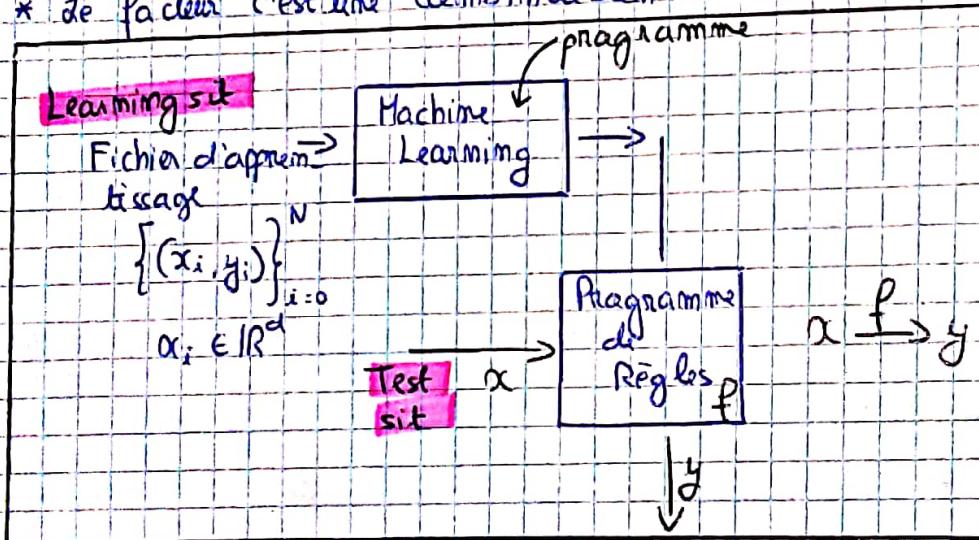
→ le poids et l'âge permettent de différencier

les groupes G_1 et G_2 lorsqu'ils sont considérés ensemble.

$y = X$ (droite qui sépare les 2 groupes)

$y - X > 0$ indépendant y : âge
 $y - X < 0$ dépendant X : poids

* de la façon c'est une combinaison linéaire de la consommation (cache, emploi, manuel)



Résumé :

Méthodes / Algos du data mining

Supervisé

* AFD

non supervisé

* ACP

③ Analyse Factorielle des correspondances AFC.

→ méthode non supervisée

→ 2 caractères qualitatifs.

→ tableau de contingence ou de correspondance c'est un tableau tq la somme d'une ligne a un sens et la somme d'une colonne a un sens.

ex

Age	Paris

⇒ la somme d'une colonne a un sens mais la somme d'une ligne n'a pas un sens ⇒ donc ce n'est pas un tableau de contingence. X

ex : Notes des étudiants X

Maths	Stat	Riaba	Sexe

variable de supervision

→ en vous demandant les notes d'un étudiant est ce que vous pouvez savoir si c'est une fille ou un garçon.

c'est un tableau de contingence ✓

• Pour faire du non supervisé on prend uniquement cette partie et on utilise l'ACP par exemple

non supervisé : on peut utiliser aussi AFC car la somme d'une ligne a un sens c'est la somme des notes d'un étudiant dans toutes les matières. ET la somme d'une colonne c'est la somme des notes en une matière de tous les étudiants et elle a un sens.

NB quand on a les m'unités il peut être si on fait un tableau pour l'ACP et un tableau pour l'AFC.

* AFC ⇒ on doit utiliser les proportions.

NB : age : <15 ; 15 < 24 ; 25 < 39 ⇒ c'est qualitatif. D

Exemple AFC: Département et centre prépa des étudiants

Info	ELM	GT	...	Centre prépa



	INFO	IP	ELM	ooo	Total
Fès	10	3			
Rabat					
ooo					
Total					

Tableau de contingence

⇒ la somme d'une ligne et la somme d'une colonne ont un sens.

→ ensuite on projette sur un axe par ex

- ELM
- FFS

• les gens qui viennent du FFS choisissent

le département ELM. ET ceux qui viennent

- INFO
- Marakech

de Marakech préfèrent plus l'INFO au

choisissent plus l'INFO que l'ELM.

AFC ⇒ croisement entre 2 variables qualitatives de modalités différentes.

exemple :

CSP = variable qualitative

↳ catégories socioprofessionnelles

↳ ouvriers - employés - cadres - fonctionnaires - ... modalités de CSP

	MA	CA	EM	oo.
Cancer				
Covid				
oo.				

← CSP

⇒ Tableau de contingence

↑ causes de mort

- Ouvrier
- Cadre

→ on fait une projection

On remarque que :

→ la cause de maladie pour les cadres

- cadre
- diabète

c'est la diabète car ils consomment

beaucoup de sucre..

→ la cause de maladie pour les ouvriers c'est la cirrhose car ils consomment trop d'alcool

exemple :

Pays	France	UK	USA	...
Prix Nobel				
Physique				
Chimie				
Médecine				
...				

=> Tableau de contingence.

On va remarquer que : la France attire le prix en Maths, USA c'est la médecine, et la physique ...

exemple :

Un membre d'un:

→ a pris le coram et il a pris 2 variables qualitatives : les mots qui se trouvent dans le Coram ensuite il a pris les mots qui se trouvent dans le coram.

الكلافين	العنوان	الله	السور
الخطاب			

← la somme d'une ligne est le nb de mots de **السور**
← l'intersection d'une ligne et une colonne
est le nb de fois qu'un mot est répété dans **الكلافين**

↑ tableau de contingence

→ ensuite on projete tout :

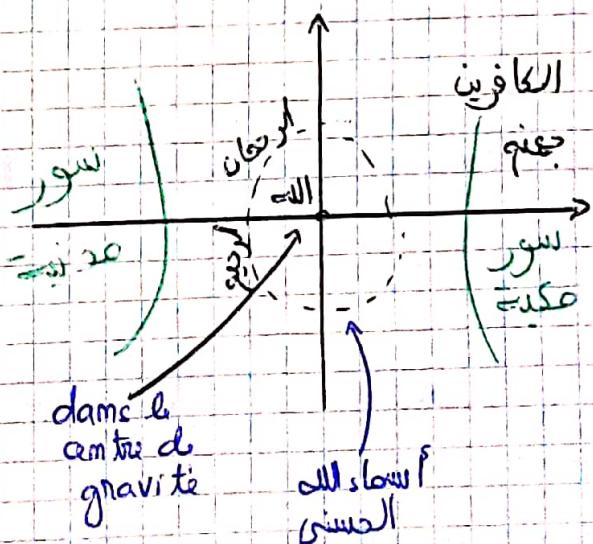
Résumé :

Méthodes du DM :

Supervisé
- AFD

Nan Supervisé :

- ACP
- AFC



① Arbres de décision :

- méthode supervisée.
- Refund : remboursement (déclare que vous n'avez pas assez suffisamment d'argent)
- Taxable income . Revenue Brut.
- * des arbres de décisions : acceptent les variables quantitatives et aussi les variables qualitatives.

* am va faire un modèle supervisé qui nous permettra par la suite de déterminer si une personne a triché ou pas à partir des 3 variables (Refund, Marital status, Taxable income).

↳ le modèle s'appelle un arbre de décision.

méthode supervisée : méthode de classification .

class : label : variable privilégiée.

→ objectif c'est de faire un modèle qui nous permettra de savoir si une personne a triché ou pas.

Comment construit-on le modèle ?

→ on prend la colonne de class (racine) au point départ.

→ on a 3 mb tricheurs et 7 mb non tricheurs.

→ on suit, on prend la variable REFUND.

Y : nb tricheurs | N : nb non tricheurs

→ on cherche des variables qui nous donneront des résultats le plus purs

possible.

→ Tous ceux qui demandent un REFUND sont des non tricheurs.

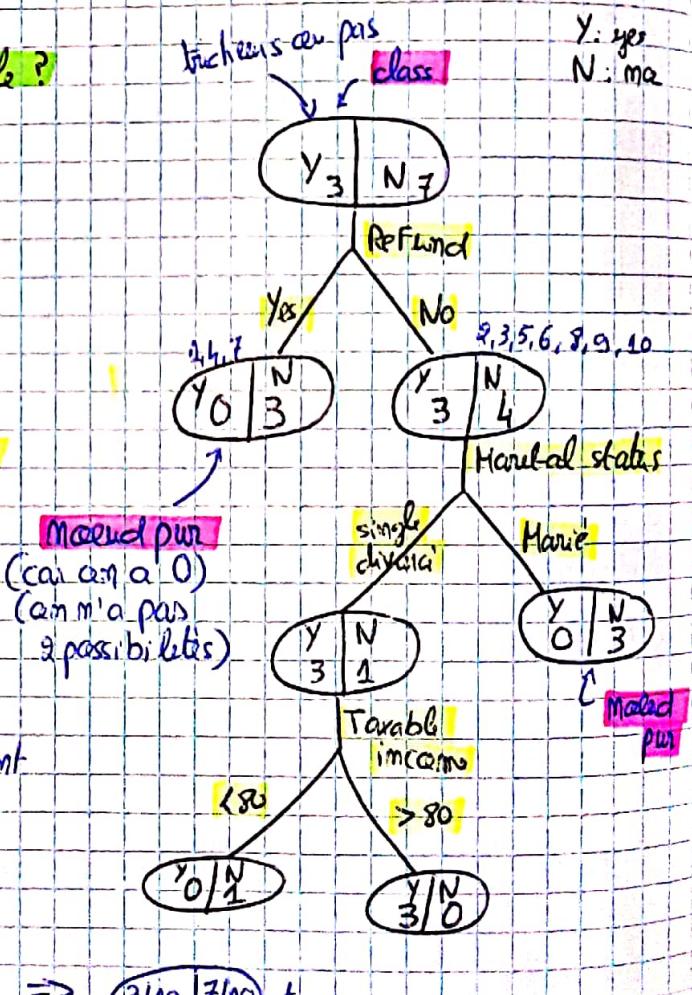
Indice de Gini :

on a par exemple un mélange



$$\Rightarrow \frac{3}{10} | \frac{7}{10}$$

→ on calcule l'indice de Gini de ce mélange :



$$GINI(t) = 1 - \left(\left(\frac{3}{10}\right)^2 + \left(\frac{7}{10}\right)^2 \right)$$

exemple: si on a plusieurs modalités.

CA	MA	EM
1/3	1/3	1/3

\Leftarrow

CA	MA	EM
1	1	1

$$GINI = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right)$$

$$GINI = \frac{2}{3}$$

\rightarrow plus l'indice de GINI est petit plus le nœud est pur. Δ

\rightarrow Nœud pur $\Rightarrow GINI = 0$ ($1 - 0^2 - 1^2 = 0$)

\rightarrow l'indice de GINI est maximal si les proportions sont égales.

exemple:

P	1-P
---	-----

$$GINI(t) = 1 - (P^2 + (1-P)^2)$$

$$= 1 - P^2 - 1 + 2P - P^2$$

$$= 2P - 2P^2$$

$$GINI(t) = 2P(1-P) \Leftarrow$$

l'variance quand la variable est Binaire

\Rightarrow quand le nœud est

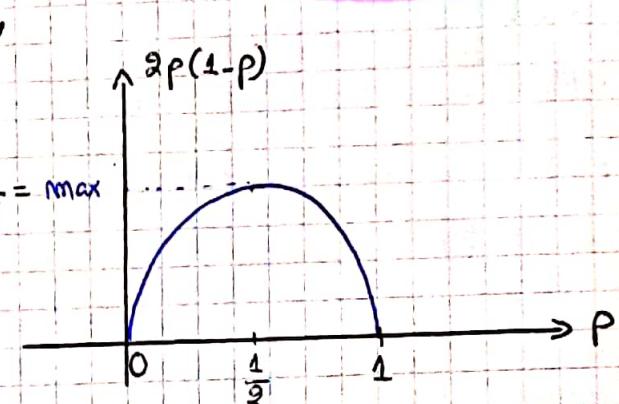
50%; 50%

l'indice de GINI

$$\text{est maximal} = \frac{1}{2}$$

\Rightarrow l'indice de GINI est proportionnel à la variance.

\Rightarrow Variance $\uparrow \Rightarrow$ nœud impur.



NB: On choisit l'attribut ayant l'indice de GINI plus petit. Δ

* on a 2 branches : une pour les attributs et l'autre pour les modalités.

↓
REFUND
Marital status
Taxable Income

Exemple,

$\rightarrow x < x_1 \Rightarrow$ Bleu lancé

$\rightarrow x > x_1$ et $y < y_1 \Rightarrow$ Bleu vel

$\rightarrow x > x_1$ et $y_1 < y < y_2 \Rightarrow$ Orange

$\rightarrow x > x_1$ et $y > y_2 \Rightarrow$ Vert

Matrice de confusion :

Predict

	B	R	BC	V
B	9	0	0	1
R	0	5	0	0
BC	0	0	4	0
V	0	1	0	7

→ combien de Bleu sont été classés Bleu : 9

→ combien de Bleu sont été classés vert : 1 000

$$\frac{25}{2+25} = \frac{25}{27} = \text{taux de classement}$$

exemple :

à partir de la consommation des fruits et du pain des individus on peut savoir si c'est un MA ou un CA ou un EM

TP 1 :

④ Decision tree (Fichier consommation)

* Node Repository : contient tous les noeuds

* New File pour créer un workspace

* workspace : c'est là où on va créer le programme du machine learning (DM) qui va créer le modèle.

→ Excel Reader : pour lire les données.

↳ clique droit → configurer → pour indiquer le fichier à lire.

↳ jaune : donc il est prêt à être exécuté

↳ clic droit : → execute

* Vert : il a lu le fichier

↳ clic droit → File table pour lire le fichier qu'il a lu

→ Decision tree banner

↳ clic droit → can figure → Aply ⇒ jaune

↳ clic droit → execute → vert

- droit clic droit → execute and open view
- ↳ Arbre de décision
- ⇒ color manager : permet d'appliquer un style à l'arbre
- ↳ clic gauche, choisir des couleurs
- ↳ clic droit : execute
- ↳ clic droit sur Decision tree browser et : execute and open view.
- * ajuste le modèle avec les données d'apprentissage.
- decision tree predictor : ajoute une variable (CSP prediction)
- * on divise les données en : données d'apprentissage + données de test.
- ↑ prediction obtenue avec le modèle.
- scanner : génère la table de confusion.
- ↳ configurer → execute and open view ⇒ Table de confusion
- $\begin{cases} \text{Taux d'erreur} = \frac{1}{12} = 8,33\% \\ \text{Taux de Bonne classification} = \frac{11}{12} = 91,66\% \end{cases}$

② Décision tree (fichier Iris)

- 3 types de fleurs : 50 - 50 - 50
- calculer la longueur et largeur de pétil et de sépal. ⇒ déterminer le type de la fleur
- préduire : l'espèce de la fleur
- * pour tester : → partitionning
- on divise les données en 2
- ↳ fichier d'apprentissage
- ↳ fichier de test.

NB: il faut pas tester le modèle sur les données qui ont été utilisées pour la construction.

stratified sampling:

80%	du type 1	⇒ m% pourcentage de données pour
80%	du type 2	chaque catégorie.
80%	du type 3	

First partition : pour l'apprentissage

Second partition : pour le test. (Prédiction)

→ PMML : permet de sauvegarder un modèle pour qu'on puisse l'utiliser après.
writer
⇒ on le met qq part dans l'ordinateur.

si on veut l'utiliser :

→ PMML reader : et on lui donne le modèle qu'on a conservé.

+ Excel Reader, Fichier de données.

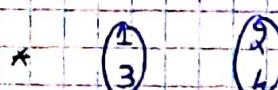
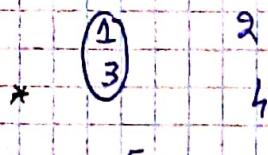
+ Decision Tree Predictor

(5) Classification non supervisée (Clustering)

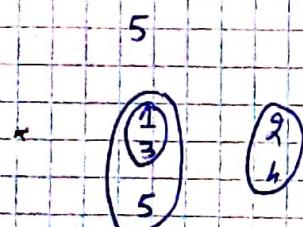
→ méthode non supervisée

→ clustering : faire des classes homogènes. (des partitions)

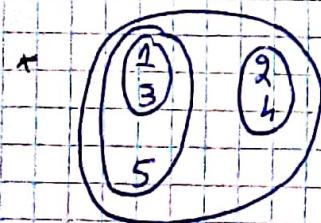
- 1 et 3 sont les 2 éléments les plus proches.



- distance entre 2 et 4 c'est $\sqrt{2}$



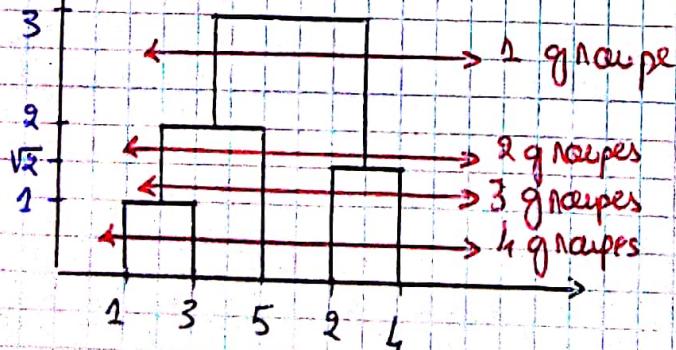
- 5 est plus proche du groupe 1-3 que 2-4



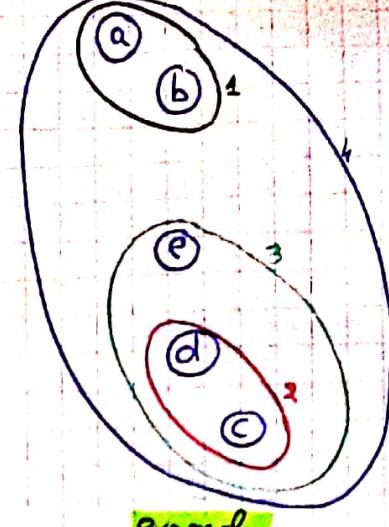
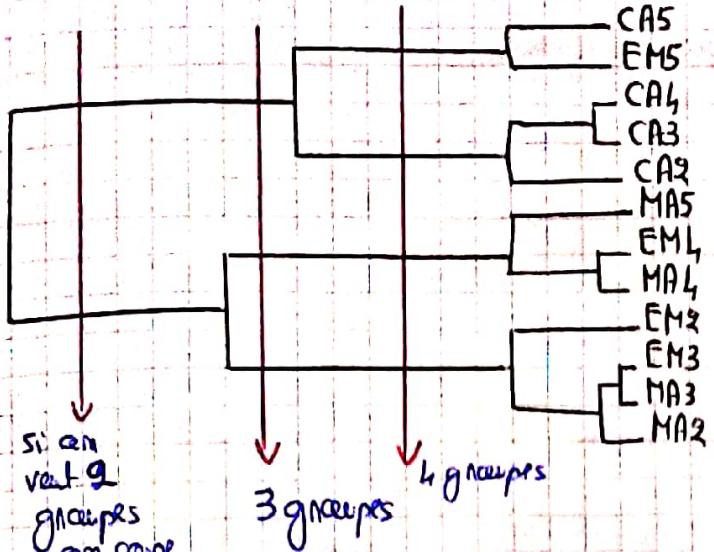
distanc

arbre de classification.

- si on veut 3 groupes homogènes



exemple :



exemple

TP)

① Clustering :

→ excel reader : fichier consommation

→ Hierarchical clustering :

↳ arbre 1

- * Il construit un arbre de classification sans savoir CSP de chaque individu.

Résumé :

supervisé

→ AFD
→ AFD arbres de décision

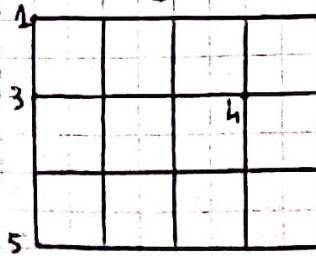
Algorithmes / Méthodes DM :

non supervisé

→ ACP
→ AFC
→ clustering

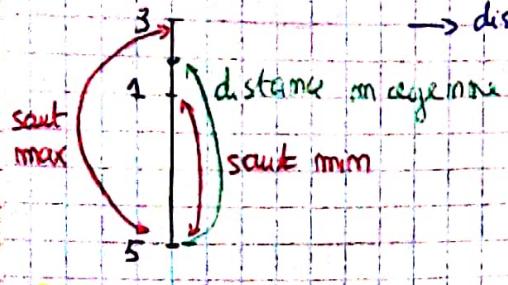
Séance ③ : 19/04/2021

⇒ Clustering hiérarchique :

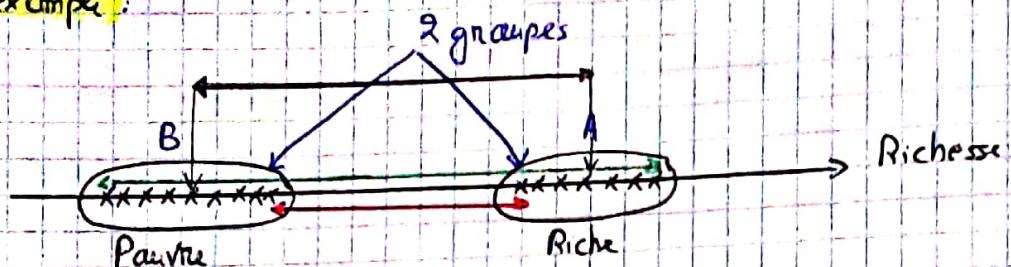


- * on cherche toujours les 2 éléments les plus proches que l'on regroupe
- * il faut définir 2 choses :
 - une distance, ici on a utilisé une distance euclidienne.
 - la distance entre un groupe et un élément :

\Rightarrow il y a 3 chaînes : \rightarrow saut-min
 \rightarrow saut-max
 \rightarrow distance moyenne



exemple :



$\forall (A, B)$
 on peut définir la distance entre A et B de 3 façons : (en fait choisir une).

\rightarrow distance entre le plus pauvre des riches et le plus riche des pauvres.

\Rightarrow saut-min

\rightarrow distance entre le plus pauvre des pauvres et le plus riche des riches.

\Rightarrow saut max

\rightarrow distance entre le centre de gravité de A et le centre de gravité de B (moyenne)

\Rightarrow distance moyenne

* dans notre exemple on a choisi le saut min.

NB: AFD, AFC, ACP sont des méthodes factuelles car on projette sur des facteurs facteurs (ou axes)

⑥ classification non supervisée par Centres Mobiles (K-means)

\rightarrow méthode de non supervisée

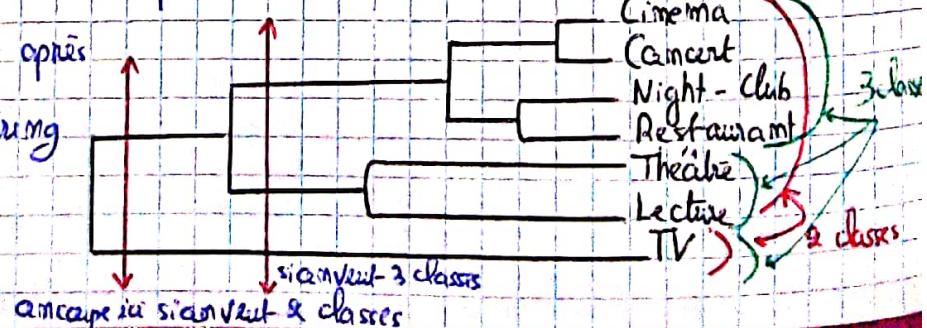
\rightarrow means = moyennes

* clustering hiérarchique on ne choisit pas les membres de classes au début. Mais à la fin on peut prendre le nombre qu'on veut de classes

* le nb de classes est défini après

avoir établi l'arbre de clustering

hiérarchique.



- par contre K-means : on définit dès le début le nombre de classes. (R)
- par exemple : on a 300 élèves et on souhaite constituer 6 groupes homogènes.
- chaque classe contient les éléments qui se ressemblent le plus. Et entre une classe et une autre les éléments sont très différents.
- ensuite on doit choisir les points au hasard. Soit on les choisit parmi les éléments soit on choisit d'autres points. = **center cluster centers**
- on affecte chaque point au centre le plus proche. (cluster center le plus proche).
- on recalcule le centre de gravité de chaque groupe.
- on déplace les cluster centers à aux nouveaux centres de gravité.
- on recommence. On réaffecte chaque point au centre le plus proche.
- 3 points ont changé de groupe. On recalcule le centre de gravité de chaque groupe.
- on affecte les points.
- on répète ces étapes jusqu'à convergence. C'est jusqu'à ce que les centres ne bougent presque pas.
- on démontre que le plus souvent l'algorithme converge.
- 2 étapes se succèdent :
- { Affecter chaque objet au centre le plus proche }
 - { Recalculer les centres des classes constituées. }

Algorithme

* matériels :

$$\rightarrow \hat{x} = \arg \min_x f(x) \Leftrightarrow f(\hat{x}) \leq f(x) \quad \forall x$$

→ Indicateur d'une prédiction :

$$I(P) = \begin{cases} 1 & ; \text{ si } P \text{ est vrai} \\ 0 & ; \text{ sinon} \end{cases}$$

→ méthode supervisée :

$$D = \{(x_i, y_i)\}_{i=1}^N : \text{data set}$$

$x_i \in \mathbb{R}^d$: $y_i \in \mathbb{R}$: **régression**
d : coefficients $y_i \in [L_1, L_2, \dots]$: **classification**
 L : labels

* Algo :

- lire D // nb de classes.

→ par ex les mesures de la fleur

$\{x_i \Rightarrow$ par ex les mesures de la fleur.

$y_i \Rightarrow$ l'espèce de la fleur.

ou $\{x_i \Rightarrow$ consommation des individus.

$y_i \Rightarrow$ CSP (cadre, manuel, employé).

$\{x_i : \text{vecteur des features} ; x_i \in \mathbb{R}^d\}$, d : nb de features
 $\{y_i : \text{label} ; y_i \in \mathbb{R} \Rightarrow \text{régression}$ ou $y_i \in \{L_1, L_2, \dots, L_C\} \Rightarrow \text{classification}\}$
 des labels.

→ **Machine supervisée**:

$$D = \{(x_i)\}_{i=1}^N ; x_i \in \mathbb{R}^d$$

D : data set = fichier de données.

* **Algèbre**: $D = \{(x_i)\}_{i=1}^N, x_i \in \mathbb{R}^d$; **Algèbre de K-means**

lire b

- tirer $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^d$ aléatoirement. Il choisit les centres au hasard.

Répéter:

choisir pour $i = 1 \text{ à } N$ faire:

le centre le plus proche $\left\{ C_i = \arg \min_{1 \leq j \leq k} \|x_i - \mu_j\| \right\}$ // C_i qui rend cette distance $\|x_i - \mu_j\|$ plus petite

pour chaque point $\|1 \leq i \leq N\|$

\Rightarrow pour chaque point on a le centre de gravité le plus proche

// on cherche l'indice du centre de gravité le plus proche à x_i .

⇒ maintenant calculez le nouveau centre de gravité

mis en œuvre pour $j = 1 \text{ à } k$ faire:

$$\left\{ \mu_j = \frac{\sum_{i=1}^N x_i I(C_i = j)}{\sum_{i=1}^N I(C_i = j)} \right\}$$

- jusqu'à convergence

→ **Algèbre propre**:

lire b

- tirer $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^d$ aléatoirement

- Répéter:

pour $i = 1 \text{ à } N$ faire:

$$\left\{ C_i = \arg \min_{1 \leq j \leq k} \|x_i - \mu_j\| \right\} // 1 \leq C_i \leq k$$

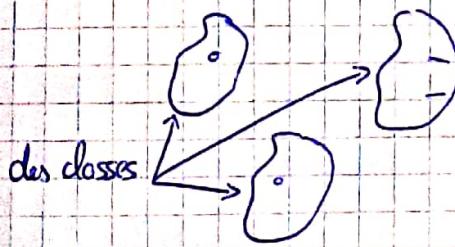
pour $j = 1 \text{ à } k$ faire:

$$\left\{ \mu_j = \frac{\sum_{i=1}^N x_i I(C_i = j)}{\sum_{i=1}^N I(C_i = j)} \right\}$$

- jusqu'à convergence

suite de la séance ③

- c_i : c'est l'indice du centre le plus proche.
- μ_j : mise à jour du centre de gravité.
- la convergence: les μ_j ne bougent presque pas.
- $\sum_{i=1}^n \alpha_i I(C_i = j)$: on somme le poids des éléments qui sont proches de μ_j .
- $\sum_{i=1}^n I(C_i = j)$: le nombre d'éléments proches de μ_j .
- de convergence
- il faut diminuer la fonction de coût (la réduire).
- $\text{coût} = \sum_{j=1}^k \sum_{i \in \text{cluster } j} \|x_i - \mu_j\|^2$
- $\sum_{i \in \text{cluster } j} \|x_i - \mu_j\|^2$: somme à l'intérieur d'une classe.
- c'est la distance entre chaque point et le centre de gravité le plus proche, au carré
- $\sum_{j=1}^k \sum_{i \in \text{cluster } j} \|x_i - \mu_j\|^2$: la somme de toutes les sommes.
- le total de toutes les sommes doit être le plus petit possible



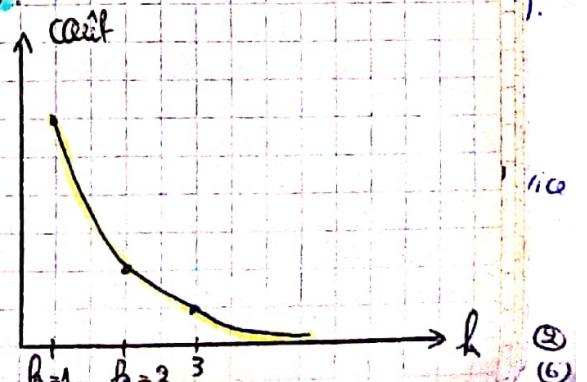
→ on peut montrer que la fonction "coût" ne fait que diminuer au cours des itérations. Jusqu'à ce qu'il ne bouge presque plus \Rightarrow convergence.

Choix du k : membres de classes (de groupes) :

→ on essaye plusieurs k et on calcule le coût.

→ $k \uparrow \Rightarrow \text{coût} \downarrow$

→ $k = m$: signifie que chaque point est dans un cluster \Rightarrow coût = nulla



- on s'arrête quand le coût commence à se stabiliser.
- cet algorithme de classe car la fonction f_k ressemble au coûte de f_k
- on choisit par exemple $f_k = 3$ si on remarque que le coût ne change pas trop avec le nombre de classes.
- on essaie avec tous les nombres de classes et puis on s'arrête avec le f_k où le coût ne change pas trop.
- on fixe un f_k et on tourne l'algorithme jusqu'à ce que le coût diminue.
- pour chaque f_k on refait le travail, on tourne l'algorithme pour $f_k = 1$ on fait tourner l'algorithme et sa se stabilise le coût. Pour $f_k = 2$... et on voit quel est le f_k dans lequel le coût ne change pas.
- ⇒ c'est la méthode pour choisir le nombre de classes f_k .
- K-means = K-moyennes = centre mobile.

Rappel:

Algorithmes de DM:

* : méthodes factorielles

supervisés :

* AFD

arbres de décision

non supervisés :

* AFC

* ACP

clustering [hiérarchique]

K-means

④ Réseaux de neurones

→ méthode supervisée.

→ on utilise les réseaux de neurones surtout dans les choses que l'être humain fait rapidement par exemple : la reconnaissance d'image, la traduction automatique de la voix.

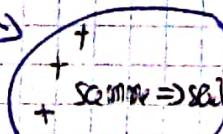
→ Réseaux de neurones : simulation du fonctionnement du cerveau humain

signaux

provenant

d'autres

Neurones



signal transmis si

le seuil est franchi

- ex de signaux d'entrée : consommation du pain, consommation du vin...
- on somme les signaux d'entrée avec des poids déterminés (chaque consommation est multipliée par son poids)

- si la somme dépasse une certaine valeur on dit que c'est un cache en un employé au manuel.

de la fonction d'activation :

- par exemple : le signe : si le signe est positif on renvoie $+1$, s'il est négatif on fait sortir -1

→ exemple : la fonction sigmoïde : $f(x) = \frac{1}{1 + e^{-x}}$; $f(0) = \frac{1}{2}$
 $\lim_{x \rightarrow +\infty} f(x) = 1$; $\lim_{x \rightarrow -\infty} f(x) = 0$

⇒ parfois on l'utilise pour mesurer la probabilité, car la proba est entre $[0, 1]$; $x \rightarrow +\infty \Rightarrow 1$

$$\begin{array}{lll} x \rightarrow -\infty & \Rightarrow & 0 \\ x = 0 & \Rightarrow & \frac{1}{2} \end{array}$$

* la fonction tangente hyperbolique: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; $f(x) \in]-1, 1[$

* la fonction step: $f(x) = 1$ si $x > 0$, 0 sinon

* la fonction relu: rectified linear unit $f(x) = x$ si $x > 0$, 0 sinon

Organisation en réseau

* le réseau de neurones

→ contient des entrées par exemple 7 consommations

→ on a en sorte des neurones, chaque neurone prend les entrées multipliées par des poids

→ chaque neurone est relié à toutes les entrées avec des poids différents.

→ si on a 4 entrées et 3 neurones \Rightarrow donc on a 12 poids !

→ ensuite on prend les 3 neurones, on les somme et on définit 3 sorties.

→ des sorties sont des combinaisons linéaires des neurones suivis d'une fonction d'activation.

Exemple :

- * entrées : tests médicaux.
- * sorties : malade ou pas.

⇒ si on a plusieurs couches cachées, on dit que le réseau de neurone est profond.

→ les couches, les poids

→ c'est nous qui définissons l'architecture du réseau : on définit combien de couches on a, et chaque couche combien elle a de neurones.

→ si on ajoute plus de couches, on aura plus de poids.

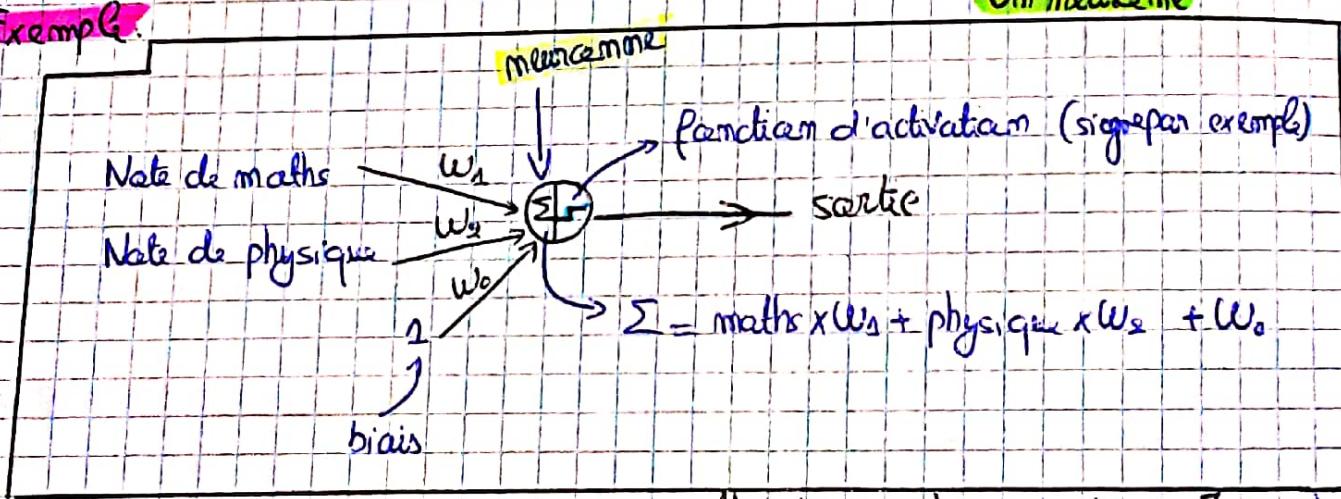
→ les poids sont à apprendre car on ne les connaît pas. On les apprend à partir d'un data set supervisé.

Apprentissage,

→ l'algo le plus utilisé : rétropropagation

→ au cœur des itérations : les poids évoluent ... jusqu'à ce qu'on a de très bonnes prédictions.

Exemple :



Un meilleur

si $\text{maths} \times w_1 + \text{physique} \times w_2 > 7 \rightarrow \text{réussir}$

- w_1 : coeff de maths
- w_2 : coeff de physique
- $w_0 = -7$

sinon → Echec

$\Rightarrow \text{maths} \times w_1 + \text{physique} \times w_2 - 7 > 0 \rightarrow \text{passé}$

→ on peut définir la fonction signe de cette manière :

$$\text{signe}(\text{maths} \times w_1 + \text{physique} \times w_2 - 7)$$

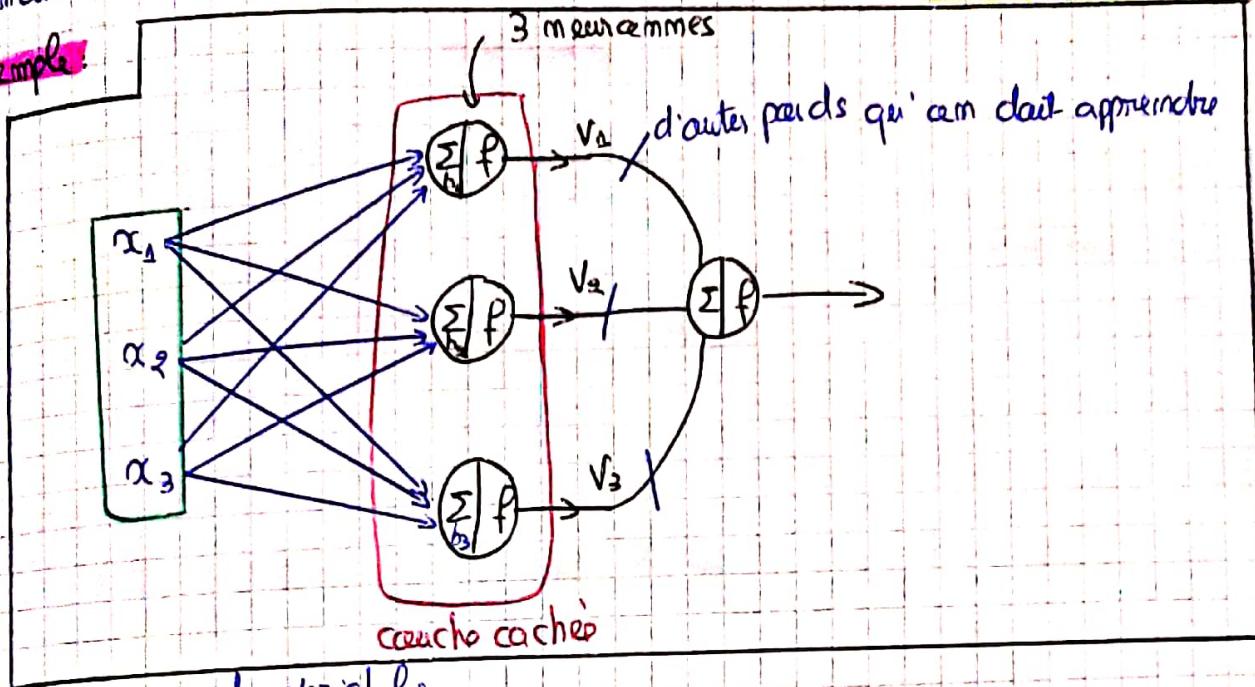
signe ($\text{maths} \times w_1 + \text{physique} \times w_2 - 7$)

fonction d'activation

- la sortie : échec ou réussite.
- le bémot du data scientist c'est de choisir les matières qui déterminent la réussite (matières principales), ensuite il faut établir le modèle, après il faut apprendre W_1, W_2, W_3 , comment les apprendre ? A travers les matières de maths et de physique des amitiés premières + est ce qu'ils ont réussi ou pas, et en fonction de ça on apprend W_1, W_2 et W_3 .

réseau de neurones

Exemple :



- x_1, x_2, x_3 : des variables
- b_1, b_2, b_3 : biais
- on a : $3 \times 3 + \text{les biais} = 3 \times 3 + 3 = 12$ poids Δ à apprendre (on ne les connaît pas)
- chaque neurone a une sortie
- dans ce réseau de neurones on a : 1 seul couche cachée et 3 neurones plus une couche de sortie contenant 1 seul neurone.
- problème fondamental : il faut définir les variables qui vont déterminer la sortie
- on a remarqué que le réseau de neurones peut déterminer les variables les plus intéressantes, il faut donner plus de poids pour sortir le bon résultat.
- Exemple :
- reconnaissance d'image : on lui donne plusieurs photos de honda, et à chaque fois qu'un photo c'est elle dans l'apprentissage on lui dit que c'est une

aux pr
autre sorties et
avec la réalité
métamorphosé
⇒ primaire
amara à phénomènes

→ l'image c'est une matrice de pixels 64x64

→ c'est un vecteur de 64x64

→ pixel : entité valeur entre 0 et 255 : densité de pixel

→ si c'est en noir et blanc cela des valeurs 0 et 255

→ si c'est en couleur cela a 3 matrices : rouge, vert et bleu.

64x64 : noir et blanc) entrées

64x64x3 : en couleurs

→ la reconnaissance d'image est utilisée dans les postes de travail

cp

où une enveloppe contient l'adresse et le code postal.

o le système tire automatiquement les enveloppes par quartiers grâce

à la reconnaissance d'image (reconnaissance de l'écriture).

→ le problème qui se pose quand on détermine les entrées et la sortie (qui est ce qu'on va décider) ⇒ l'algorithme qui tente d'apprendre les poids (les seuls incertains)

→ on définit la Σ et la fonction d'activation

Comment déterminer les poids qu'on va utiliser :

⇒ algorithme de rétropropagation permet d'apprendre les poids

Exemple :

o entrées : notes d'un étudiant

o on initialise les poids d'une façon au hasard.

"l'algo"

o on calcule ...

o on a une sortie qu'on a calculé avec les poids qu'on a tiré au hasard.

o la 1^{ère} sortie va être fausse, on a la vraie sortie y , on compare

la sortie avec y et on définit la fonction coût : est ce que je suis bien au près. Notre objectif c'est de rendre la fonction coût la plus petite possible.

(distance entre réalité et préiction).

o on la dérive pour la rendre minimale, on la dérive % à chaque poids

et on définit une fonction itérative qui améliore les poids, on revient sur

⇒ Impar

→ si on

il est actif

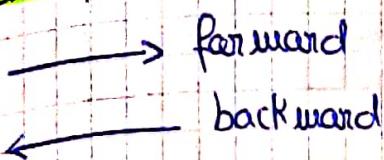
mais non

⇒ qua

de savoir

aux pour mettre à jour les poids - une fois les poids sont mis à jour. On met une autre entrée et on calcule les nouveaux poids. La sortie, on compare la sortie avec la réalité, si on est loin de la réalité on revient marche à l'arrière on en mettant à jour les poids, on introduit une autre entrée ...
⇒ principe de rétropropagation. ↳ L'entrée c'est les poids

on a 2 phases :



→ jusqu'à ce que les poids ne changent plus énormément.

Exemple fichier consommation:

→ on a 24 poids. $8 \times 3 = 24$.

→ ensuite on a 12 poids $4 \times 3 = 12$

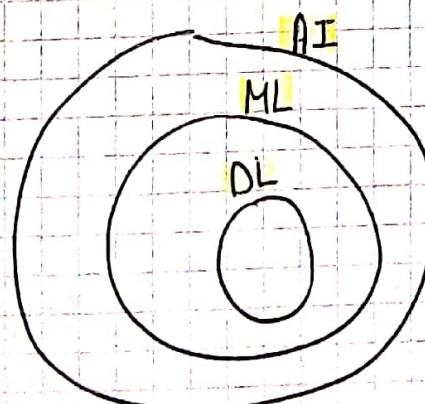
→ au total on a 36 poids à apprendre.

→ pour les sorties : { la première retourne 1 si c'est un cadre, 0 sinon
{ la 2^{ème} " 1 " employé, 0 sinon
{ la 3^{ème} " 1 " manuel, 0 sinon

⇒ Importance standardisée.

→ on a un trait bleu dans le paum par ex aide bien à mesurer le murane, il est actif. Par contre les légumes ne jouent pas un bon rôle dans le 2^{ème} murane...

→ quand l'algorithme finit d'apprendre, il renvoie un graphique permettant de savoir les variables qui jouent le rôle le plus important.



AI : intelligence artificielle

ML : machine learning (DM)

DL : deep learning (réseaux de neurones).

Rappel.

Algorithmes de DM

supervisés

- AFD
- arbres de décisions
- réseaux de neurones

non supervisés

- * méthodes factorielles

AFC

hierarchique

clustering

[K - means]

- panier de ménagerie = méthodes d'association

⑧ Méthodes d'association (Panier de ménagerie)

→ méthode non supervisée.

→ dans les pays développés, ce sont surtout les femmes qui font des courses.

→ les supermarchés demandent aux clients des cartes de fidélité (gagner des points) mais ils comprennent de plus en plus le client avec ça.

→ Panier de ménagerie permet de savoir les produits achetés ensemble. Ceci va bien permettre de faire le rayonnage dans le supermarché. Par exemple : les gens achètent souvent avec le pain du beurre. Donc le supermarché met à côté du pain le beurre pour que les clients n'oublient pas de l'acheter.

↳ recherche d'association = panier de ménagerie.

→ Amazon : lorsque un client achète qq chose ils vous proposent d'autres choses, vous disent que 90% qui ont acheté ça ont aussi acheté ça.

TP 2

(variable)

v_2

x

x

x

x

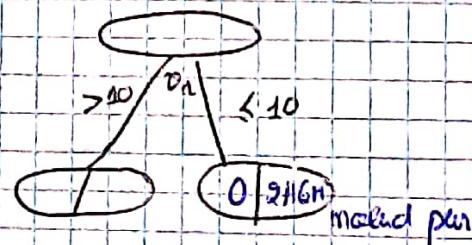
x

ELM

MSIP

v_1

10

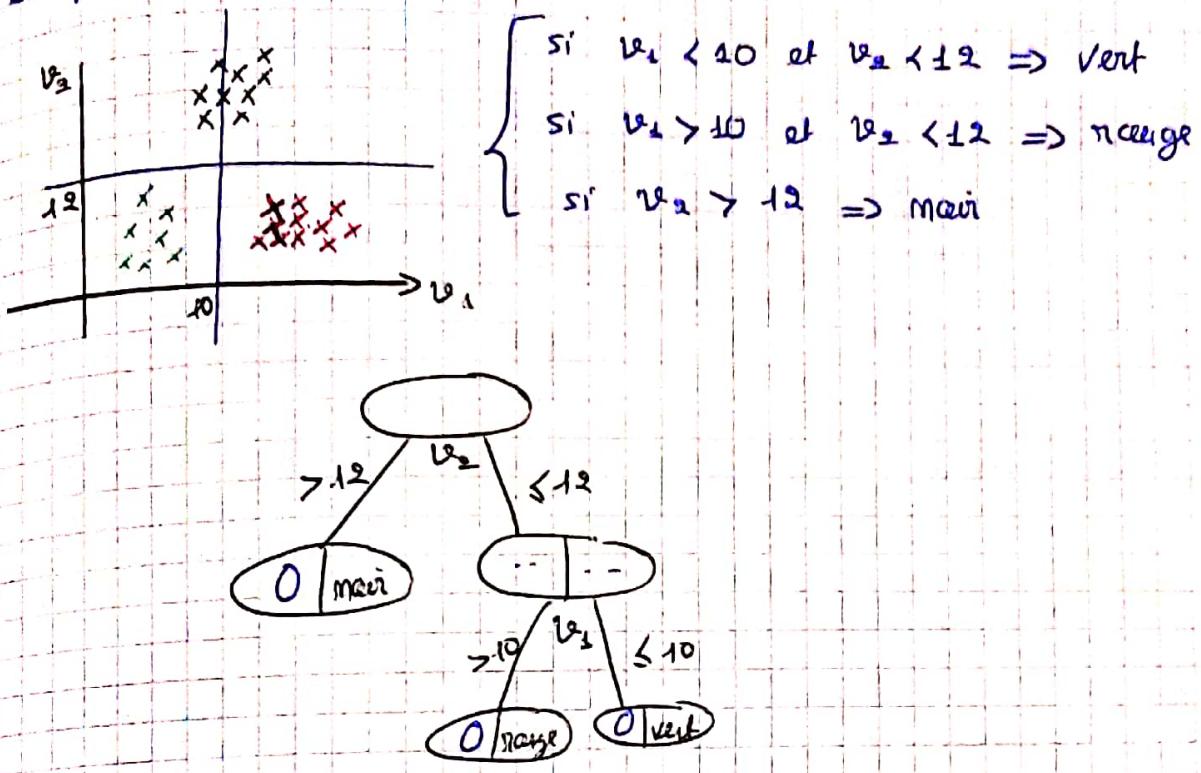


→ a 2 variables par ex les notes de maths et de PC pour 3 départements

→ l'algorithme de l'arbre de décision cherche une variable qui permet de séparer les groupes. Par exemple si on veut séparer 2HGM en utilisant la variable v_1

- l'arbre de décision n'en va pas forcément dans ce cas car si v_1 et v_2 ne permettent de discriminer entre MSIP et ELP. Par contre la variable v_2 permet de discriminer entre 2GM et MIP+ELH.
- l'arbres de décisions fait juste des coupures.

exemple 2:



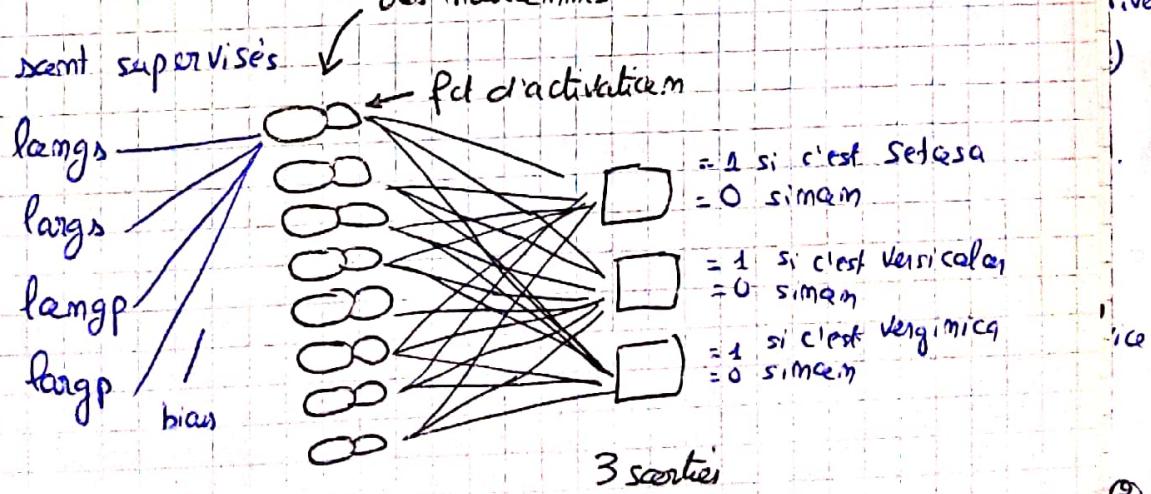
→ Il faut faire de la visualisation des données pour pouvoir décider. ⚡

→ le choix des variables est important. La visualisation permet de choisir les variables intéressantes.

→ dans le TP2, on a comparé le modèle d'arbre de décision et le modèle de réseaux de neurones et on a trouvé que le réseau de neurones est le meilleur modèle.

→ ces 2 modèles sont supervisés

Réseaux de neurones du TP2



→ apprend les produits pour pouvoir faire de la prédiction.

→ on c'est qu'il fait des combinaisons linéaires mais on ne sait pas exactement ce qu'il est en train de calculer.

Suite. Pami ces deux méthodes

→ on cherche à savoir si que les gens achètent en même temps.

Fichier de données

	P_1	P_2	...	P_m	←	produits
Tickets 1	1	0	-	0	1	1. Il a acheté le produit 0. Il n'a pas acheté.
Tickets 2	0	1	1	-	0	
⋮	⋮	⋮	⋮	⋮	⋮	

→ l'algorithme cherche les produits achetés ensemble.

Règle :

condition → Résultat

ex : pain + beurre ⇒ Lait

$C \Rightarrow R$

Indice de confiance: $P(C \text{ et } R) / P(C) = P(R|C)$

Indice de support : $P(C \text{ et } R)$

$$P(A \cap B) = P(A|B) \times P(B)$$

$$\text{gain (rule)} = \frac{\text{Indice de support}}{\text{Indice de confiance}}$$

$$\text{gain}(B \rightarrow E) = \frac{\text{Indice de support}}{\text{Indice de confiance}} = \frac{P(E|B)}{P(E)} = \frac{3/4}{3/5} = \frac{5}{4} > 1$$

$$\text{gain}(B \rightarrow E) = \frac{P(E \cap B)}{P(E) \times P(B)}$$

$$= \frac{\text{Indice de confiance}}{\text{Probabilité d'avoir}}$$

$C \rightarrow R$: règle

\Rightarrow Indice de confiance: $P(C|R|C) = \frac{P(C \cap R)}{P(C)}$

\Rightarrow Indice de support: $P(C \text{ et } R)$

\Rightarrow Lift = gain d'une règle. = $\frac{P(C \text{ et } R)}{P(C) \times P(R)}$

on compare $P(C) \times P(R)$ avec $P(C \text{ et } R)$

\Rightarrow 2 événements indépendants si $P(C \cap R) = P(C) \times P(R)$

\Rightarrow si le gain > 1 \Rightarrow la règle est intéressante que l'absence.

\Rightarrow si le gain < 1 \Rightarrow mauvaise règle

$$\text{gain} = \frac{\text{Praba (candidat et résultat)}}{\text{Praba (candidat) } \times \text{praba (résultat)}}$$

\Rightarrow Règle: $C \rightarrow R$

Règle inverse: $C \rightarrow \bar{R}$

\rightarrow on connaît l'indice de confiance de $C \rightarrow R$

\rightarrow on veut calculer l'indice de confiance de $C \rightarrow \bar{R}$

$$P(C) = P(C \cap \overbrace{R \cup \bar{R}}^{S2})$$

$$= P(C \cap R) \cup (C \cap \bar{R})$$

$$\underset{\text{disjoints}}{=} P(C \cap R) + P(C \cap \bar{R})$$

$$\frac{P(C)}{P(C)} = 1 = \frac{P(C \cap R)}{P(C)} + \frac{P(C \cap \bar{R})}{P(C)}$$

$$= P(R|C) + P(\bar{R}|C)$$

$$1 = IC(C \rightarrow R) + IC(C \rightarrow \bar{R})$$

Indice de confiance

$$IC(C \rightarrow \bar{R}) = 1 - IC(C \rightarrow R)$$

Kmine \rightarrow association rules.

Séance 1 : 19/10/2021

→ Ret Python sont des langages du développement de modèles.

→ scikit-learn : librairie de python qui fait du data mining.

Rappel

→ matrice qui sera dans le cours :

→ $u = \begin{pmatrix} u_1 \\ \vdots \\ u_d \end{pmatrix} \in \mathbb{R}^d$; u est un vecteur ayant d composantes.

→ le vecteur transposé : $u^t = (u_1, \dots, u_d)$: vecteur ligne.

→ $v = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \in \mathbb{R}^d$

→ le produit scalaire : $\langle u, v \rangle = u^t v = (u_1, \dots, u_d) \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}$

→ pour python on peut utiliser la bibliothèque numpy pour calculer le produit scalaire de 2 vecteurs : `np.dot(u, v)`

→ il faut toujours éviter d'utiliser la boussole for. Car elle est trop gourmande en temps d'exécution.

⇒ dans google colab :

• on va créer un programme en python qui calcule la somme $u.v$ avec la boussole for et un autre qui l'utilise avec le dot.

• la vectorisation, c'est travailler avec des vecteurs beaucoup plus que avec la boussole for.

* produit scalaire avec la boussole for :

→ la bibliothèque permet de calculer le temps d'exécution du programme

→ `mp.timeit.timeit(1000000)` ⇒ fournit 1000000 mesures comprises entre 0 et 1.

→ le temps fourni par la bibliothèque est en millisecondes.
- msec

⇒ Résolution :

```
import numpy as np
```

```
import time
```

```
u = np.random.rand(1000000)
```

```
v = np.random.rand(1000000)
```

```
w = 0
```

```
d = time.time()
```

```
for i in range(1000000):
```

```
    w += u[i] * v[i]
```

```
f = time.time()
```

```
print(w)
```

```
print("boucle for : " + str(1000 * (f - d)) + " msec")
```

⇒ Résultat :

```
249708.29430390463
```

```
boucle for : 694.0708827972419 msec
```

msec □

* produit scalaire avec le dot :

```
d = time.time()
```

```
w = np.dot(u, v)
```

```
f = time.time()
```

```
print(w)
```

```
print("forme vectorisée : " + str(1000 * (f - d)) + " msec")
```

⇒ Résultat :

```
249708.29430390463
```

```
forme vectorisée : 6.673097610473633 msec
```

- NB : la forme vectorisée est 100 fois plus rapide que la boucle for pour calculer le produit scalaire.
- En Python, il est recommandé de ne pas utiliser la boucle for.

Exemple:

A : une matrice

$$A = \begin{bmatrix} a_1^t \\ a_2^t \\ a_3^t \\ a_4^t \end{bmatrix}$$

matrice

$$\begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix}$$

$$Ab = \begin{bmatrix} a_1^t b \\ a_2^t b \\ a_3^t b \\ a_4^t b \end{bmatrix} \rightarrow \text{sa revient à faire } d \text{ fois un produit scalaire.}$$

→ m dans le produit matriciel on a le choix d'utiliser le mp. c'est au la matrice $A @ b$: produit matriciel qui vecteurise automatiquement.

→ donc il ne faut pas utiliser la boucle for dans le cas où il y a la possibilité de vectoriser.

Chapitre ②: Apprentissage supervisé : 3 Modèles linéaires

→ on va voir qu'il faut que l'on ait la linéarité qu'est ce qu'on doit faire

Objectif:

• après avoir défini l'objectif et réunit les données, va à la fin du fichier de données comme

• il se présente (ensemble d'apprentissage): $D = \{(x_i, y_i)\}_{i=1}^N$; y_i c'est le label;

par exemple: x_i : les notes de l'étudiant dans un concours et y_i : c'est par exemple le salaire de l'étudiant après 5ans (on doit le prédire).

• si $y_i \in \mathbb{R}$: \Rightarrow régression

• si y_i est qualitatif \Rightarrow classification (exemple: cache-marron - employé)

• Le point de départ c'est le fichier d'apprentissage. Il faut toujours avoir un ensemble de données

→ Machine Learning (\Rightarrow Apprentissage de données)

• d'objectif: à partir de ces données, on doit trouver une fonction qui on lui donnera par la suite des x_i et elle sera capable de prédire le y_i . (supervisé)

• Pour se faire on utilise un algorithme d'apprentissage par exemple l'arbre de décision, Réservoir de modèles, analyse factorielle discriminante etc.

On suppose que l'algorithme c'est l'arbre de décision. Au début il commence avec un arbre avec 1 seul nœud, ensuite 3 nœuds, 5 nœuds ... donc il cherche l'arbre.

- h : c'est l'ensemble de tous les arbres. Il cherche un arbre parmi ces arbres appelés les hypothèses. Donc il cherche une hypothèse h qui est intéressante. Après la recherche, il s'arrête quand il trouve une hypothèse satisfaisante. On espère que l'hypothèse finale soit proche à la fonction f qu'on cherche.
- On suppose que les x sont tirés d'une loi distribution, où ils ont la loi générale : une distribution de probabilité multivariée par exemple la loi normale c'est univariée \Rightarrow elle génère un seul membre aléatoire. Mais il y a des fois qui sont multivariées. Donc on suppose que tous les dernières sont à plusieurs distributions. Ensuite, à partir de cette loi distribution on tire un autre x et on teste le h (hypothèse qu'on a choisie). $\Rightarrow h(x)$ pour déterminer ce qui prévoit la valeur de $f(x)$ approximative.

\Rightarrow C'est le principe de l'apprentissage supervisé !

- Quand on trouve le h comment on l'évalue ?

Evaluation d'un modèle supervisé :

- Si j'ai un fichier de données D : $E(h)$ c'est la fonction coût empirique associé au choix h , ou bien c'est l'ensemble des erreurs qu'on commet sur h sur un fichier de données D .

$$E(h) = \frac{1}{N} \sum_{i=1}^N L(y_i, h(x_i))$$

loss \rightarrow perte

- on calcule l'erreur, grâce à la fonction L dit fonction perte. qui est ce qu'on a perdu quand on remplace le vrai y_i par $h(x_i)$.

$$\rightarrow \text{si régression: } L(y_i, h(x_i)) = (y_i - h(x_i))^2$$

$$\rightarrow \text{si classification: } L(y_i, h(x_i)) = I(h(x_i) \neq y_i) \text{ avec: } I(p) = \begin{cases} 1, & \text{si } p \neq \text{vrai} \\ 0, & \text{sinon,} \end{cases}$$

- la différence $\uparrow \Rightarrow$ on perd plus (ce qui est bon de ce qu'on doit prévoir.)

fonction indicateur

- $E(h)$: dans les cas de classification, ça va être le pourcentage de fois qui on s'est trompé. Je compte combien de fois on s'est trompé sur le total.
 \Rightarrow taux de mauvaise classification.

2 types d'erreurs

$E_{\text{in}}(h)$: si on utilise le même fichier d'apprentissage

$E_{\text{out}}(h)$: si on utilise un autre fichier (fichier test).

Théorème de Vapnik - Chervonenkis

- Si on connaît E_{in} on peut déterminer E_{out}

$$\rightarrow \text{classification: } E_{\text{out}}(h) = E_{\text{in}}(h) + O\left(\sqrt{\frac{d \ln(N)}{N}}\right)$$

$$\rightarrow \text{régression: } E_{\text{out}}(h) = E_{\text{in}}(h) + O\left(\frac{d}{N}\right)$$

d : dimension du vecteur x_i

\rightarrow On démontre théoriquement que si les x_i sont tirés d'une distribution $P(x_i \sim P_{x_i})$

$$E_{\text{out}}(h) = E_{\text{in}}(h) + O(g(N, d))$$

avec:

$g(N, d)$:

$$\rightarrow E_{\text{out}}(h) \approx E_{\text{in}}(h)$$

• fonction décroissante % à N ($N \rightarrow +\infty$, $g(N, d) \rightarrow 0$)

• fonction croissante % à d .

⚠ plus j'ai des données, mieux c'est. (Big data)

⚠ $d \uparrow \Rightarrow$ erreur ↑ : c'est parce que il faut redire la dimension du vecteur x_i . Je faut choisir les variables les plus intéressantes pour gagner en précision.

⚠ plus la dimension du vecteur x_i augmente plus on a besoin de beaucoup de données.

exemple:

on suppose que toutes les données sont entre 0 et 1. On divise l'intervalle en 10 morceaux.

$d=1$ 

• au moins il faut une petite observation sur chaque petit intervalle

• donc il faut 10 observations

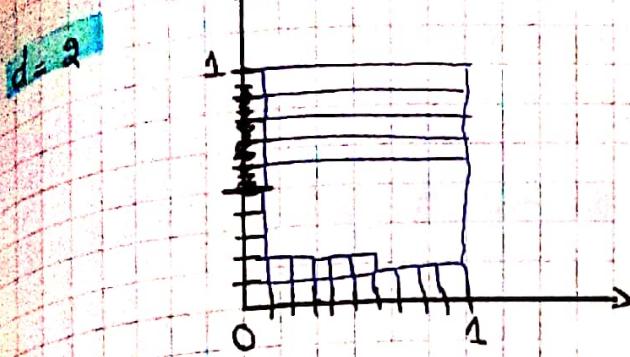
• au moins il faut une petite observation sur chaque petit intervalle

• donc il faut 10 observations

si $d=1$.

• Pour avoir un bon modèle on doit avoir suffisamment de données sur chaque petit intervalle

• intervalle



\Rightarrow donc ça fait 100 carrés.

- dans chaque carré on doit avoir 1 observation.

- donc $d=2 \Rightarrow 100$ observations pour avoir une bonne modélisation.

- $d=3 \Rightarrow 1000$ observations

\Rightarrow plus la dimension augmente, plus on a besoin de plus d'observations.

\Rightarrow c'est la malédiction de la dimension.

- si on a des vecteurs ayant plusieurs caractéristiques donc on aura besoin de bcp de données.

- on peut sauver l'erreur sur les données qui on a pas encore, à partir de l'erreur qui on a sur les données d'apprentissage. Ça dépend de la taille des données qui doit être grande et la dimension du vecteur x_i qui doit être petite pour avoir une erreur petite. et $E_{out}(h) \approx E_{in}(h)$

des modèles linéaires:

$$x_i \in \mathbb{R}^d ; x_i = (x_i^1, x_i^2, \dots, x_i^d)^t$$

transposé; car x_i est un vecteur colonne.

1, 2, 3, ..., d : indice de la composante | i : indice de l'observation.

exemple :

- on a les notes des étudiants : $x_i^1, x_i^2, \dots, x_i^d$

- chaque note on la multiplie par un poids: w_1, w_2, \dots, w_d

- donc c'est une combinaison linéaire de $x_i^1, x_i^2, \dots, x_i^d$ et lui rajoute une constante w_0 .

- donc on considère $Z_i = \sum w_j x_i^j$; i est fixe car on a pris l'observation
- on applique une fonction d'activation sur le Z : $a_i = h(z_i) = a(Z_i) = \hat{Y}_i$
- et c'est exactement le y_i qu'on prédit
- l'objectif est d'estimer (apprendre) les w_j
- ce qui différencie les modèles linéaires des autres modèles qu'on a vu c'est la fonction d'activation.

Les fonctions d'activation :

- La fonction sigmoïde $a(z) = \text{sigm}(z) = \begin{cases} +1 & \text{si } z > 0 \\ -1 & \text{si } z \leq 0 \end{cases}$

→ Perceptron Modèle

- la fonction sigmoid $a(z) = \text{sigmoid}(z) = \frac{1}{1+e^{-z}}$ [c'est presque la probabilité c'est entre 0 et 1]

→ Modèle logistique

- La fonction $a(z) = \text{id}(z) = z$

→ Modèle de Régression linéaire

linéaire : on en fait une combinaison linéaire

Perceptron Modèle :

exemple: crédit Bancaire (voir)

- le signe de w_1 est positif car plus le revenu augmente plus Z va augmenter et donc la probabilité pour prendre le crédit va augmenter.
- w_2, w_3, w_4 : doivent être positifs
- w_5 : doit être négatif ; car plus le total de dettes est grand, la probabilité de prendre un crédit va diminuer. et Z diminue.
- si le seuil on l'apprend ; seuil = w_0

- donc on définit le vecteur $w = (w_0, w_1, w_2, w_3, w_4)^t = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$
- et $x = (1, x^1, x^2, x^3, x^4)^t = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$
- on ajoute 1 car on va multiplier $w_0 \times 1$.

- on calcule $w^t x = (w_0, w_1, w_2, w_3, w_4) \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$: c'est un produit scalaire

$$w^t x = \sum_{j=0}^4 w_j x_i^j = Z$$

ensuite on applique la fonction sigm sur $h(x)$

$$h(x) = \text{sigm}(w^t x) = \begin{cases} +1 & \text{si } w^t x > 0 \rightarrow \text{accorder le crédit} \\ -1 & \text{si } w^t x < 0 \rightarrow \text{refuser} \end{cases}$$

exemple 2:

$$d=2, w = (w_0, w_1, w_2), \quad \begin{array}{l|l} x^1 : \text{revenu} \\ x^2 : \text{ancienneté de la résidence} \end{array}$$

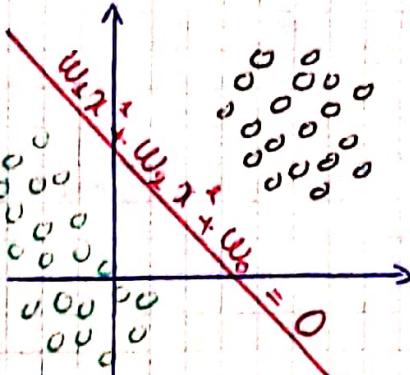
→ on construit $Z = w_0 + w_1 x^1 + w_2 x^2 \rightarrow$ fonction d'une droite

→ au dessus de la droite : accorder le crédit

→ en dessous de la droite : refuser.

⇒ on dit que les données sont linéairement séparables. Si on est en 2 dimensions

si il y a une droite qui les sépare.



→ si on est en 3 dimensions ou plus on dit qu'il y a un plan ou un hyperplan qui les sépare.

⇒ c'est ça la perception.

→ le problème de la perception c'est de déterminer les w_j : quels poids donner aux x_i^j pour que la séparation soit bonne.

Algorithme de perception:

→ comment il calcule le w (vecteur).

→ $D = \{(x_i, y_i)\}_{i=1}^N$; on suppose que les données sont linéairement séparables (condition importante) cād les 2 groupes ne sont pas mélangés.

→ à l'itération 0, on initialise w à 0, $w \in \mathbb{R}^{d+1}$ car on ajoute le w_0 .

→ c'est un algorithme itératif et w change au cours des itérations.

→ on prend de D un exemplaire (x_m, y_m) mal classé cād $y_m(w^t x_m) \leq 0$ mais le vrai y_m est > 0

$$\left. \begin{array}{l} w^t x_m \leq 0 \\ y_m > 0 \end{array} \right\} \Rightarrow y_m (w^t x_m) \leq 0 : \text{un exemplaire mal classé}$$

produit scalaire

→ si on a un exemplaire mal classé, on met à jour le w : $w \leftarrow w + x_m y_m$

→ x_m, y_m vecteur de même dimension que W .

→ jusqu'à ce que tous les éléments soient bien classés, c'est-à-dire que les classes sont séparées et renvoyer $h(x) = \text{sign}(W^t x)$ et au final l'algorithme peut faire la prédiction.

→ le dernier W trouve comme vecteur W^*

→ si W^* est le vecteur final alors $y_m (W^t x_m) \geq 0 \forall m$. c'est-à-dire aucun élément n'est mal classé.

→ l'algorithme ne converge pas dans le cas où les classes ne sont pas linéairement séparables.

→ Cet algorithme suppose qu'il y a un plan qui sépare les positifs des négatifs.

Pourquoi l'algorithme marche-t-il?

Justification intuitive:

si (x_m, y_m) est mal classé $\Rightarrow y_m (W^t x_m) < 0$

produit scalaire $W^t = (w_0, \dots, w_N)$ et $x_m = \begin{pmatrix} 1 \\ x_m^1 \\ x_m^2 \\ \vdots \\ x_m^N \end{pmatrix}$

$\Rightarrow W_{m+1} = W_m + x_m y_m$ on met à jour le W

$\Rightarrow W_{m+1}^t x_m = (W_m + x_m y_m)^t x_m$ on teste si l'élément devient mieux classé à l'itération $m+1$

$$\Rightarrow W_{m+1}^t x_m = W_m^t x_m + y_m x_m^t x_m$$

$$\Rightarrow y_m W_{m+1}^t x_m = y_m W_m^t x_m + y_m^2 x_m^t x_m$$
 on multiplie les 2 termes par y_m

$$\Rightarrow y_m W_{m+1}^t x_m = y_m W_m^t x_m + y_m^2 \|x_m\|^2$$

$$\Rightarrow y_m W_{m+1}^t x_m = y_m W_m^t x_m + \|x_m\|^2$$

mieux classé

mal classé

\Rightarrow le classement s'améliore car on ajoute un terme positif $\|x_m\|^2$.

→ mais on a pas encore montré que l'algorithme converge.

exemple:

$x_1 \quad x_2 \quad y = x_1 \text{ ou } x_2 \Rightarrow$ si on représente graphiquement les

0 0

0

0 1

1

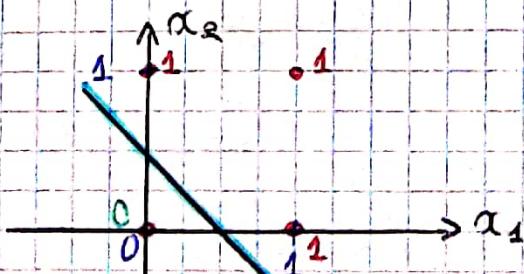
1 0

1

1 1

1

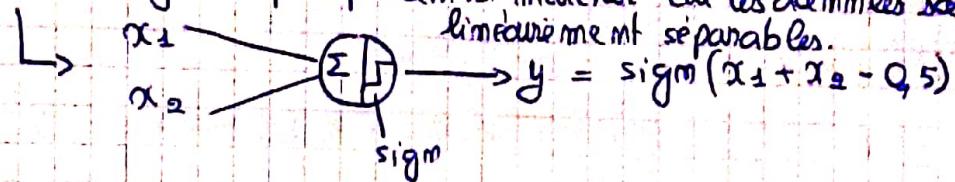
Valeurs on aura :



→ est ce que les données sont linéairement séparables dans ce cas? (les 0 avec les 1) \Rightarrow oui, car il y a une droite qui sépare les 0 des 1.

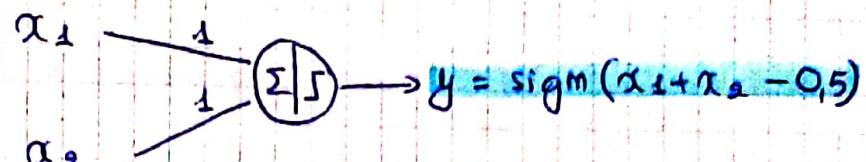
\Rightarrow dans ce cas l'algorithme de perceptron va marcher car les données sont linéairement séparables.

exemple:



poids du $w_0 = 0$

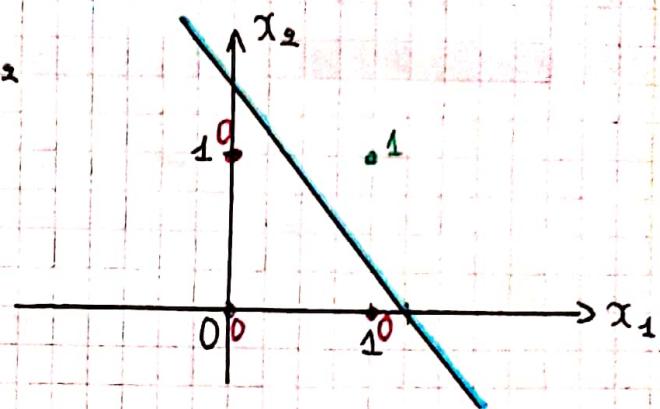
$$\begin{aligned} w_1 &= 1 \\ w_2 &= 1 \end{aligned}$$



exemple:

x_1	x_2
0	0
0	1
1	0
1	1

$$y = x_1 \text{ and } x_2$$



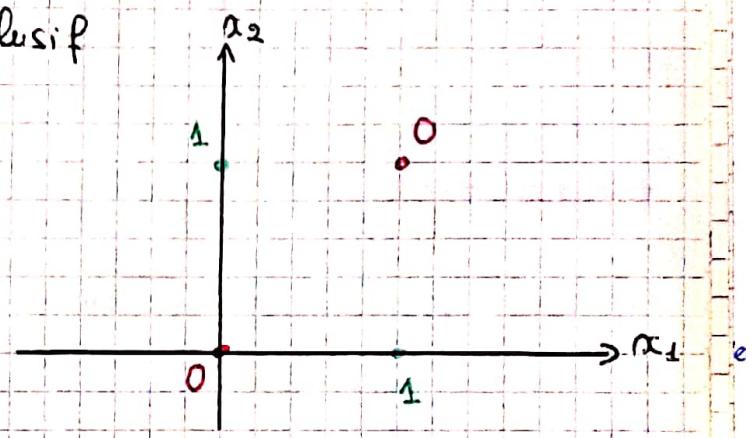
\Rightarrow dans ce cas les données sont linéairement séparables.

L'équation de la droite: $x_1 + x_2 - 1,5 = 0$

exemple:

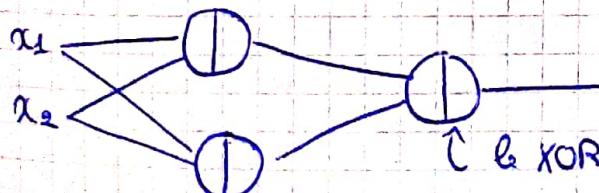
x_1	x_2	$y = x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0

↓ en exclusif



\Rightarrow dans ce cas les données ne sont pas linéairement séparables car y a aucune droite qui sépare les 0 des 1. Dans ce cas l'algorithme converge pas.

→ On peut le faire avec 2 neurones.



⚠ le perceptron c'est un seul neurone.

→ ça met à jour le w que lorsque un élément est mal classé.

Exercice p 22

① Quelle est la différence entre l'algorithme de l'ergo et l'algorithme que tu as vu?

* le premier algorithme ⇒ prend uniquement les points mal classés.

* l'algorithme de l'ergo ⇒ prend tous les points, donc il peut trop plus de temps.
il prend chaque point, ensuite il teste s'il est mal classé.

→ en Python et R : il y a une instruction qui permet de choisir ceux qui sont positifs ou négatifs ($y(u^T x)$)

↳ ça a un vecteur x : comment choisir les composantes de x positifs.

↳ L'instruction : $x = x[x > 0]$

$$\text{ou } \cancel{x} \quad z = x[x > 0]$$

met dans z les composantes de x positifs

Simulation : (voir) Algorithme de perceptron | p 23

→ données linéairement séparables ⇒ il trouve la droite

→ données non séparables ⇒ il ne trouve jamais la droite.
non linéairement

TP, mini-project ⚡ Programme permettant d'apprendre les w d'un perceptron

Page 35 : Python

→ ça a l'implémentation de l'algorithme de perceptron.

* Apprentissage :

→ c'est une programmation orientée objet : il faut définir une classe Perceptron
contenant un certain nb de méthodes.

• la 1^{ère} méthode : fit (ajuster) : ça te donne les X, y et tu dois nous donner le w . On lui donne une matrice X contenant tout le fichier de données, X contient les x_i , vecteur ~~longue~~ colonne. ensuite on la transpose (Python ⇒ transposé de la matrice A c'est $B = A.T$)

→ m_samples = $X.shape[0]$: nombre d'échantillons c'est le
nombre de lignes

→ m -features = X .shape[1] : membre du cam paramètres c'est le nb de colonnes
→ ensuite on ajoute la colonne de 1 \hookrightarrow de variables

self-weights = np.zeros((m-features + 1,))

→ contient m lignes et $d+1$ colonnes (features)

→ on a 2 bouches

→ au lieu d'utiliser la 2^{ème} bouché on peut utiliser " $z = x[\alpha > 0]$ "
→ si le produit scalaire < 0 \Rightarrow mise à zéro du produit

→ l'algorithme jusqu'à ce qu'il retourne b & w

* Fonction de Prediction

→ prend un point et prend son produit scalaire et teste si c'est la m^{me} valeur que y

→ retourne uniquement la valeur de y

* Fonction d'évaluation

→ calcule combien de points sont mal classés.

→ pour créer les données : on va générer des nb aléatoires, pour se faire on utilise la fonction make_classification de python

* m -features = 2 . nb de données

* m -samples = 200 : nb d'observations

* le reste ne change pas.

→ on appelle la fonction pour faire le modèle, la fonction prediction pour prédire et scorer pour évaluer le modèle.

⇒ il faut implementer le programme et l'améliorer.

→ il faut ajouter des librairies pour utiliser quelques fonctions.

d'erreur

• pour mesurer l'erreur sur un fichier d'apprentissage on utilise la fonction qu'on

a déjà vu : si on utilise de nouvelles données (on génère d'autres données)

et on mesure le taux d'erreur. $\hat{E}_{out}(h) = \frac{1}{M} \sum_{j=1}^M I(h(\hat{x}_j) \neq \hat{y}_j)$

Qui montre que l'erreur sur un fichier test:

l'espérance de

$$E(I(h(x) \neq y)) = P(h(x) \neq y) = E_{out}(h)$$

Preuve

V.a $X = \begin{cases} 1 & p \\ 0 & 1-p \end{cases}$

$$\Rightarrow E(X) = 1 \times p + 0 \times (1-p) = p$$

l'espérance d'une var binomiale $E(X) = p$

$$E(X) = P(X=1) \text{ si } X \sim \text{Binomial}(p)$$

paramètre

$$\Rightarrow E_{out} = E(I(h(x_i) \neq y_i))$$

$$\hookrightarrow \begin{cases} 1 & \text{si } \neq \\ 0 & \text{sinon} \end{cases}$$

dans l'espérance de I = probabilité de ne pas avoir une bonne prédiction

Voir la preuve de l'algorithme converge.

→ si les données ne sont pas linéairement séparables on utilise l'algorithme de Perceptron.

Modèle de Perceptron

⇒ Perceptron : cas de la non séparabilité linéaire (voir)

→ on initialise W et W_0 à 0 $W = W_0 = 0$

→ si un élément est mal classé on fait la mise à jour dans W_0

→ si on remarque que $E_{in}(W_0)$ diminue, on remplace W par W_0 .

→ au début, il faut fixer un nb d'itérations sinon on ne l'arrêtera jamais
car tous les éléments ne sont pas séparés linéairement.

différence entre PLA (Perceptron Linéairement Alg) et Perceptron :

→ E_{in} augmente et diminue de façon alternée (PLA)

→ E_{in} ne fait que diminuer (Perceptron)

et la fin elle devient constante.

on va mettre à jour la w que si le nouveau E_{in} est meilleur que l'ancien E_{in} .

→ importer cet algo de Perceptron

l'algorithme retourne la meilleure droite. Il y a un tour d'erreur quand même.

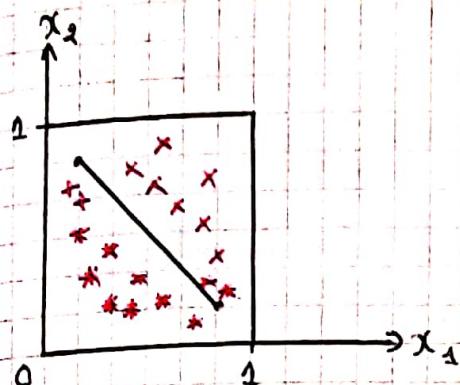
Par contre le PLA ne permet pas que si les données sont linéairement séparables.

→ Installer la R studio.

Programme R :

on a 2 var x_1 et x_2

on crée une droite qlq dans le carré $\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$. On lui demande de me donner 2 points au hasard pour construire la droite



ensuite on lui demande de me donner 200 points au hasard et on les place.

on a créé 200 pts linéairement séparables

ensuite, on déclenche l'algorithme de Perceptron jusqu'à ce qu'il nous donne la droite finale et on la compare avec la droite réelle qui on a créé au début.

Voir le programme (capture)

2 valeurs plutôt

$x = runif(2, -1, 1)$ # random uniforme \Rightarrow 2 points entre -1 et 1

$w = c(1, -w_1, -w_0)$: on créé le vecteur w .

l'erreur diminue qd le nb d'observations augmente.

Modèle de la régression linéaire :

on a un fichier de données $D = \{(x_i, y_i)\}_{i=1}^N$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

la fonction d'activation = la combinaison linéaire = sortie

exemple.

→ on va prédire le montant de crédit au de lieu de prédire si la demande est acceptée ou rejetée.

→ si on veut prédire la récolte du blé pour le Maroc en fonction de la pluie on utilise la régression.

→ prédire le taux de sucre dans notre corps en fonction d'un certain nb de test médicaux c'est de la régression.

→ Chaque fois que $y \in \mathbb{R}$ c'est de la régression.

→ Comment on écrit le modèle?

* on veut apprendre $h(x) = \hat{y}$: valeur approximative de y .

* $y_i = h(x_i) + \varepsilon_i$ # la vrai y c'est h + une erreur.

$$y_i = w^T x_i + \varepsilon_i$$

$$y_i = \hat{y}_i + \varepsilon_i$$

→ le seul inconnu c'est le vecteur w .

dès lors $E_{\text{im}}(w) = E_{\text{im}}(h) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$ fonction de perte

⇒ Méthode des moindres carrés

→ c'est la moyenne des différences au carré.

→ on cherche le w^* qui minimise $E_{\text{im}}(w)$:

dès lors $w^* = \arg \min_w E_{\text{im}}(w)$ [quel est le w qui rend $E_{\text{im}}(w)$ mini]

* E_{im} du Perceptron calcule combien de fois on a fait l'erreur (Probabilité)

Perceptron : $E_{\text{im}}(h) = \frac{1}{N} \sum_{i=1}^N I(h(x_i) \neq y_i) \rightarrow L(h(x_i), y_i)$

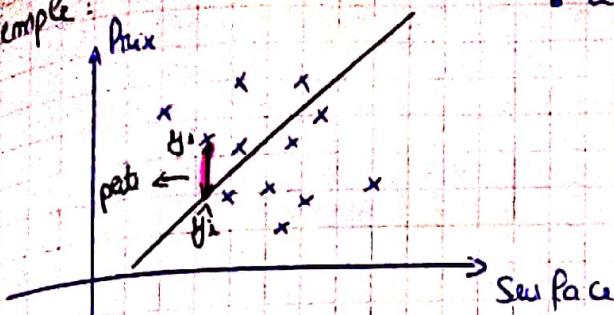
$\underbrace{\hspace{10em}}_{\text{Nombre d'erreurs}}$

$\underbrace{\hspace{10em}}_{\text{Pourcentage d'erreurs}}$

Régression : $E_{\text{im}}(h) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$

||
h(x_i)
||
w^T x_i

exemple :



on apprend une droite

$\rightarrow E_{im}$: erreur sur le fichier d'apprentissage, on cherche à la diminuer.

\rightarrow on peut avoir la régression sur 1 seul variable ou sur plusieurs variables.

\rightarrow par exemple on a plusieurs points dans l'espace et on a plusieurs variables

x_1, x_2, \dots et on essaie de tracer une fonction $y = f(x_1, x_2)$ on peut être une surface, un plan ...

\rightarrow Trouver le W qui rend $E_{im}(W)$ minimal.

Les équations $D = \{(x_i, y_i)\}_{i=1}^N$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

$w^t x_i = x_i^t w$ produit scalaire

$x_i = (1, x_i^1, x_i^2, \dots, x_i^d)^t$ vecteur colonne $(d+1) \times 1$

$x_i^t = (1, x_i^1, x_i^2, \dots, x_i^d)$ vecteur ligne $1 \times (d+1)$

$$X = \begin{pmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_N^t \end{pmatrix}, \quad W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix}; \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

← les vraies valeurs de y

matrice X
N lignes
 $d+1$ colonnes

$d+1$ car pour chaque ligne on ajoute le $\frac{1}{2}$

$$\Rightarrow X = \begin{pmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & x_N^2 & \cdots & x_N^d \end{pmatrix}$$

!

N lignes / $d+1$ colonnes

N : nombre d'échantillons, d'observations

d : nombre de variables

→ chaque x_i possède un y_i

\downarrow

$\in \mathbb{R}^d$

$$Xw = \begin{pmatrix} x_1^t w \\ x_2^t w \\ \vdots \\ x_N^t w \end{pmatrix} ; \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$\sum (y_i - w^t x_i)^2 = \sum (y_i - x_i^t w)^2$$

= la somme des lignes \uparrow de la matrice suivant

$$Xw - y = \begin{pmatrix} x_1^t w - y_1 \\ x_2^t w - y_2 \\ \vdots \\ x_N^t w - y_N \end{pmatrix}$$

$$\Rightarrow \text{c'est } \|Xw - y\|^2$$

$$\Rightarrow E_{\text{im}}(w) = \frac{\|Xw - y\|^2}{N}$$

on cherche à minimiser $\|Xw - y\|^2$

on sait que $\|a\|^2 = a^t a = \sum a_i^2$: somme des composantes au carré

$$\begin{aligned} \Rightarrow \|Xw - y\|^2 &= (Xw - y)^t (Xw - y) \\ &= ((Xw)^t - y^t)(Xw - y) \\ &= (w^t X^t - y^t)(Xw - y) \\ &= w^t X^t Xw - w^t X^t y - y^t Xw + y^t y \\ &= w^t X^t Xw - \cancel{w^t X^t y} - \cancel{y^t Xw} + y^t y \\ &= w^t X^t Xw - 2w^t X^t y + y^t y \end{aligned}$$

→ il faut trouver le w qui rend cette quantité le plus petite possible.

→ donc on doit la dériver % à w et annuler la dérivée.

→ w est un vecteur donc on doit connaître la dérivée % à un vecteur.

le gradient : $\nabla_x(x^t a) = a$

$$\nabla_w(w^t x) = x$$

$$\text{dans } \nabla_w (w^t X^t Y) = X^t Y$$

$$\frac{\partial U(x)}{\partial x} = \nabla_x (u(x)) = \nabla_x (x^t A x) = 2 A x \quad \text{avec } u(x) = x^t A x$$

$$\text{dans } \nabla_w (w^t X^t X w) = 2 X^t X w$$

$$\nabla_w (Y^t Y) = 0$$

$$\text{dans } \nabla_w E_{\text{lin}} = \frac{1}{N} (2 X^t X w - 2 X^t Y) = 0 \quad (\text{minimum})$$

$$\Rightarrow X^t X w^* = X^t Y$$

$$\text{et } \frac{d \nabla_w E_{\text{lin}}}{d w} = X^t X \quad (\text{matrice définie positive})$$

↓
dans c'est un minimum

$$w^* = \underset{w}{\operatorname{argmin}} E_{\text{lin}}(w)$$

$$\Rightarrow X^t X w^* = X^t Y \quad \text{Il faut résoudre ce système}$$

$$\Rightarrow w^* = (X^t X)^{-1} X^t Y \quad \text{!} \leftarrow \text{déconseillé d'utiliser l'inverse.}$$

$$\Rightarrow \begin{bmatrix} X & (N \times (d+1)) \\ X^t & ((d+1) \times N) \end{bmatrix} \Rightarrow X^t X \quad ((d+1) \times (d+1)) \text{ hennement}$$

car N peut être assez grande (Big data)

dim de la matrice ↴ c'est le nb d'observations.

$$\Rightarrow X^t Y \quad ((d+1) \times 1)$$

$$\Rightarrow h(x) = w^* x = \left[(X^t X)^{-1} X^t Y \right] x$$

on utilise le w qui a été trouvé pour prédire le y .

Résumé :

Perceptron : suppose les données linéairement séparables.
 ↳ pour la classification

Paquet : $w \leftarrow w + y_i x_i$ si (x_i, y_i) mal classé.

La régression : $w^* = (X^t X)^{-1} X^t Y$

séance 5 : 03 - 05 - 2021

- $D = \{(x_i, y_i)\}_{i=1}^N$: Pichier de données.
 - $x_i \in \mathbb{R}^d$: contient d'amples.
 - On ajoute un biais par exemple si c'était une droite on ajoute le coeff b pour faire déplacer la droite.
 - on débat on calcule la combinaison linéaire $z_i = \sum w_j x_{ij}$ ensuite on applique une fonction d'activation.
 - pour le Perceptron: la fonction d'activation c'est la fonction sigm(z)
 - détermine si le client va rembourser le crédit ou non.
 - pour la régression linéaire: c'est la fonction identité id(z)
 - pour le modèle logistique: c'est la fonction sigmoid(z).
 - si $z \rightarrow +\infty \Rightarrow a(z) = 1$ si $z = 0 \Rightarrow a(z) = 1/2$
 - si $z \rightarrow -\infty \Rightarrow a(z) = 0$ si $z \uparrow \Rightarrow$ plus de chance pour rembourser le crédit.
 - ⇒ on transforme la valeur z (le score calculé pour chaque client) en une probabilité pour qu'il rembourse le crédit.
- ⇒
- sigm(z)**: calcule est ce qu'il va rembourser le crédit (oui/non)
 - id(z)**: combien on va lui donner de crédit.
 - sigmoid(z)**: calcule la probabilité pour que le client rembourse le crédit.

exemple : Appréciation de crédit bancaire :

x^1 : Revenu

x^2 : Âge/menage de la résidence

x^3 : Propriétaire / locataire (0/1)

x^4 : Total des dettes

⇒ on calcule une moyenne pondérée par w :

$$z = w_1 x^1 + w_2 x^2 + w_3 x^3 + w_4 x^4$$

→ si on utilise la fonction **sigm(z)**

$$w_1 x^1 + w_2 x^2 + w_3 x^3 + w_4 x^4 - seuil > 0 \rightarrow accorder le crédit$$

$$w_1 x^1 + w_2 x^2 + w_3 x^3 + w_4 x^4 - seuil \leq 0 \rightarrow refuser$$

- si on utilise la fonction $\text{id}(z)$.
- $z \geq 1 \Rightarrow$ le montant du crédit sera grand.
- on calcule avec z la somme du crédit qu'on donnera au client.
- si on utilise la fonction $\text{sigmoid}(z)$.
- on transforme z en une probabilité de rembourser le crédit.
- * Même pour la régression il y a un critère à minimiser c'est le coût.

Régression $\rightarrow \text{coût}(w) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$

y_i estimé
 \hat{y}_i
 $w^T x_i$

$y_i - \hat{y}_i$: distance entre la réalité et la prédiction

$(y_i - \hat{y}_i)^2$: Loss (w) pour 1 seule observation

on cherche à minimiser le coût = trouver w qui rend le coût minimal.

\Rightarrow Toutes les méthodes (Algorithmes) du Machine Learning définissent un

coût à minimiser. Y compris l'algorithme de Perceptron (il a un coût)

\rightarrow Pour la régression on a défini le coût, on la définit pour trouver la solution: $w^* = (X^T X)^{-1} X^T Y$

\rightarrow Le Perceptron a un coût, le Réseau de neurones a un coût...
qui on cherche à minimiser.

\Rightarrow On cherche toujours à minimiser le: $E_{\text{in}}(w)$: la w qu'on cherche doit minimiser l'erreur sur le fichier d'apprentissage.

\Rightarrow donc il faut définir un algorithme de minimisation :

la descente du gradient : algèbre très connue de minimisation.

Algorithme de la descente du gradient (Minimisation d'une fonction $E_{\text{in}}(w)$):

• on suppose qu'on veut minimiser une fonction $E_{\text{in}}(w)$ et on suppose que

w est un scalaire. $E_{\text{in}}(w)$: fonction réelle.

• on suppose que la fonction a un seul minimum \Rightarrow

• on cherche le minimum

* * • on choisit un w_0 au hasard : point

de départ.

→ Toutes les méthodes du machine

Learning utilisent un algorithme

itératif dans la descente du

gradient. (Sauf la régression linéaire).

(Pour le Perceptron c'était itératif).

→ Pour la régression on peut utiliser la méthode de descente de gradient.

• ensuite on calcule la dérivée en w_0 (la tangente). $\frac{\partial E_{\text{lin}}(w)}{\partial w}$ en w_0

$$\Rightarrow \text{le gradient de } E_{\text{lin}} \text{ en } w_0 : \nabla_w E_{\text{lin}}(w_0) = \frac{\partial E_{\text{lin}}(w_0)}{\partial w}$$

le gradient ici va être positif car la fonction est croissante

• on met à jour le w : $w_1 = w_0 - \alpha \nabla_w E_{\text{lin}}(w_0)$, α : très petit
cad : on prend un morceau de gradient et plus que le gradient > 0 donc

$w_1 < w_0$ parce qu'on fait baiger le w_0

• de la même façon pour w_1 : on calcule la pente ensuite on calcule w_2 :

$$w_2 = w_1 - \alpha \nabla_w E_{\text{lin}}(w_1)$$

• on continue comme ça jusqu'à ce que le gradient est nul. Et donc on est arrivé au minimum.

* * • on suppose qu'on a commencé de très gauche.

• on choisit un w_0 à gauche au hasard

• on calcule le gradient :

il est négatif

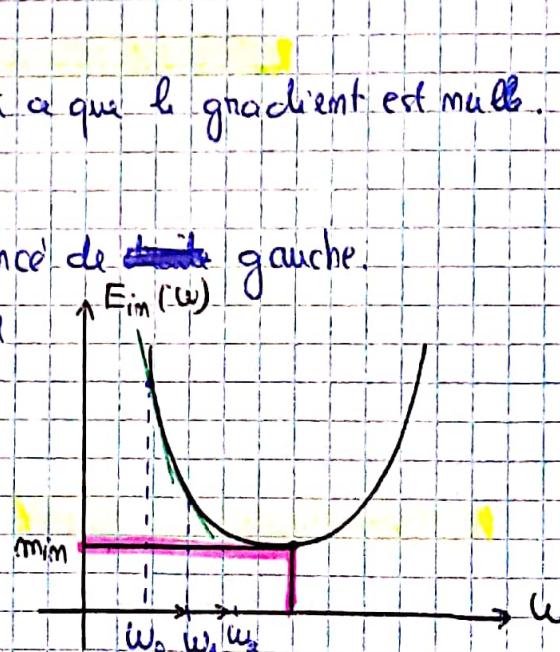
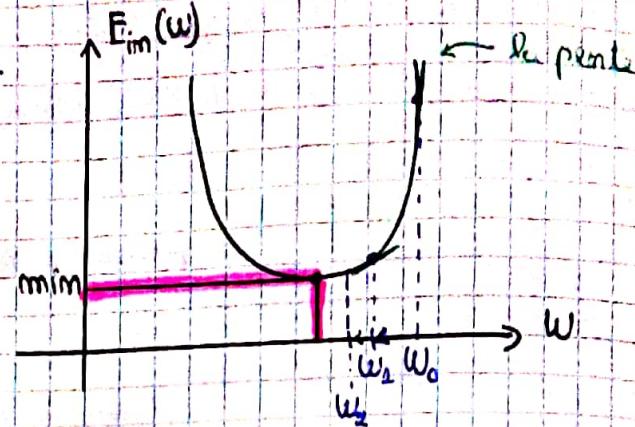
$$w_1 = w_0 - \alpha \nabla_w E_{\text{lin}}(w_0)$$

→ on enlève qd chose de négatif

\Rightarrow cad on ajoute un terme positif à w_0

• jusqu'à ce que le gradient soit nul \Rightarrow on retrouve donc le minimum.

\Rightarrow c'est ça l'algorithme de descente du gradient dans le cas univarié.



→ dans le cas général : on calcule toujours le gradient de E_{im} par rapport à w

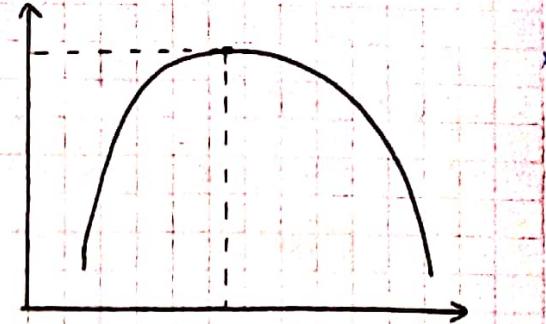
$\nabla_w E_{\text{im}}(w)$ et plus on va aller dans le sens contraire du gradient. ↗ (-)

* si par exemple on a une fonction à maximiser on va faire :

$$w_{t+1} = w_t + \alpha \nabla f(w_t)$$

gradient de la fonction à maximiser

→ on doit aller dans le sens du gradient (+)



NB

choix de α :

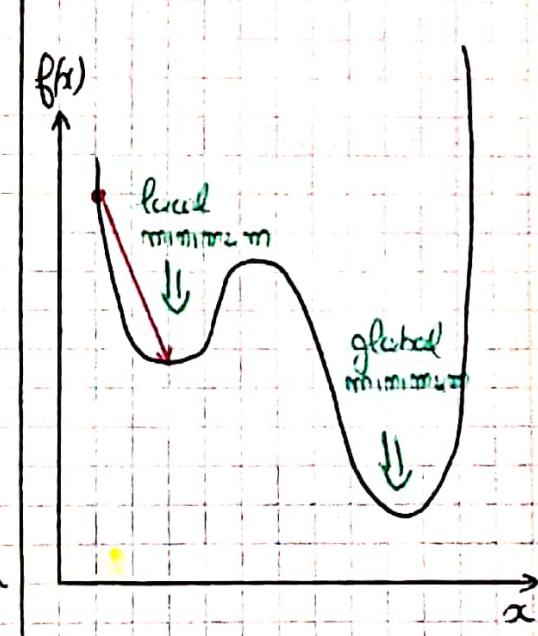
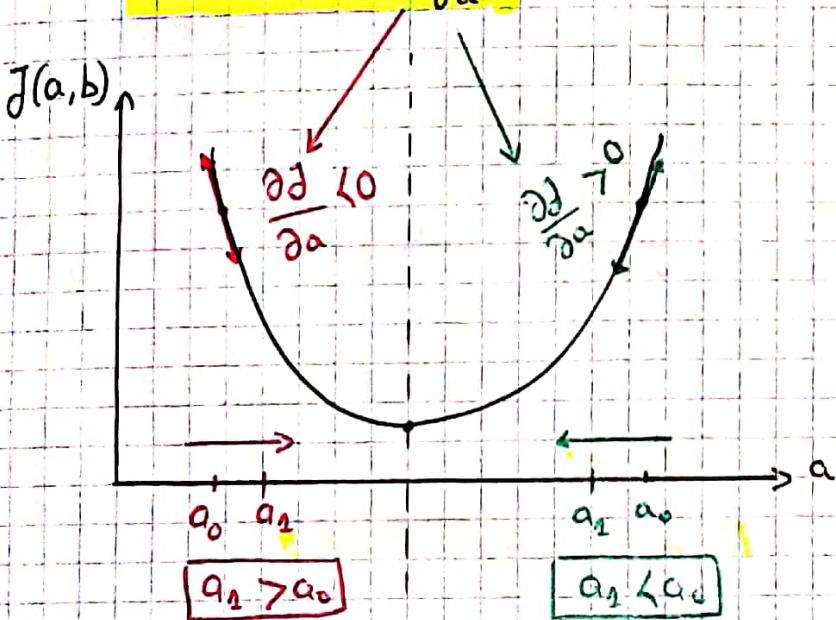
- on prend une entrée très petite par exemple 0.1. On essaie avec plusieurs α .
- on peut choisir α grande et la faire diminuer...
- on bien choisir α constante et petite.

NB

si on a 2 minima on va prendre le 1^{er} qu'on va rencontrer.

Résumé :

$$\alpha_1 = \alpha_0 - \alpha \frac{\partial J}{\partial a}$$



il faut prendre plusieurs points de départ au hasard et à chaque on va trouver un minimum et ensuite on prendra le minimum des minimums.

NB (Voir)

le cas de 2 paramètres θ_1 et θ_0 donc on aura sur face. Le coût $J(\theta)$ est en fonction de θ_0 et θ_1 . L'algorithme de descente de gradient prend toujours un point de départ et commence à descendre jusqu'à trouver un minimum local.

- Si on a à faire plusieurs minimums et à la fin choisir le minimum de tous les minimums.

Cours :

- on cherche à $w^* = \operatorname{argmin}_w E_{\text{im}}(w)$

• la tangente en $w_0 \Rightarrow \nabla_w E_{\text{im}}(w_0)$

$$w_1 = w_0 - \alpha \nabla_w E_{\text{im}}(w_0)$$

$$w_2 = w_1 - \alpha \nabla_w E_{\text{im}}(w_1)$$

etc

α : the learning rate

Algorithmes: p35

- cas multivarié: c'est la même formule. $w(t+1) = w(t) - \alpha \nabla_w E_{\text{im}}(w(t))$ *

- on calcule $w(t+1)$: c'est w à l'itération $t+1$

Justification: Pourquoi l'algorithme de gradient marche?

$$E_{\text{im}}(w_0 + \beta \vec{v}) = E_{\text{im}}(w_0) + \underbrace{\beta \vec{v}^T \nabla_w E_{\text{im}}(w_0)}_{\text{minimale si } \vec{v} = -\frac{\nabla_w E_{\text{im}}(w_0)}{\| \nabla_w E_{\text{im}}(w_0) \|}} + O(\beta^2)$$

vecteur départ

→ on suppose qu'on est dans w_0 et on veut aller dans une direction \vec{v} (inconnue)

→ on prend un morceau de la direction \vec{v}

→ on fait un DL de E_{im} à w_0

pour que E_{im} diminue le plus grandement possible

$$\begin{aligned} (\vec{v})^T \nabla_w E_{\text{im}}(w_0) &= \|\vec{v}\| \|\nabla_w E_{\text{im}}(w_0)\| \cos(\theta) \\ &= \|\nabla_w E_{\text{im}}(w_0)\| \cos(\theta) \end{aligned}$$

on prend $\vec{v} = \|\nabla_w E_{\text{im}}(w_0)\| \vec{u}$

$$w_1 = w_0 - \alpha \nabla_w E_{\text{im}}(w_0)$$

- on cherche un vecteur \vec{v} qui permet de diminuer $E_{\text{im}}(w_0 + \beta \vec{v})$.

⇒ c'est la méthode du gradient

→ si α très petit : la convergence va être très lente.

→ si α très grand: on risque de diverger.

⇒ prendre un α raisonnable !

Algorithme de gradient ⇒ Formule: $w(t+1) = w(t) - \alpha \nabla_w E_{\text{im}}(w(t))$

- Comment utiliser l'algorithme de descente de gradient ? Quand est ce qu'on s'arrête ?
- c'est un algorithme itératif et on cherche le ω minimisant la E_{im} .
- Plusieurs critères d'arrêt :

- ① se fixer un nb maximal d'itération n .
- ② Arrêter si $\|\nabla_{\omega} E_{\text{im}}(\omega)\| \leq \epsilon$. (on cherche à rentrer dans un plateau)
- ③ Arrêter si $E_{\text{im}}(\omega) < \epsilon'$.

On combine ① avec ② ou ① avec ③

Ex 1. Régression linéaire

Mq : $\nabla_{\omega} E_{\text{im}}(\omega) = \frac{2}{N} \sum_{i=1}^N - (y_i - \omega^t x_i) x_i = \frac{4}{N} (-2X^t y + 2X^t \omega)$

$$X^t y = \begin{bmatrix} x_1^t \\ x_2^t \\ \vdots \\ x_N^t \end{bmatrix}$$

$$\text{donc } X^t y = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1, x_2, \dots, x_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$$X^t y = \sum_{i=1}^N x_i y_i$$

→ pour la régression linéaire : on peut utiliser la méthode de gradient au lieu de dériver et trouver le minimum directement.

⇒ Tirer ω autrement.

Tant que $\|\nabla_{\omega} E_{\text{im}}(\omega)\| > \epsilon$
 $\omega \leftarrow \omega - \alpha \nabla_{\omega} E_{\text{im}}(\omega)$

Retour ω

pour le gradient on utilise l'une des formules ↑

→ $\frac{2}{N} \sum_{i=1}^N - (y_i - \omega^t x_i) x_i$: cette formule contient la somme de toutes les observations. A chaque itération il faut calculer cette somme. ☺

Si on a 1 000 000 itérations : il faut à chaque itération calculer la somme donc ça fait perdre du temps.

⇒ donc on définit :

- **descente du gradient batch**: utiliser toutes les données à chaque itération du gradient. (toutes les observations)
- **descente du gradient mini-batch**: utiliser une partie de données à chaque itération du gradient. (une partie des observations)
- **descente du gradient stochastique**: utiliser une seule observation à chaque itération tirée au hasard. (1 seul observation)

Exemple 1: Régression linéaire

Stochastique: $\nabla_w E_{\text{lin}}(w) = -2(y_i - w^T x_i)x_i$

Algorithmme: Régression (descente du gradient stochastique).

Tirer w aléatoirement

```

Répéter
  { Tirer un  $(x_i, y_i)$  de D
    Calculer  $\nabla_w E_{\text{lin}}(w) = -2(y_i - w^T x_i)x_i$ 
     $w \leftarrow w + 2\alpha(y_i - w^T x_i)x_i$ 
  } Jusqu'à  $|\nabla_w E_{\text{lin}}(w)| < \epsilon$ 
Fim
Fim
  
```

→ On remarque que ça va donner la même solution que la méthode itérative.

* minimiser → descente du gradient.

* maximiser → montée du gradient.

→ On applique la méthode du gradient au Perceptron \Rightarrow ça va donner le même algorithme.

Exemple 2: Le perceptron (descente du gradient stochastique)

→ On définit la fonction perte (loss) sur 1 seul observation:

$$* L(y_i, h(x_i)) = \max(0, -y_i w^T x_i)$$

→ donc $y_i w^T x_i > 0$
d'où $-y_i w^T x_i < 0$

$$* L(y_i, w^T x_i) = \begin{cases} 0, & \text{si } (x_i, y_i) \text{ bien classé} \\ -y_i w^T x_i, & \text{sinon} \end{cases}$$

→ $-y_i w^T x_i > 0$

$$* \nabla_w L(y_i, w^T x_i) = \begin{cases} 0, & \text{si } (x_i, y_i) \text{ bien classé} \\ -x_i y_i, & \text{si } (x_i, y_i) \text{ mal classé} \end{cases}$$

Algorithmme:

Tirer w aléatoirement au 0

→ le perceptron c'est un cas

Répéter

Tirer un (x_i, y_i) mal classé

particulier du ↪

$$\nabla_w E_{\text{lin}}(w) = -x_i y_i$$

$$w \leftarrow w + y_i x_i$$

« gradient descent »

ça marche pour

$$\alpha = 1$$

①

- En général, si on ne sait pas si une fonction est convexe, en particulier dans les réseaux de neurones on ne sait pas si la fonction est convexe ou pas. Donc on fixe un nb maximum d'itérations (ex → TP)
- Par contre pour le Perceptron, la régression ⇒ la fonction est convexe. Donc on sait qu'on va le trouver. Donc on peut ne pas utiliser la branche far, car il y a un nb d'itérations fixe car on sait qu'on va converger pour la Régression.

Modèle de la régression logistique:

→ modèle simple et assez.

exemple: * probabilité qu'un client rembourse le crédit.

* probabilité qu'une transaction soit malhonnête.

exemple:

$$x = (1, \alpha^1, \alpha^2, \alpha^3, \alpha^4)^t$$

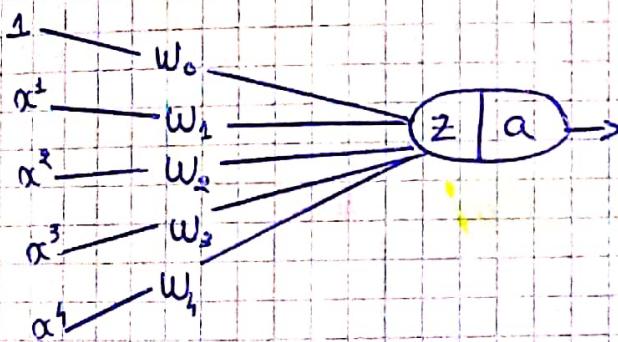
α^1 : niveau cholestérol.

α^2 : âge

α^3 : pression sanguine

α^4 : poids

=> on veut prévoir la probabilité d'avoir une attaque cardiaque?



$$\hat{y} = P(y = +1 | x, w)$$

Probabilité conditionnelle

$$Z = w^t x ; \quad a(Z) = \frac{1}{1 + e^{-Z}} = \text{sigmoid}(Z)$$

→ Plus Z est grande plus la probabilité d'avoir une attaque cardiaque est grande.

→ L'inconnue c'est les W (les poids). On cherche les W qu'il faut utiliser.

→ Pour le Perceptron : on a plusieurs mises en œuvre.

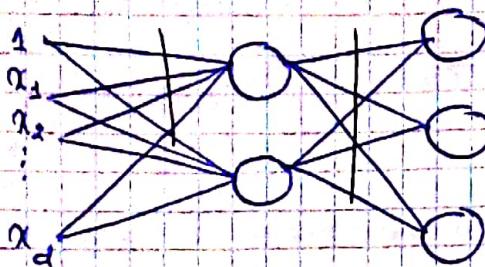
Par exemple : 2 mises en œuvre :



on a 2 vecteurs.
dans le nb de composantes
c'est : 2x(d+1)

→ donc il faut estimer $2x(d+1)$ variables.

→ on peut avoir une autre couche de mélange:



→ donc on a un autre vecteur à calculer

→ plus j'ai de couches plus on a des incarramens

→ on doit estimer tous les paramètres

7 composantes à calculer ??

NB estimer → statistiques | apprendre → machine learning

Comment apprendre les w ?

on a $Z = w^T x$ sachant qu'on connaît w et x

$$P(y = +1 | x, w) = \frac{1}{1 + e^{-Z}} = \frac{1}{1 + e^{-w^T x}}$$

avec une attaque cardiaque

$$\begin{aligned} P(y = -1 | x, w) &= 1 - \frac{1}{1 + e^{-Z}} = 1 - \frac{1}{1 + e^{-w^T x}} \\ &= 1 - P(y = +1 | x, w) \\ &= \frac{e^{-w^T x}}{1 + e^{-w^T x}} = \frac{1}{1 + e^{w^T x}} \end{aligned}$$

$$\Rightarrow P(y | x, w) = \frac{1}{1 + e^{-y w^T x}}$$

→ Maximum de vraisemblance (Likelihood) rappel:

• on dispose d'un ensemble de données : $D = \{(x_i, y_i)\}_{i=1}^N$

• on cherche w : $P(D|w)$ maximal

$$w_{MLE} = \arg \max_w P(D|w)$$

MLE = max Likelihood estimator

Exemple ⇒ Rappel: max vraisemblance:

→ on calcule la probabilité qu'un étudiant porte des lunettes. on a $P(y = +1) = \theta$

$$P(y = +1) = \theta \Rightarrow \text{porte des lunettes}$$

$$P(y = 0) = 1 - \theta \Rightarrow \text{ne porte pas des lunettes}$$

on dit que $y \sim \text{Bernoulli}(\theta)$ si l'on doit estimer à θ car on ne le connaît pas.

→ on a un fichier de données. on prend au hasard une certaine étude

y_1, y_2, \dots, y_N avec $N = 100$. Des y_i vont être égaux à +1 ou 0.

pour l'échantillon 1 : on voit s'il porte des lunettes ou non (1/0) ...

→ $D = \{y_1, y_2, \dots, y_N\}$: fichier de données

→ on veut trouver θ .

→ dans on utilise la méthode de maximum de vraisemblance : ↴

→ $P(D|\theta)$ ⇒ connaissant θ quelle est la probabilité d'observer ces données.

$$P(D|\theta) = \prod_{i=1}^N P(y_i|\theta) \text{ car les } y_i \text{ sont indépendants}$$

$$\text{or } P(y|\theta) = \theta^y \cdot (1-\theta)^{1-y}$$

$$\begin{cases} y = 0 \Rightarrow 1 - \theta \\ y = 1 \Rightarrow \theta \end{cases}$$

$$P(y|\theta) = \begin{cases} \theta & \text{si } y = 1 \\ 1 - \theta & \text{si } y = 0 \end{cases}$$

→ on doit calculer le produit \prod et ensuite trouver son max.

$$P(D|\theta) = \prod_{i=1}^N P(y_i|\theta) = \prod_{i=1}^N \theta^{y_i} \cdot (1-\theta)^{1-y_i}$$

$$\Rightarrow P(D|\theta) = \theta^{\sum y_i} \cdot (1-\theta)^{N - \sum y_i} \quad \triangleleft \text{ Je faut la maximiser.}$$

→ lorsqu'on a des exposants comme ça, maximiser cette fonction c'est son logarithme

c'est la même chose.

$$\text{D'où : } \ln(P(D|\theta)) = \sum_{i=1}^N y_i \ln(\theta) + (N - \sum_{i=1}^N y_i) \ln(1-\theta)$$

on dérive $\frac{\partial}{\partial \theta}$ car on cherche θ qui maximise cette fonction.

$$\frac{\partial \ln(P(D|\theta))}{\partial \theta} = \frac{\sum_{i=1}^N y_i}{\theta} - \frac{(N - \sum_{i=1}^N y_i)}{1-\theta} = \frac{(1-\theta)\sum_{i=1}^N y_i - \theta(N - \sum_{i=1}^N y_i)}{\theta(1-\theta)}$$

$$\text{maximum } \Rightarrow \frac{\partial \ln(P(D|\theta))}{\partial \theta} = 0$$

$$\Rightarrow (1-\theta) \sum_{i=1}^N y_i - \theta(N - \sum_{i=1}^N y_i) = 0$$

$$\Rightarrow \sum_{i=1}^N y_i - \theta \sum_{i=1}^N y_i - \theta \cdot N + \theta \sum_{i=1}^N y_i = 0$$

$$\Rightarrow \theta \cdot N = \sum_{i=1}^N y_i$$

$$\Rightarrow \theta_{MLM} = \frac{\sum y_i}{N}$$

$$\theta_{MLE} = \frac{\sum_{i=1}^N y_i}{N}$$

c'est le pourcentage de ceux qui portent des lunettes dans notre échantillon D .

→ dans le fichier D : si 30 personnes portent des lunettes on aura : $\theta = \frac{30}{100} = 0.3$

$$\text{si } P(y = +1 | \theta) = \theta$$

$$\text{et } P(y = -1 | \theta) = 1 - \theta$$

→ Formule générale :

$$P(y | \theta) = \theta^{\frac{y+1}{2}} \cdot (1-\theta)^{\frac{1-y}{2}}$$

on utilise la méthode du maximum de vraisemblance pour trouver l' θ .

Maximum de vraisemblance : Régression linéaire :

→ cette méthode marche pour la régression linéaire.

$$y_i = w^t x_i + \varepsilon_i$$

erreur

le moins

$$P(D|w) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2} \cdot \frac{(y_i - w^t x_i)^2}{\sigma^2}}$$

Densité de la loi normale

$$\Rightarrow P(D|w) = \left(\frac{1}{\sqrt{2\pi}\sigma} \right)^N \cdot N \cdot e^{-\frac{1}{2} \sum_{i=1}^N \frac{(y_i - w^t x_i)^2}{\sigma^2}}$$

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2} \cdot \frac{|x-\mu|^2}{\sigma^2}}$$

$$w_{MLE} = \arg \max_w (P(D|w))$$

$$w_{MLE} = \arg \min_w \sum_{i=1}^N (y_i - w^t x_i)^2$$

Estimateur w des moindres carrés = w_{MLE}

C'est la même quantité qu'on veut minimiser lorsque on a fait la régression.

Suite : Régression logistique : p48

→ La fonction perte $L(h(x_i), y_i)$ augmente si y_i et $w^t x_i$ sont de signes différents (mal classé)

→ elle diminue si y_i et $w^t x_i$ sont du même signe. (Bien classé)

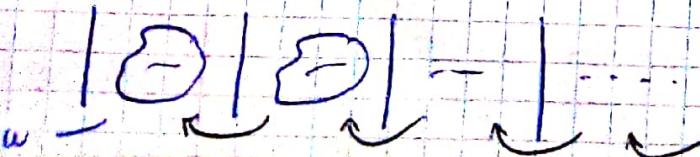
$$\rightarrow y_i = 1 \text{ ou } -1$$

$$\nabla_w L = \frac{-y_i x_i e^{-y_i w^t x_i}}{1 + e^{-y_i w^t x_i}} = \frac{-y_i x_i}{1 + e^{y_i w^t x_i}}$$

Pour les réseaux de neurones on utilise la méthode :

Computational graph

on a plusieurs couches de w à calculer et la sortie. On n'a pas dérivé % à toutes les couches



sortie \Rightarrow ça devient compliqué \rightarrow suis

seance 5 : 03-05-2022

c'est la méthode de descente du gradient par back propagation.
dans le cas simple

$$\begin{array}{c} x \\ w \\ y \end{array} \rightarrow \boxed{z = y w^t x} \rightarrow \boxed{E = e^{-z}} \rightarrow \boxed{U = \ln(1+E)}$$

$$E = e^{-y w^t x}$$

formule finale

entrées → on peut calculer la dérivée de U par rapport à w .

logiciel: Tensorflow utilisé dans la machine learning. En particulier, dans les réseaux de neurones. Lorsqu'on a un réseau de neurones comme ça il nous donne le graphe et nous on lui donne les équations. C'est à travers du graphe que le logiciel calcule la dérivée.

→ on peut calculer $\frac{\partial U}{\partial w}$ donc c'est : $\frac{\partial U}{\partial w} = \frac{\partial U}{\partial E} \cdot \frac{\partial E}{\partial z} \cdot \frac{\partial z}{\partial w}$

$$\Rightarrow \frac{\partial U}{\partial w} = \frac{1}{1+E} (-e^{-2}) y x$$

$$\Rightarrow \frac{\partial U}{\partial w} = \frac{-y x e^{-y w^t x}}{1 + e^{-y w^t x}}$$

→ la fonction de la régression logistique est convexe. $-\ln(P(C|w))$ dans elle a un seul minimum.

Algorithm: Régression logistique:

Tirer un w aléatoirement

Pour $i=1$ jusqu'à M (souvent 100, 200)

Début

Prendre un (x_i, y_i)

$$w \leftarrow w + \frac{ay_i x_i}{1 + e^{y_i w^t x_i}} \quad (\text{descente de gradient stochastique}).$$

Fini

Retour w

→ le gradient $\langle 0 | \rightarrow$ une fois cela à w c'est terminé.

NB

→ si L est difficile à dériver: on la remplace par :

$$\frac{L(w+\varepsilon) - L(w-\varepsilon)}{2\varepsilon}, \varepsilon \text{ petit.}$$

→ Si mmr am a une nouvelle observation x . Par exemple un patient avec les tests du cholestérol: on pourra calculer la probabilité $P(y = +1 | x, w)$

$$x : P(y = +1 | x, w) = \frac{1}{1 + e^{-w^t x}} = \frac{1}{1 + e^{-w^t x}}$$

la proba prédictive la valeur réelle
on remplace y par 1

$$P(y = -1 | x, w) = \frac{1}{1 + e^{w^t x}}$$

⇒ donc on peut calculer la probabilité d'avoir une attaque cardiaque ou non pour une nouvelle observation x et en remettant de calculer sur un fichier d'apprentissage et venant de trouver b et w .

TP: Kmme

- le fichier breast → étude du cancer du sein.
 - La dimension du vecteur x_i : c'est $d = 9$
 - y_i : classe : qu'on veut prédire par la suite.
 - on a suffisamment de données donc 90% → apprentissage
10% → test ayant
 - Pour la régression : on choisit les variables significatives $\rightarrow P_{\text{value}} < 0.05$ Δ
 - lorsqu'on filtre les données : le taux (taux de classification) de bonne classification a augmenté.
- Plus la dimension est grande → plus c'est mauvais.
- on ajoute 2 colonnes pour les proba.

si proba (malign) $> 50\%$ → prediction: malign

si proba (benign) $> 50\%$ → prediction: + benign

→ la méthode de descente du gradient est d'ordre 1 → on dévele oppe jusqu'à l'ordre 1

→ la méthode de Newton est d'ordre 2.

$$\nabla L(w) = L(w_0) + (w - w_0)^t \nabla_w L(w_0) + \frac{1}{2} (w - w_0)^t H(w - w_0) + \dots$$

H: dérivé seconde $L(w) = \frac{\partial^2 L(w)}{\partial w^t \partial w} = H$

on veut que $L(w)$ soit minimale. donc on dérive $L(w)$ par w :

$$\frac{\partial L(\omega)}{\partial \omega} = 0 + \nabla_{\omega} L(\omega_0) + H(\omega - \omega_0) + \dots = 0$$

$$H(\omega - \omega_0) = -\nabla_{\omega} L(\omega_0) \quad H \text{ matrice symétrique}$$

$$\Rightarrow \omega - \omega_0 = -H^{-1}(\omega_0) \nabla_{\omega} L(\omega_0) \quad (x^t A x)' = 2Ax$$

$$\Rightarrow \omega(t+1) = \omega(t) - H^{-1}(\omega(t)) \nabla_{\omega} L(\omega(t))$$

Exemple: Calcul du Hessien:

$$\text{soit } f(\omega) = 2\omega_1\omega_2 + 3\omega_2^2$$

$$\nabla f(\omega) = \begin{pmatrix} 2\omega_2 \\ 2\omega_1 + 6\omega_2 \end{pmatrix}; \quad H = \begin{pmatrix} \frac{\partial f}{\partial \omega_1^2} & \frac{\partial f}{\partial \omega_1 \partial \omega_2} \\ \frac{\partial f}{\partial \omega_2 \partial \omega_1} & \frac{\partial f}{\partial \omega_2^2} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & 6 \end{pmatrix}$$

dérivé de $f(\omega)$ % ω_1

dérivé de $f(\omega)$ % ω_2

H^{-1} : il faut inverser la matrice H

Exemple pour la régression: (avec la méthode de Newton)

solution avec la méthode de Newton:

$$L(h(x), y) = (y - \omega^t x)^2$$

$$\begin{aligned} E_{lm}(\omega) &= \text{coût} = \frac{1}{N} \|y - X\omega\|^2 = \frac{1}{N} (y^t y - 2\omega^t x + \omega^t X^t x \omega) \quad (\text{déjà vu}) \\ &= \frac{1}{N} \sum_{i=1}^N (y_i - \omega^t x_i)^2 \end{aligned}$$

$$\nabla E_{lm}(\omega) = \frac{1}{N} (-2X^t y + 2X^t X \omega) \quad (\text{déjà vu})$$

$$\text{donc } H(\omega) = \frac{2X^t X}{N} \quad \text{l'Hessien pour la régression.}$$

On utilise la méthode de Newton sur la méthode de gradient.

on remplace dans $\omega(t+1) = \omega(t) - H^{-1}(\omega(t)) \nabla_{\omega} L(\omega(t))$

$$\begin{aligned} \omega(t+1) &= \omega(t) - \frac{N}{2} (X^t X)^{-1} \left(\frac{1}{N} (-2X^t y + 2X^t X \omega(t)) \right) \\ &= \omega(t) + (X^t X)^{-1} X^t y - \omega(t) \end{aligned}$$

$$\Rightarrow \omega(t+1) = (X^t X)^{-1} X^t y \quad \text{convergence en une itération.}$$

Pourquoi la méthode de Newton dans la régression converge en 1 seule itération?

Quand l'équation du coût est de 2^{ème} ordre (WW) lorsque on la dérive 2 fois toutes les méthodes quadratiques avec la méthode de Newton se marche. On y a plus d'erreur après la 2^{ème} dérivée. On néglige rien dans $L(\omega)$ donc $\omega(t+1)$ n'est pas une approximation: c'est exacte \rightarrow en 1 seule itération.

* Pour la régression logistique souvent représentée avec 1 et 0.

Exo: $P(y = +1 | w, x) = \frac{1}{1 + e^{-w^t x}}$

$P(y = 0 | w, x) = \frac{1}{1 + e^{w^t x}}$

en utilisant: $P(D | w, x) = \prod_{i=1}^m \left(\frac{1}{1 + e^{-w^t x_i}} \right)^{y_i} \left(\frac{1}{1 + e^{w^t x_i}} \right)^{1-y_i}$ rema utile la formule déjà vu

→ déterminer la formule qui génère l'ix.

→ on utilise le logarithme... la formule va changer

→ il faut marquer sur $P(D | w, x)$ % w

→ Régression logistique / Problème \Rightarrow Bininaire (2 classes possibles: malade ou pas / tumoreux: benignes/maligne / tache dans les impacts ou pas)

→ si la classe contient plus de 2 classes c'est la variable à prédire contient plus de 2 classes (ex: des fleurs : 3 types)

↳ Modèle de la régression logistique Multi-class (Multi-mammale), p55

Exemple

sac contenant des Balles: 5 bleus, 6 Rouges

Odds(Bleu) = $5 : 6$ (échelle)
= $5/6$

gain

dans le cas de la régression logistique: $\ln \left(\frac{\pi}{1-\pi} \right) = w^t x = \text{logit}(\pi)$



Binnaire

perte

$P(y = +1 | w, x) = \pi$

$P(y = -1 | w, x) = 1 - \pi$

→ Odds($y = 1$) = $\pi : 1 - \pi$

Odds($y = -1$) = $1 - \pi : \pi$

ex: Pil: $\frac{1}{4}$. Fac: $\frac{3}{4}$ || ~~meilleur~~ ~~plus~~

Odds(Pil) = $\frac{1}{4} : \frac{3}{4} = \frac{1}{3}$

Odds(Fac) = $\frac{3}{4} : \frac{1}{4} = 3$

Pour la régression logistique, $\pi = P(y = +1 | w, x) = \frac{1}{1 + e^{-w^t x}} = \frac{e^{w^t x}}{e^{w^t x} + 1}$

$1 - \pi = P(y = -1 | w, x) = \frac{1}{1 + e^{w^t x}}$

$$\Rightarrow \frac{\pi}{1-\pi} = \frac{1+e^{w^t x}}{1+e^{-w^t x}} = \frac{e^{w^t x}}{e^{-w^t x}} \cdot \frac{1+e^{w^t x}}{1+e^{-w^t x}} = e^{w^t x}$$

$$\Rightarrow \text{logit}(\pi) = \ln\left(\frac{\pi}{1-\pi}\right) = w^t x \quad \text{linaire}$$

on a comparé à 1 et à -1.

$\pi > 1 - \pi$ si $\ln > 0$ si $w^t x > 0 \Rightarrow \text{gain}$

$\pi < 1 - \pi$ si $\ln < 0$ si $w^t x < 0 \Rightarrow \text{perte}$

cas: plusieurs classes:

$$y \in \{1, 2, \dots, k\} \quad k > 2$$

on prend la dernière classe comme référence:

$$\ln\left(\frac{P(y=1)}{P(y=k)}\right) = w_1^t x$$

comme si on fait plusieurs regressions logistiques.

$$\ln\left(\frac{P(y=2)}{P(y=k)}\right) = w_2^t x$$

on doit calculer $k-1$ vecteurs:

$$\ln\left(\frac{P(y=k-1)}{P(y=k)}\right) = w_{k-1}^t x$$

$$w_1^t x, w_2^t x, \dots, w_{k-1}^t x$$

$$\Rightarrow \ln\left(\frac{P(y=l)}{P(y=k)}\right) = w_l^t x \quad ; \quad 1 \leq l \leq k-1$$

$$\Rightarrow P(y=l) = P(y=k) \cdot e^{w_l^t x} \quad ; \quad 1 \leq l \leq k-1$$

$$\Rightarrow \underbrace{\sum_{l=1}^{k-1} P(y=l)}_{1 - P(y=k)} = P(y=k) \cdot \sum_{l=1}^{k-1} e^{w_l^t x}$$

$$\Rightarrow 1 - P(y=k) = P(y=k) \sum_{l=1}^{k-1} e^{w_l^t x}$$

$$\Rightarrow \begin{cases} P(y=k) = \frac{1}{1 + \sum_{l=1}^{k-1} e^{w_l^t x}} \\ P(y=l) = \frac{e^{w_l^t x}}{1 + \sum_{l=1}^{k-1} e^{w_l^t x}} \end{cases} \quad 1 \leq l \leq k-1$$

→ Estimer les $w \Rightarrow$ Méthodes de Vraisemblance:

$$P(y=l) = \frac{e^{w_l^t x}}{1 + \sum_{l=1}^{k-1} e^{w_l^t x}} = \pi_l \quad ; \quad 1 \leq l \leq k-1$$

$$P(y=k) = \dots = \pi_k$$

→ estimer $\pi(x)$:

$$L = \prod_{i=1}^m \left(\prod_{m=1}^M \pi_m^{y_m(x_i)} \right) \text{ avec } \begin{cases} y_m(x_i) = 1 \text{ si } y_i = m \\ y_m(x_i) = 0 \text{ si } y_i \neq m \end{cases}$$

→ il faut maximiser L (dérivée...)

NB

→ les autres méthodes: lorsque on fait un DL: on dit c'est que le nouveau w qui est proche de w_0 , on néglige les termes

par ex: méthode du descente du gradient on a négligé les termes à partir de la fonction $(x_2 + g_2 + \dots)$

→ méthode quadratique: on néglige tout du tout car y a plus aucun terme à négliger. C'est plus une approximation donc dès la 1^{ère} itération on trouve la solution (Méthode de Newton).

dérivée seconde = cté

dérivée troisième = 0 donc y a plus rien.

séance 6 : 10-05-2021

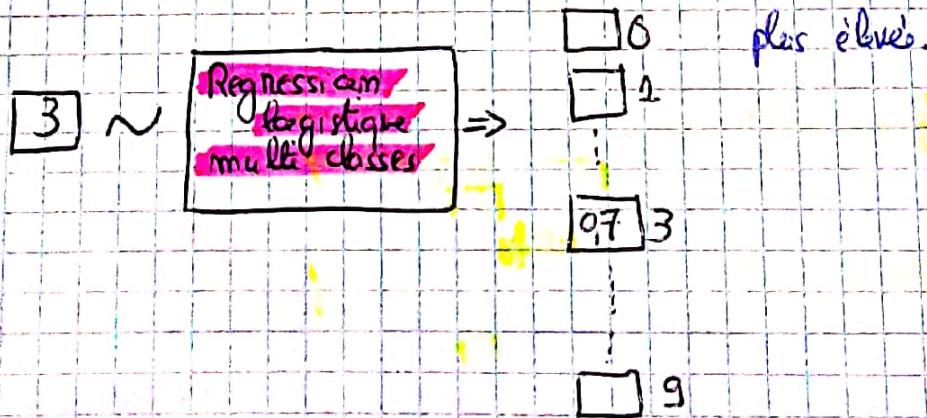
$$y = \{1, 2, \dots, k\}$$

exemple :

• les fleurs (vergimico, setosa, versicolor)

• reconnaissance des chiffres

→ on introduit en entrée l'image d'un chiffre par exemple 3 ensuite il va sortir 10 probabilités pour que ça soit 0, proba pour qu'il soit 1 ... 9. Dans le 3 on a bien une forte probabilité par ex 0,7 et donc on affecte la classe à la probabilité la



→ on a plusieurs classes au lieu d'avoir 1 seule.

→ on a démontré que:

Regression logistique multi classe (la dernière class comme référence)

$$P(y = l) = \frac{e^{w_l^T x}}{1 + \sum_{k=1}^{K-1} e^{w_k^T x}} = \pi_l \quad ; \quad 1 \leq l \leq K-1$$

$$P(y = K) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{w_l^T x}} = \pi_K$$

w : vecteur de $d+1$ dimensions

$$P(y = K) + \sum_{l=1}^{K-1} P(y = l) = 1$$

→ pour trouver les π_l on utilise la méthode du maximum de vraisemblance.

$$L = \prod_{i=1}^N \left(\prod_{m=1}^K (\pi_m)^{y_m(x_i)} \right)$$

sous la contrainte

$$\sum_{m=1}^K \pi_m = 1$$

prend en toutes les classes

prend en toutes les échantillons

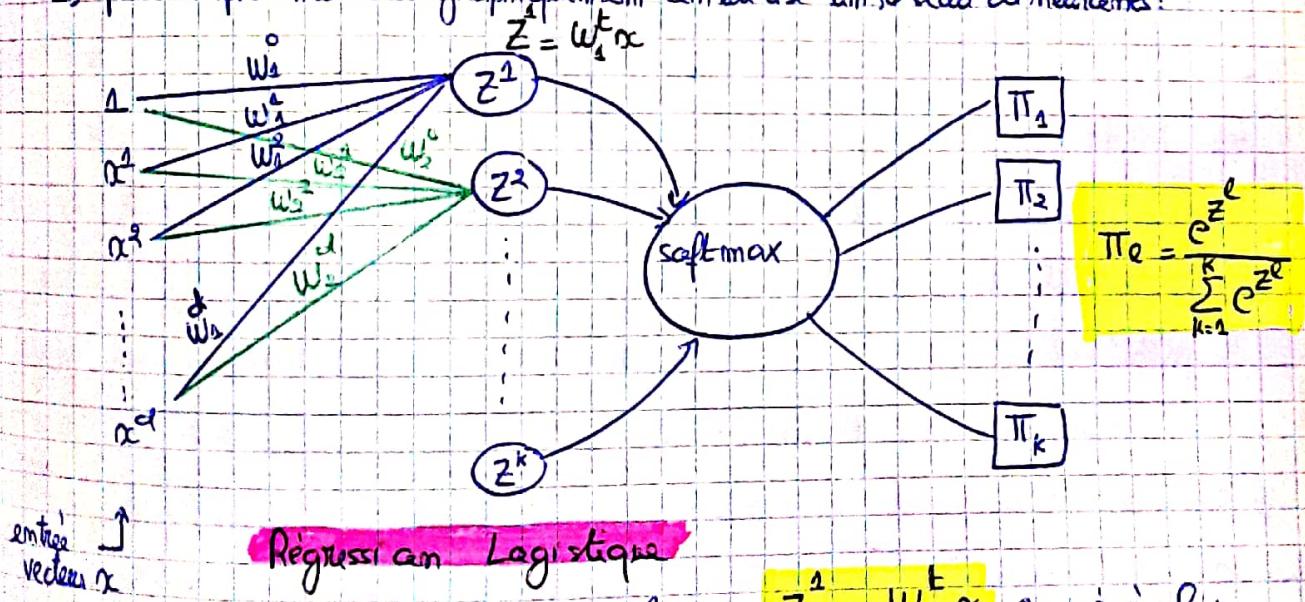
$$\text{et } \begin{cases} y_m(x_i) = 1 & \text{si } y_i = m \\ y_m(x_i) = 0 & \text{si } y_i \neq m \end{cases} \text{ donc}$$

$$\prod_{m=1}^K (\pi_m)^{y_m(x_i)} = \pi_m \quad (1 \text{ seul } 1 \text{ dans})$$

$$\Rightarrow m = y_i$$

→ pour trouver les estimations des π_l

→ pour représenter ceci graphiquement on utilise un réseau de neurones:



Régression Logistique

→ on construit un vecteur de K classes $Z^T = W^T x$ en général:

$Z^T = W^T x$. Ici Z c'est comme des moyennes des K classes.

→ maintenant on va chercher les probabilités associées. On applique une fonction qui

On forme un vecteur en des probabilités appelé softmax. Elle calcule les probabilités π_1 (classe 1), π_2 , ..., π_K (pour que π_k soit de la classe k).

→ On applique softmax sur les Z et $\pi_e = \frac{e^{z_e}}{\sum_{k=1}^K e^{z_k}}$

$$\pi_e = \frac{e^{z_e}}{\sum_{k=1}^K e^{z_k}} : \begin{array}{l} \text{exp} : \text{sa transforme les combinaisons} \\ \text{linéaires en exponentiel} \end{array}$$

$$\sum_{k=1}^K \pi_k = 1 :$$

⇒ On a K : W et chaque W est de dimension $d+1$

→ On peut déduire 1 probabilité à partir des autres car $\sum_{k=1}^K \pi_k = 1$

Fonction qui prend comme entrée un vecteur en une probabilité.

import numpy as np

def softmax(z):

$u = \text{np.exp}(z)$ // calcul l'exp d'un vecteur

return u / u.sum()

$z = \text{np.array}([5, 27, 9])$ // 3 classes

V = softmax(z)

print(V, V.sum())

⇒ [2.78966805e-10, 9.9999984e-01, 1.52299795e-03] 1.0

→ Il a transformé le vecteur $[5, 27, 9]$ en des probabilités

→ La probabilité de 5 est presque nulle,

→ La probabilité de 27 est la plus grande

→ La somme de toutes les probabilités = 1

→ softmax: on forme un vecteur en des probabilités de classes.

→ On utilise les méthodes déjà vues (descente de gradient par exemple) il faut au début initialiser tous les W ensuite on établit la méthode de descente de gradient par exemple pour déterminer le super vecteur de W en matrice de W .

$$\begin{pmatrix} z^1 \\ z^2 \\ \vdots \\ z^k \end{pmatrix} = \begin{pmatrix} t \\ w_1^t x \\ w_2^t x \\ \vdots \\ w_k^t x \end{pmatrix} = \underbrace{\begin{pmatrix} t \\ w_1^t \\ w_2^t \\ \vdots \\ w_k^t \end{pmatrix}}_W \cdot \underbrace{x}_\alpha$$

matrice W
↓
lignes : k
colonnes : d+1
matrice W

$$Z = W \cdot \alpha$$

→ donc on peut calculer d'une façon vectorielle toutes les vecteurs Z (meilleurs) d'un seul coup (1 seul produit matriciel)

→ si on a une autre couche on peut appliquer une autre matrice pour trouver d'autres meilleurs

⇒ donc on peut faire ça plusieurs fois.

$$z_1 \quad z_2$$

$$\begin{matrix} 1 \\ x^1 \\ \vdots \\ x^d \end{matrix} \quad \begin{matrix} \vdots \\ \vdots \end{matrix}$$

$$z_1 = W^1 \alpha$$

$$z_2 = \text{matrice } \times z_1 \quad (\text{si on applique pas une fonction d'activation.})$$

→ on fait des multiplications matricielles.

→ pour faire de la régression logistique on doit apprendre tous les W :

avec W matrice de $k \times (d+1)$ mais on peut manquer qu'on a besoin que de $(k-1) \times d$ car la somme des probabilités = 1 donc si on connaît toutes les probabilités sauf une donc on peut la déduire à partir des autres, on a pas besoin de son w_k pour la calculer.

Régression logistique multiconnexionne:

import numpy as np

import sklearn import datasets # librairie pour flux de machine learning

from sklearn.model_selection import train_test_split # sklearn contient plus d'outils

Import des modules Keras (Keras est une bibliothèque pour la création des réseaux de neurones)

from keras import models

from keras import layers

from keras.utils import to_categorical

Load the iris dataset

#

iris = datasets.load_iris()

X = iris.data # (long sépal, largeur sépal, long pétil, largeur pétil)

y = iris.target # (setosa, versicolor, virginica)

Create training and test split

#

X_train, X_test, y_train, y_test = train_test_split(X, y,

test_size = 0.20, stratify = y, random_state = 42)

Create categorical labels

#

train_labels = to_categorical(y_train)

Dense \Rightarrow toutes les classes possibles

test_labels = to_categorical(y_test) input_shape = (4,) \Rightarrow 4 en trés (il y a 4 à 1 pour la seconde)

print(test_labels)

#

Create the network (le modèle)

on crée une couche avec 3 Z sauf laquelle on va appliquer la fonction softmax

network = models.Sequential()

Sequential \Rightarrow couche après couche

network.add(layers.Dense(8, activation='relu'))

network.add(layers.Dense(3, activation='softmax', input_shape=(4,)))

#

Compile the network

#

network.compile(optimizer='sgd', # sgd \Rightarrow descente de gradient stochastique loss="categorical_crossentropy", # la fonction perte metrics=['accuracy'])

#

Fit the neural network (entraînement) \Rightarrow le modèle est prêt \Rightarrow on entre dans le fichier du dossier

network.fit(X_train, train_labels, epochs=100)

epoch(100) durant 0.90 acc Répéter 100 fois \rightarrow ça permet de converger. 0.42 loss

Get the accuracy of test data set. \Rightarrow Prediction

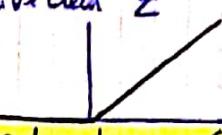
test-loss, test-acc = metwork.evaluate(X_test, test_labels)

Print the test accuracy

print("Test Accuracy", test-acc, "\nTest loss", test-loss) \Rightarrow 0,96 accuracy
0,39 loss

\Rightarrow On ajoute une autre couche et on va faire 8 nn travail.
avant la couche des Z

\Rightarrow elle contient 8 neurones Z_1, Z_2, \dots, Z_8 et sur cette couche on applique la fonction logistique et on sort un peu par le vecteur Z

\Rightarrow il faut ajouter cette ligne. $\text{relu} \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } x > 0 \end{cases}$ \Rightarrow 

network.add(layers.Dense(8, activation='relu', input_shape=(4,)))

\Rightarrow et modifier l'autre ligne comme suit.

network.add(layers.Dense(3, activation='softmax'))

\Rightarrow et on exécute

\Rightarrow on remarque que l'accuracy augmente sur le fichier test et sur le fichier train.

\Rightarrow on ayant une autre couche \rightarrow le taux de Bonne classification a augmenté

\Rightarrow on peut ajouter d'autres couches

\Rightarrow on ajoute une 3ème couche avant la couche finale.

network.add(layers.Dense(5, activation='relu'))

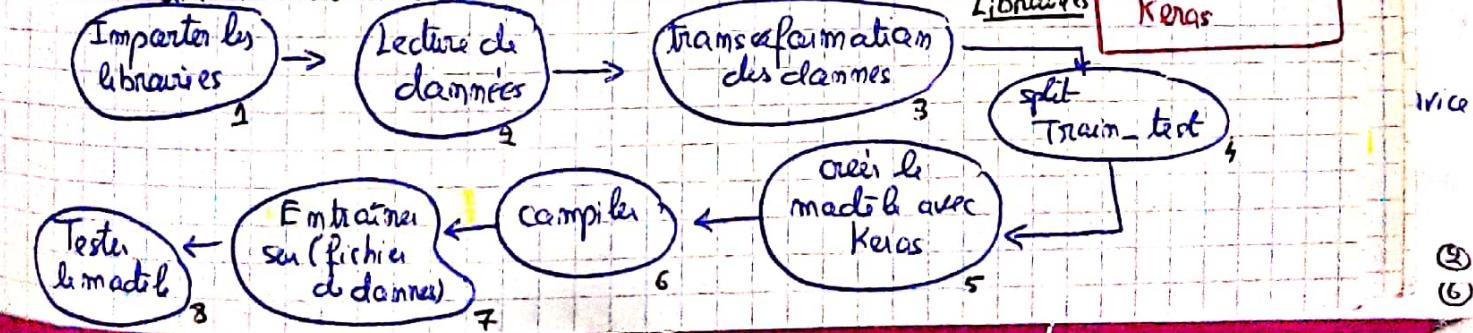
↑
5 neurones

\Rightarrow et on exécute. \Rightarrow on remarque que cette couche n'a servi rien du tout

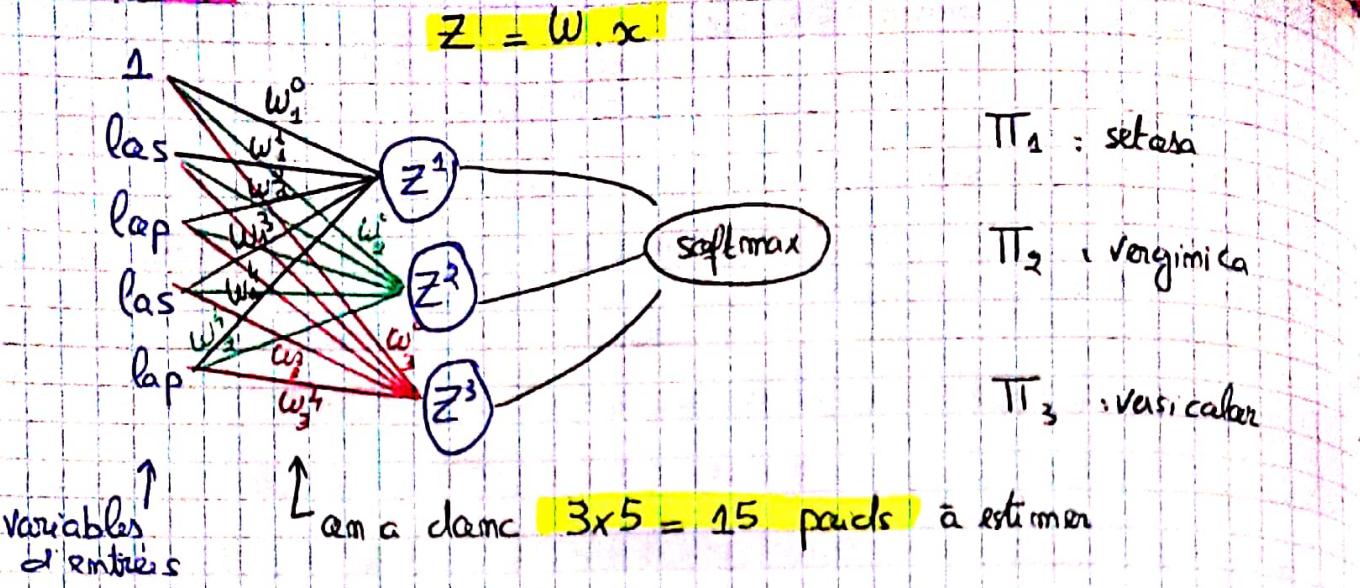
car le taux de Bonne classification a diminué sur le fichier de d'apprentissage et sur le fichier test. (on ne recalcule pas le début)

\Rightarrow donc il ne faut pas ajouter de plus en plus de couches pour rien du tout

↑ créer un réseau avec Keras



Iris



Π_1 : setosa

Π_2 : virginica

Π_3 : versicolor

- devrait constituer $3 \cdot Z^1, Z^2, Z^3$ car on a 3 classes.
 - on a 3 Z qui vont former un probabilité donc pour utiliser la fonction softmax. Et qui va nous donner 3 probabilités.
 - Π_1 : probabilité pour que ça soit setosa
 - Π_2 : " " " virginica
 - Π_3 : " " " versicolor
- la matrice W : 3×5 : 3 lignes et 5 colonnes
- \uparrow \uparrow
mb classes mb Variables + 1
- $Z = W \cdot x$ → on obtient une matrice Z (3×1) sur lequel on applique la fonction softmax pour calculer les probabilités.

NB

- sklearn : bibliothèque contenant tous ce qu'il faut pour faire de machine learning.
- keras : bibliothèque pour faire de deep learning (réseaux de neurones).
- parmi les fichiers de données qu'on a sur sklearn.datasets : le fichier iris.

Fichier de données	matrice X	classe y												
	las lap las lap	setosa												
		virginica												
		versicolor												
		setosa versicolor virginica												
		S V V												
→ on transforme le y ⇒		<table border="1"> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>.</td><td>.</td><td>.</td></tr> </table>	1	0	0	0	0	1	0	1	0	.	.	.
1	0	0												
0	0	1												
0	1	0												
.	.	.												
		Hat-coding												

on transforme le y en hot-coding. On transfoit parmi chaque catégories un vecteur contenant 1 si c'est la catégorie et 0 partout.

↳ cela est fait avec la méthode **to_categorical** de Keras.

Keras \rightarrow deep learning (crée des sous-sous de mélange)

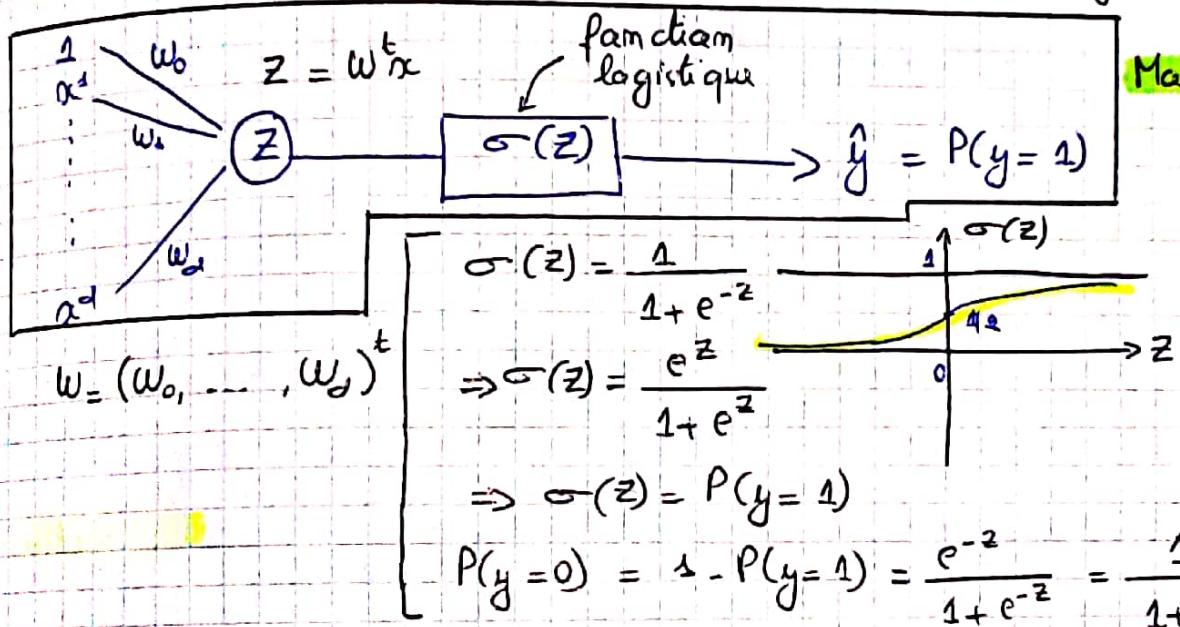
sklearn \rightarrow machine learning

TD n°1 :

Exercice ① Régression logistique

On se propose dans cet exercice de repérer la modélisation de la régression logistique binaire en codant les deux valeurs du vrai (target) par 1 pour la classe positive et 0 (au lieu de -1) pour la classe négative. [1 et 0 comme le Perceptron]

a) Rappeler les formules des probabilités : $P(y=1|w,x)$ et $P(y=0|w,x)$:



$0 < \sigma(z) < 1 \Rightarrow \sigma(z)$ c'est la probabilité (\hat{y}) qu'on veut prédire.

\rightarrow on a un seul Z .

\rightarrow c'est binaire donc on doit avoir Z_1 et Z_2 mais on économise un Z car on sait que $P(y=0) = 1 - P(y=1)$

$\rightarrow \hat{y}$: mesure de la probabilité (on a remplacé 1 et -1 par 1 et 0 car la proba ne peut pas être égale à -1)

$$\Rightarrow Z = w^T x, \quad P(y=1) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} \quad | P(y=0) = \frac{1}{1+e^z} = \frac{e^{-z}}{1+e^{-z}}$$

annote :

$$\begin{cases} \pi = P(y=1) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} \\ 1-\pi = P(y=0) = \frac{1}{1+e^z} = \frac{e^{-z}}{1+e^{-z}} \end{cases}$$

b) écrivez une seule formule $P(y|w, z)$ englobant les 2 formules précédentes.

Variable en m temps pour $y=1$ et $y=0$

$$P(y) = \pi^y \cdot (1-\pi)^{1-y} \quad \text{formule générale}$$

c) quelle est la formule du vraisemblance $L(D|w)$ dans ce cas (D : dataset).

$$P(y_i) = \pi_i^{y_i} \cdot (1-\pi_i)^{1-y_i} \quad \text{variable pour tous les } (x_i, y_i)$$

$$D = \{(x_i, y_i)\}_{i=1}^N$$

\Rightarrow la fonction de vraisemblance des données D :

$$L(D|w) = \prod_{i=1}^N P(y_i) \quad \text{fonction de vraisemblance.}$$

$$\begin{aligned} &= \prod_{i=1}^N \pi_i^{y_i} \cdot (1-\pi_i)^{1-y_i} \\ &= \pi_1^{y_1} \cdots \pi_N^{y_N} \cdot (1-\pi_1)^{N-y_1} \cdots (1-\pi_N)^{N-y_N} \end{aligned}$$

$$L(D|w) = \prod_{i=1}^N \pi_i^{y_i} \cdot (1-\pi_i)^{1-y_i}$$

d) Prenez le grec logarithme de c) et montrez que $\ln(L(D|w))$ est convexe et donc possède un seul minimum.

$$\ln(L) = \sum_{i=1}^N \left[y_i \cdot \ln(\pi_i) + (1-y_i) \cdot \ln(1-\pi_i) \right]$$

$$\pi_i \cdot \text{proba d'obtention} \rightarrow \pi_i = \hat{y}_i = P(y=1)$$

$$\text{caul} = \sum_{i=1}^N \left[-y_i \cdot \ln(\pi_i) - (1-y_i) \cdot \ln(1-\pi_i) \right] \quad \text{on cherche à minimiser}$$

$\max \ln(L) \Rightarrow$ minimiser caul-

le caul est calculé sur toutes les observations

e) donner la formule de Loss pour une seule observation (interpréter).

1 seule observation \Rightarrow descente de gradient stochastique (1 seule donnée)

$$\hookrightarrow \text{la fonction porte } L(y_i, \pi_i)$$

$$cout = \sum_{i=1}^N L(y_i, \hat{y}_i)$$

$$L(y_i, \hat{y}_i) = - (y_i \ln(\hat{y}_i) + (1-y_i) \ln(1-\hat{y}_i)) \quad | \Delta \text{ seul observation}$$

$$L(y, \hat{y}) = - (y \ln(\hat{y}) + (1-y) \ln(1-\hat{y})) \quad \text{cross entropy}$$

→ Il faut minimiser le Loss pour chaque observation.

→ donc on minimise pour toutes les observations.

$$y=1 \Rightarrow L_{\text{loss}} = - \ln(\hat{y}) \quad \hat{y} \in [0, 1]$$

$$\Rightarrow \text{la valeur qui minimise le loss } L \text{ c'est } \hat{y} = 1 \Rightarrow L_{\text{loss}} = 0$$

$$y=0 \Rightarrow L_{\text{loss}} = (y-1) \ln(1-\hat{y}) = -\ln(1-\hat{y})$$

$$\Rightarrow \text{la valeur qui minimise le loss } L \text{ c'est } \hat{y} = 0 \Rightarrow L_{\text{loss}} = 0$$

$$f). \text{ Probabilité : } P(y=1) = \frac{e^z}{1+e^z} = \frac{e^{w^t x}}{1+e^{w^t x}} = \frac{e^{w^t x}}{1+e^{w^t x}}$$

f) quel est le gradient associé à e) entrées

$$\begin{array}{c} x \\ y \\ w \end{array} \Rightarrow \boxed{z = w^t x} \rightarrow \boxed{\sigma(z) = \hat{y}} \rightarrow L_{\text{loss}} = L = - (y \ln \hat{y} + (1-y) \ln(1-\hat{y}))$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

à l'loss : on perd quand on remplace y par \hat{y}

→ l'objectif c'est de chercher w entrée qui permet de réduire le cout.

$$\rightarrow \text{donc on dérive } L = - (y \ln \hat{y} + (1-y) \ln(1-\hat{y})) \text{ par rapport à } w$$

$$\text{min. } \nabla_w L = \frac{\partial L}{\partial w} = \frac{\partial L}{\partial \sigma(z)} \cdot \frac{\partial \sigma(z)}{\partial z} \cdot \frac{\partial z}{\partial w} \quad \begin{aligned} &\Rightarrow \text{on doit calculer ces 3 dérivées.} \\ &\text{pour trouver le } w \text{ qui minimise} \\ &\text{le loss } L. \end{aligned}$$

→ gradient

$$\rightarrow \frac{\partial z}{\partial w} = \frac{\partial w^t x}{\partial w} = x \quad \left\{ \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\rightarrow \frac{\partial \sigma(z)}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} \quad \left\{ L(\sigma(z)) = - \left[y \ln(\sigma(z)) + (1-y) \ln(1-\sigma(z)) \right] \right.$$

$$= \frac{1}{1+e^{-z}} - \left(\frac{1}{1+e^{-z}} \right)^2$$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z) - (\sigma(z))^2 = \sigma(z)(1-\sigma(z))$$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z)) = y(1 - y) = \pi(1 - \pi)$$

$$\Rightarrow L = -(y \ln \sigma(z) + (1-y) \ln(1-\sigma(z)))$$

$$\frac{\partial L(\sigma(z))}{\partial \sigma(z)} = \frac{-y}{\sigma(z)} + \frac{1-y}{1-\sigma(z)} = \frac{-y + \sigma(z)}{\sigma(z)(1-\sigma(z))}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \sigma(z)} \cdot \frac{\partial \sigma(z)}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$$= \frac{\sigma(z) - y}{\sigma(z)(1-\sigma(z))} \cdot \sigma(z)(1-\sigma(z)) \cdot \alpha$$

$$\frac{\partial L}{\partial w} = \alpha (\sigma(z) - y)$$

Chabot François → Keras : il a développé Keras

↳ ENSTA → travail avec google

calcul matriciel supérieur des techniques avancées

Prix Turing → = prix nobel en informatique

$$\Rightarrow \frac{\partial L}{\partial w} = \alpha (\sigma(z) - y) = \alpha (\hat{y} - y) = \nabla_w (\text{coût})$$

$$\left(\frac{\partial \text{coût}}{\partial w} \right) \Rightarrow \frac{\partial L}{\partial w} = \alpha \left(\frac{1}{1 + e^{w^T x_i}} - y_i \right) \quad \text{pour l'absent}$$

$$\text{pour l'absent } i: \frac{\partial L}{\partial w} = \alpha_i \left(\frac{1}{1 + e^{-w^T x_i}} - y_i \right)$$

$$\Rightarrow \text{pour toutes les observations: } \sum_{i=1}^N \alpha_i \left(\frac{1}{1 + e^{-w^T x_i}} - y_i \right)$$

⇒ pour trouver les w qui minimise le co.

$$\sum_{i=1}^N -\alpha_i \left(\frac{1}{1 + e^{-w^T x_i}} - y_i \right) = 0$$

⇒ donc on doit faire un algorithme pour trouver les w

on va utiliser un algo itératif.

→ on a vu 2 algos itératifs:
Pour minimiser une fonction

descendre de gradient (1er cercle)

Méthode du ménutien (2nd cercle)

g) Ecrire l'algorithme de descente du gradient stochastique permettant de trouver la valeur optimale de w^*

$$w \leftarrow w - \alpha \nabla_w (L) \quad [\text{mini x à jour}]$$

Algorithme

$w \leftarrow 0$
pour $i = 1 \dots M$:

prendre un (x_i, y_i) de D qu'on a pris de D

$$\text{calculer } \nabla_w (L) = (\sigma(z_i) - y_i) x_i$$

$$w \leftarrow w - \alpha \nabla_w (L) \quad | \quad \frac{1}{1 + e^{w^T x_i}}$$

Imprimer w

→ c'est à w qu'on va utiliser pour faire la prédiction (calculer les probas)

→ c'est la méthode : descente de gradient stochastique car le gradient est calculé à partir d'une seule observation prise au hasard. On prend une observation par observation.

→ le gradient en totalité : à chaque fois il faut faire une batch pour calculer le gradient : \sum (Batch)

$$\frac{\partial L}{\partial w} = (\hat{y} - y)x$$

si $\hat{y} - y = 0$ (prédiction = valeur réelle) \Rightarrow donc on a rien perdu.

h) Ecrire l'algorithme permettant de trouver la valeur optimale w^*

la même chose de m'en souviens \Rightarrow il faut calculer le Hessian $\Rightarrow \frac{\partial^2 L}{\partial w^2}$

$$\text{Hessian} = \frac{\partial^2 L}{\partial w \partial w^t} \quad | \quad \frac{\partial L}{\partial w} = (\hat{y} - y)x = (\sigma(z) - y)x$$

$$\begin{aligned} \Rightarrow \frac{\partial^2 L}{\partial w \partial w^t} &= \frac{\partial \sigma(z)}{\partial w} \cdot x = \sigma(z)(1 - \sigma(z))x \\ &= \frac{\partial}{\partial w} \left(\frac{1}{1 + e^{-w^t x}} \right) x \end{aligned}$$

$$\Rightarrow \frac{\partial^2 L}{\partial w \partial w^t} = \frac{\partial \sigma(z)x}{\partial w} = \frac{\partial \sigma(z)x}{\partial z} \cdot \frac{\partial z}{\partial w^t} = \sigma(z)(1 - \sigma(z)).x.x^t$$

$$\text{Hessien} = \sigma(z)(1 - \sigma(z))x_i x^t$$

on sait que $\sum_i \sigma(z_i)(1 - \sigma(z_i))x_i \cdot x^t$ est positive

→ dérivé second > 0 ⇒ minimum

α = taux d'apprentissage = Learning Rate ⇒ apprend une partie du gradient.

→ α petit

→ si α grand → au risque de diverger.
→ au pénalise α qui diminue au cours des itérations.

la fonction L est convexe donc elle a un seul minimum.

Modèle de Régression (Régularisation).

en régression régularisée.

le terme qu'on cherche à minimiser dans la régression: $\|y - Xw\|^2$

→ de peur que les w prennent des valeurs très grandes on pénalise le coût par le terme $\lambda \|w\|^2$

⇒ il faut minimiser les 2 termes: $\|y - Xw\|^2 + \lambda \|w\|^2 \Rightarrow$ le coût est petit

→ Régression Ridge

$$\text{coût}_R = \|y - Xw\|^2 + \lambda \|w\|^2 \quad \text{avec } \lambda > 0$$

→ Régression LASSO:

$$\text{coût}_L = \|y - Xw\|^2 + \lambda \sum_j |w_j|$$

→ on pénalise les w pour ne pas avoir de grande valeur des w

→ on régularise le coût par une pénalité.

Voir l'exemple (PC)

= sur ajustement

si le degré 1 ⇒ over fitting (trop ajusté) ⇒ au risque d'avoir une erreur très grande
du Polynôme

on a suivie toutes les données pas à pas

→ les coefficients sont très grands

⇒ pénalité ⇒ on ne va pas permettre à w d'apprendre des valeurs très grandes.

⇒ pour corriger le over fitting on va prendre plus de observations.

on leu de minimiser $\|y - Xw\|^2$ amoyenne un terme de pénalisation qui malheur les w qui sont très grands.

d'am minimiser tout sa. $E(w) = \frac{1}{2} \sum \{y(x_m, w) - t_m\}^2 + \frac{\lambda}{2} \|w\|^2$

on cherche les w qui minimisent les 2 termes.

terme de régularisation.

$\lambda = 0 \Rightarrow y$ a pas de régularisation

$\lambda \uparrow \Rightarrow y$ a de régularisation

$\lambda = 1 \Rightarrow$ c'est le meilleur cas.

$$\|y - Xw\|^2 = \sum (y_i - \hat{y}_i)^2$$

Ridge : $\sum (y_i - \hat{y}_i)^2 + \lambda \|w\|^2 \Leftrightarrow \begin{cases} \text{minimiser } \sum (y_i - \hat{y}_i)^2 \\ \text{sous la contrainte } \lambda \|w\|^2 \leq t \end{cases}$

LASSO $\sum (y_i - \hat{y}_i)^2 + \lambda \sum_j |w_j| \Leftrightarrow \begin{cases} \text{minimiser } \sum (y_i - \hat{y}_i)^2 \\ \text{sous la contrainte } \lambda \sum_j |w_j| \leq t \end{cases}$

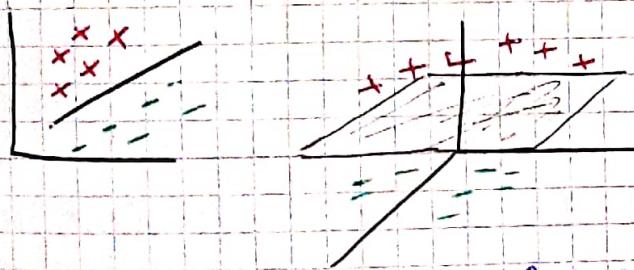
NB

on utilise la régularisation lorsqu'on pour ne pas avoir des w qui sont au plus grand. \Rightarrow pour ne pas avoir le surajustement.

Quand est ce que le Perceptron converge?

CNS \Rightarrow pourquoi l'algorithme de Perceptron converge

\Rightarrow les données doivent être linéairement séparables. (droite en 2D, plan en 3D, hyperplan ...)



\rightarrow lorsque les données ne sont pas linéairement séparables on utilise l'algorithme de Pocket.

cas où les données ne sont pas linéairement séparables:

transformer par une fonction φ l'espace de travail et réduire les x dans un

Espace où les données sont linéairement séparables.

au lieu de travail avec $x = (1, x_1, x_2, \dots, x_d)^t \in \mathbb{R}^{d+1}$

↳ les, log, log, log par ex

on travaille avec $\Phi(x)$

$$\Phi(x) = (1, \phi_1(x), \phi_2(x), \dots, \phi_{d+1}(x))^t \in \mathbb{R}^{d+1}$$

$d > d$ en général.

exemple:

$$x = (x_1, x_2) \rightarrow x = (1, x_1, x_2) \quad d=3$$

$$\Phi(x) = (1, x_1^2, 2x_1x_2, x_2^2) \quad d'=4$$

on travail dans un nouveau espace et les données deviennent linéairement séparables. On ne travaille plus avec les x avec des transpositions de x .

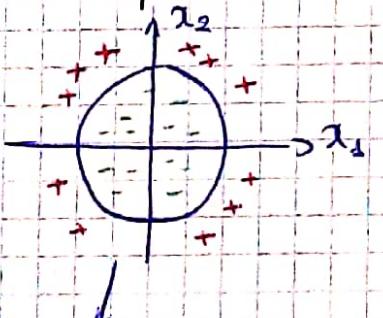
exemple:

$$x = (x_1, x_2) \in \mathbb{R}^2 \rightarrow \Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$$

⇒ les données deviennent linéairement séparables.

→ Il faut trouver une transformation qui permet dans un nouveau espace d'avoir des données linéairement séparables.

exemple: classification $d=3$



$$\begin{cases} x_1^2 + x_2^2 > 1 \rightarrow + \\ x_1^2 + x_2^2 < 1 \rightarrow - \end{cases}$$

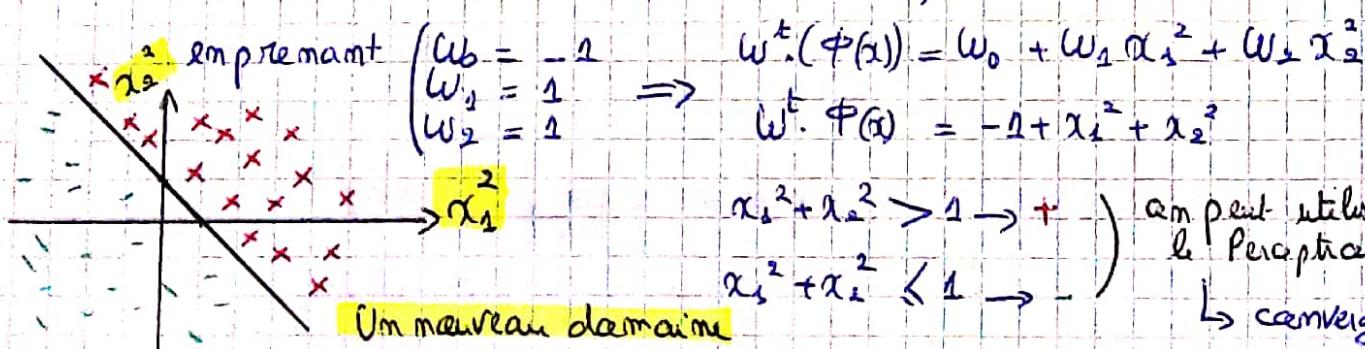
$$x = (1, x_1, x_2)^t$$

 $y \in \{+, -\}$

pas linéairement séparables.

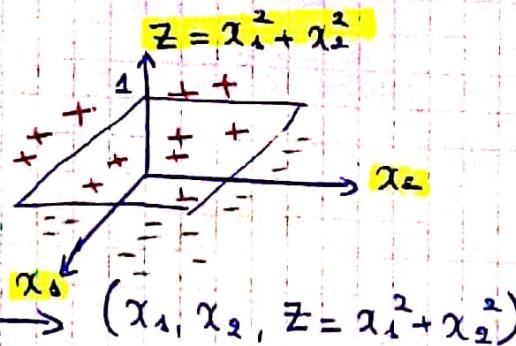
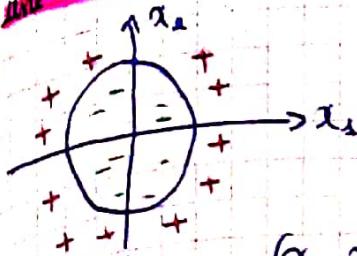
→ on cherche une transformation.

$$x = (1, x_1, x_2)^t \rightarrow \Phi(x) = (1, \phi_1(x), \phi_2(x)) = (1, x_1^2, x_2^2)$$



Un nouveau domaine

\Rightarrow on cherche un espace dans lequel les données seront linéairement séparables.
une autre transformation



dans ce nouveau espace les données sont linéairement séparables
(par un plan $x_1^2 + x_2^2 = 1$)

\rightarrow on a ajouté une dimension de plus et les données sont devenues lin. sep.

seance 7 : 17 - 05 - 2021

Chapitre 3 : Analyse en composantes principales ACP

\rightarrow on a un tableau :

\Rightarrow tableaux rectangulaires X de J variables quantitatives mesurées sur I individus : voici

x_i^j : la valeur de la variable j mesurée sur l'individu i

avec $i = 1, 2, \dots, I$ et $j = 1, 2, \dots, J$

- l'individu $x_i = (x_i^1, x_i^2, \dots, x_i^J)^t$: c'est un vecteur de J dimensions.

\Rightarrow est un point de l'espace R^J

- la variable $x_i^j = (x_1^j, x_2^j, \dots, x_I^j)^t$: c'est un vecteur de I dimensions.

\Rightarrow est un point de l'espace R^I

* si on s'intéresse à 1 variable le j est constant et les individus changent.

* si on s'intéresse à 1 individu le i est constant et les variables changent.

\Rightarrow donc on peut faire 2 analyses différentes : on peut analyser les individus ou bien analyser les variables.

→ Analyse directe :

- c'est une analyse des individus.

- Nuage $N(I)$: des I individus dans l'espace R^J .

→ Analyse dualiste :

- c'est l'analyse des variables.