Méthodologie en IA

Coucou les machines, cours mythique depuis deux ans mdr de SP/SN voici la méthodo à l'IA, ➔ ps il fait 22 pages mais c'est beaucoup d'explictaions je vous conseille par ailleurs de mettre la vidéo en même temps que vous lirez ma fiche ➔ des bisous et plein de courage et oubliez pas : force et honneur spartiate

Alors le prof commence par se présenter :
« Bonjour à tous, je me présente, je m'appelle David Chardin, je suis médecin nucléaire et je travaille sur le machine learning et son application en biologie et en médecine.
Je vais vous faire un cours sur des bases méthodologiques sur l'intelligence artificielle, le but étant que vous compreniez un petit peu ce qui se cache derrière le machine learning pour que vous puissiez comprendre dans quelles situations on va s'en servir pour résoudre certains problèmes. »

I. Définitions

L'intelligence artificielle : concept vaste ➔ regroupe l'ensemble des **théories** et des **techniques** qui sont mis en œuvre pour réaliser des machines qui sont capables de simuler l'intelligence humaine. (Larousse)

Ensuite l'automatisation, *c'est un concept que en général vous connaissez qui peut être rattaché à l'intelligence artificielle dans certains contextes, mais pas systématiquement.*
Elle concerne **l'exécution de tâches techniques par des machines qui fonctionneraient sans l'intervention humaine**,
exemple : l'automatisation dans la fabrication de voitures à tout ce qui est programme informatique où on automatise des tâches.

Le « machine learning » : domaine de l'intelligence artificielle qui concerne **l'automatisation de l'apprentissage par la machine**.

II. Qu'est ce qu'on veut dire par apprentissage par la machine ou machine learning ?

Au cours du temps il y a différentes définitions qui ont été proposées.
Une 1ère : par **Arthur Samuel** qui explique que le « machine learning c'est le domaine d'études qui permet aux machines d'apprendre sans être explicitement programmé par un humain. »

Ensuite une 2ème : par **Tom Mitchell** qui est plus technique et qui explique que si on considère une tache T, dont la performance est évaluée par P. Alors une machine apprend, quant au fur à mesure d'une expérience E, ses performances pour la tache T augmentent.

*Donc c'est une des choses qui va être **importante** en machine learning.*
Dans le cours du Dr. Imbert, vous avez vu qu'en intelligence artificielle, on va parler d'intelligence artificielle **model driven**, ce qui va au final plus se rapprocher de **l'automatisation**, où l'homme connaît le modèle qu'il veut appliquer, et d'intelligence

artificielle **data driven**, dont le **machine learning** va plus se rapprocher, car c'est bien **l'expérience** et donc les **data** qui permettent **d'améliorer la tâche T.**
Cependant ce n'est pas quelque chose qui est restreint, on peut en fait combiner un peu des deux, on peut **démarrer** une tâche sur un **modèle driven** et faire en sorte que les **data améliorent** cette tâche sur une idée du **datatrium**.

Une autre façon de représenter ceci et d'expliquer l'objectif qu'on a en machine learning, c'est de parler d'entrée, de fonction ou de tâche et de sortie.

En **entrée**, on va avoir de **l'information** ou de l'expérience => appelé **X**.
La **tâche**, ça va être **l'action** de la machine*, en général,* c'est une **fonction** => appelé f(x)
Et la **sortie**, ça va être par exemple un **label** ou une **classe** => appelé **Y**

Notre **objectif** : **créer un modèle ou une fonction f qui permette d'obtenir Y quand on l'applique à X => trouver f(x) tel que f(x) = y**

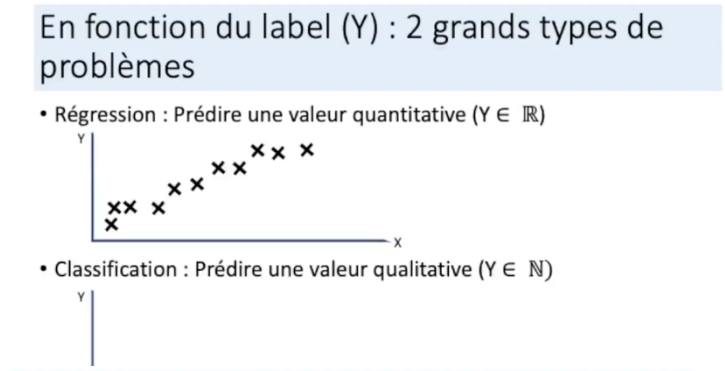
Et ça c'est une notion importante de comprendre déjà si on vous donne des données qu'est-ce qui correspond à une entrée X qu'est-ce qui correspond à une sortie Y un objectif et ensuite comprendre que le but c'est de créer une fonction f qui au final sera plus ou moins complexe ensemble on va voir des fonctions simples mais ça peut devenir beaucoup plus compliqué, mais on ne rentrera pas trop dans les détails par rapport à ça.

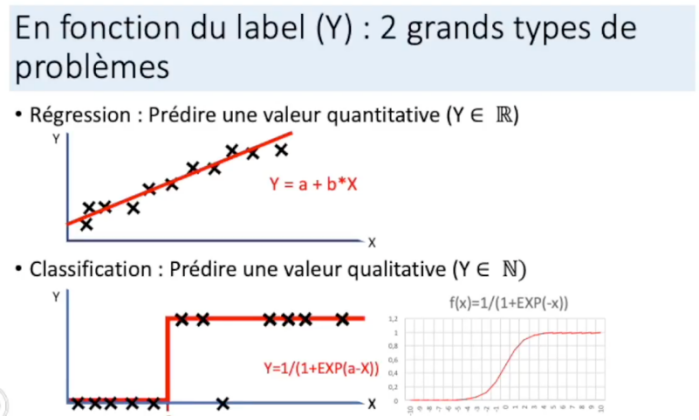
Exemple : un tableau de données qui représente des quantités de différentes molécules chez différents patients qui présente des caractéristiques différentes.
1 ère ligne = numéro de patient (osf) on ne cherche pas à le prédire
2 -ème ligne = label soit égale à 2 soit égale à 1 = > traduit une condition pour chaque patient
Exemple ici : label = 1 => tumeur bénigne et label = 2 => tumeur maligne
Reste des lignes = différentes molécules et pour chaque mol on a des valeurs qui correspondent à la quantité de molécules pour chaque patient.

Dans cet exemple si on veut entrainer une machine à déterminer si un patient présente une tumeur maligne ou bénigne en fonction des quantités de chaque molécule, on va désigner par X toutes les qtés de molécules et donc Y sera notre label
1 = bénigne ou 2 = maligne => on a bien une entrée X et un objectif Y et ce qu'on veut créer *(absent dans notre tableau)* = > fonction qui permet d'obtenir Y à partir de X.

En fonction du label (Y) on a 2 grands types de problèmes.
Si Y appartient à une valeur **quantitative** (Y *appartient* au réel € **R**) => logik mais bon : la fonction qu'on recherche pourra donner comme résultat n'importe quel nombre réel
→ Ex : prédire le nombre de patients atteints du COVID-19 après 1 mois de pandémie.
Ex : prédire le prix de vente d'un bien immobilier.

Si Y est une valeur **qualitative** : c'est donc une catégorie qui pourra être transformé en 1,2… entiers = > sortie peut prendre un nombre **limité** de valeurs.
Ex : prédire si un patient malade aura une forme grave ou non de la maladie.
On a bien deux catégories qu'on sépare en sévère ou non.
On peut si on veut rajouter une troisième catégorie en disant ça c'est la forme asymptomatique. Dans TOUT les cas, y'aura que deux ou trois sorties possibles.





Régression *: On veut prédire y à partir d'une seule variable x → prédire le prix d'une maison en fonction de la surface en m^2 => les prix (X) en fonction de la surface (Y) et inversement si on veut définir le prix à partir de la surface au m^2 il nous faut une fonction f qui relie surface au prix.*
Y= a+b*X = > régression linéaire
Classification : *on peut vouloir prédire le risque de survenue d'un cancer en fonction de la durée d'exposition aux radiations. Donc là la sortie Y : il y a deux types de possibilités. Le patient a eu un cancer = 1. Le patient n'a pas eu de cancer = 0. (fait un aparté sur Tchernobyl et dit qu'on dirai qu'il y a un seuil à partir duquel les patients ont eu un cancer. on veut savoir maintenant, on a un patient, on sait combien de temps il a été exposé et on veut savoir si oui ou non il a un cancer, on va utiliser une fonction donc la fonction de l'exemple c'est une fonction en S : sigmoïde c'est une fonction logistique, donc elle sera bornée, elle ne pourra que donner des valeurs entre 0 et 1 . et donc on pourra décider que toutes les valeurs en dessous de 0,5 seront passés à zéro et au-dessus 0,5 à un .*

Apprentissage supervisé : Le **plus fréquent** en médecine
- On **connait** les **labels** / les valeurs qu'on veut prédire avec notre modèle. = quand on connait l'objectif qu'on veut atteindre.

Exemples :
- Dépistage de cancer : Cancer vs Pas de cancer,
- Prédiction du prix de l'immobilier,
- Triage de mails : SPAM ou non SPAM.

Apprentissage non supervisé :
- On veut trouver des tendances / des groupes, mais on ne sait pas à l'avance lesquels

Dans la majeure partie des cas ce sera des problèmes de classifications non supervisé
Exemples :
- Regroupement de sites web similaires pour faire des suggestions,
- Recherche de sous-groupes de tumeurs.

Apprentissage semi-supervisé : qu'on ne va pas développer dans ce cours qui concerne des bases de données, pour lesquelles seul une partie des labels à prédire sont connus.

III. Comment apprendre ?

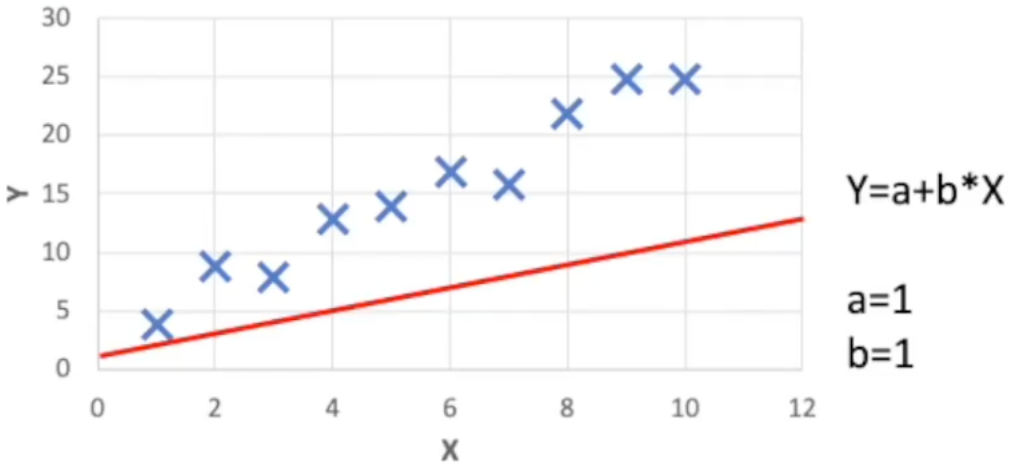
En principe, on rappelle que le but = apprendre à effectuer une tâche et que pour savoir si on effectue correctement cette tâche on va **mesurer** une performance et ce qu'il faut c'est qu'au fur et à mesure qu'on apprenne on devienne de plus en plus performant.

C'est une notion **très générale** pas spécifique aux machines, même pour nous au fur et à mesure qu'on acquiert de l'expérience on devient de plus en plus performant pour certaines tâches.

Comment est-ce qu'on sait qu'on devient plus performant ?

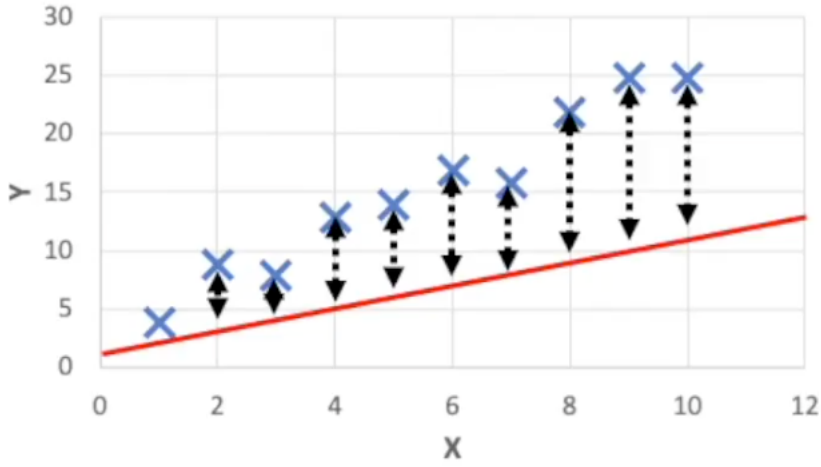
Au fur et à mesure qu'on apprend on se rapproche d'un objectif final ou autrement dit on fait de moins en moins d'erreur et si pour nous c'est quelque chose d'assez intuitif.
En machine learning il va falloir programmer la machine pour qu'elle minimise l'erreur au fur et à mesure de l'expérience et il va falloir définir ce qu'est l'erreur.
Pour définir cette erreur on va utiliser ce qu'on appelle la fonction de coût ou fonction de perte qui va représenter globalement la différence entre ce qu'on a obtenu et ce que l'on souhaite obtenir comme résultat.
On va voir ensemble qu'il existe des algorithmes qui permettent de minimiser petit à petit cette erreur.

Pour illustrer cette fonction de coût on va reprendre notre régression linéaire de l'exemple précédent et on va voir comment on arrive à cette droite qui représente donc la régression linéaire avec cette fonction Y = a + b*X.
Donc on va repartir des données avec Y en fonction de X et on sait que la forme de la droite qu'on a envie de fabriquer est de type Y = a + b*X et l'objectif ça va être de trouver a et b.
Au départ on connaît ni a ni b donc on va faire un premier test par exemple en prenant a = 1 et b = 1 ça va nous donner cette courbe :



Ce qui on le voit n'est pas la courbe qu'on attend une fois qu'on a cette courbe il va falloir mesurer l'erreur qu'on a engendré en utilisant cette courbe donc pour ce faire on va

utiliser la formule ci-dessous



$$\text{Erreur: } J(a,b)=\frac{1}{2m}\sum_{i=1}^{m}(h(x_i)-Y)^2$$

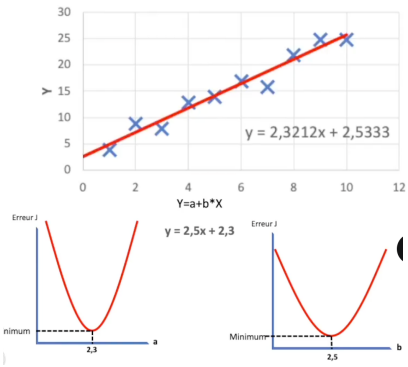
Y predit Y objectif

que vous n'avez pas besoin de connaître par cœur et pour laquelle il faut comprendre que ça représente la moyenne de la différence absolue entre le résultat de notre courbe et le résultat qu'on attend .

Alors si on s'intéresse à la formule en réalité (mais mdr elle était pas censé être inintéressante pour nous) c'est pas strictement une moyenne mais la moyenne des carrés des différences pour chaque point divisé par deux mais on utilise cette formule uniquement pour faciliter les calculs (faciliter quoi ???) et la programmation qu'il y a derrière et vous n'avez pas besoin de la connaître par cœur (mais big lol répete le bien et dit le vite fait).

Ce qu'il faut comprendre par contre c'est que si la courbe s'éloigne du résultat qu'on attend par exemple ici si on diminue a en le passant à 0 cette erreur elle va augmenter alors que si la courbe se rapproche du résultat attendu par exemple en passant a à 2 l'erreur va diminuer ce qui est donc notre objectif et quand on va écrire notre algorithme on va pouvoir faire en sorte que l'erreur continue à diminuer jusqu'à ce qu'elle ne puisse plus diminuer jusqu'à ce qu'on arrive à un minimum.

Tut'l'explique : si tu augmentes ton erreur tu t'éloignes de la courbe et si tu diminues l'erreur tu t'en rapproche notre objectif c'est de diminuer un max l'erreur pour arriver à un minimum d'erreur.



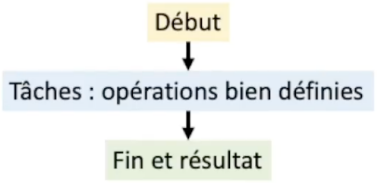
Ainsi petit à petit on va pouvoir trouver les meilleures paramètres et à partir de là la fonction de coût (mdr c'est moi qui vait me mettre un coup) ne pourra plus diminuer elle sera au minimum possible et on va pouvoir tracer l'erreur associé au paramètre a en fonction du paramètre a et l'erreur associé au paramètre b en fonction du paramètre b et pour arriver à une erreur minimale pour a et b on va utiliser un **algorithme de base** en **machine learning** qui s'appelle la d**escente de gradients**

IV. Qu'est-ce que la descente de gradients (spoiler alert : c'est pas la mienne même si ce cours me pousse à bout)

Def : La descente de gradients c'est un algorithme

V. Qu'est-ce que c'est un algorithme ?

Un algorithme = « un ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations »



C'est quelque chose qui a un début qui consiste en l'ensemble d'opération bien définie et qui aboutit à un résultat donc au final ça peut s'appliquer à beaucoup de situations.

Par exemple : si on cherche un nom dans une liste mal classé on va sûrement appliquer un algorithme qui sera très simple on va lire le premier nom ce sera la première tâche élémentaire ensuite on va suivre une règle si c'est le nom qu'on cherche on s'arrête sinon on passe à la ligne suivante et on continue comme ça jusqu'à ce qu'on s'arrête.

On lit le premier nom ce n'est pas le bon on lit le deuxième nom ce n'est pas bon on lit le troisième nom c'est celui qu'on cherche on s'arrête.
(moi je l'ai toujours dit médecine c'est durrrr lachez pas l'école)

En médecine on va aussi suivre des algorithmes notamment des choses qu'on appelle des algorithmes décisionnels ou arbres décisionnels pour lesquels on sera dans la même situation on va partir d'une situation initiale et on va suivre un ensemble de règles jusqu'à aboutir à un résultat.

Par exemple (décidemment) : une décision thérapeutique ou un diagnostic

La **descente de gradient** = **algorithme** donc il doit aussi être composé d'opérations bien définie.
Mais quels sont ces opérations ? En réalité ça peut se résumer à ce qu'on a ici si on prend une fonction qui a différents paramètres par exemple la fonction simple de tout à l'heure (celle-là l l'a kiffee) Y = a + b*X, la descente de gradient va démarrer en prenant des paramètres arbitraires (ex : a = 0 et b = 0) puis elle va les mettre à jour suivant la formule et

$$\theta := \theta - \alpha * \frac{\partial}{\partial \theta} J(\theta_0, \theta_1 \ldots)$$

ceci jusqu'à ce qu'on atteigne la convergence ce qui veut dire que les paramètres ne bougeront plus au fur et à mesure qu'on les met à jour.

On va s'intéresser à cette formule vous n'avez pas besoin de la connaître par cœur (mdr c'est la deuxième à pas connaitre par cœur il s'agirait de doser).
Par contre on va essayer de comprendre ce que ça veut dire donc déjà ce symbole deux points égale = met à jour le paramètre au fur et à mesure donc le nouveau paramètre à gauche va devenir ce qu'on a à droite.

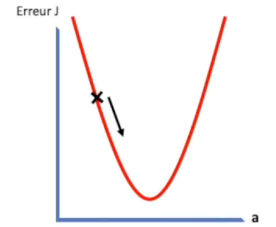
Ensuite les paramètres (thêta) qui sont une notation générale pour les paramètres de notre fonction on utilise en général le signe thêta.
Sur la droite le thêta 0 et thêta 1 etc correspond à différents paramètres de la fonction.
Dans notre exp : on pourrait remplacer si on travaille sur a comme ceci donc on met à jour a et à droite on s'intéresse à la fonction de coup qui prend en compte à la fois a et b

$$a := a - \alpha * \frac{\partial}{\partial a} J(a, b)$$

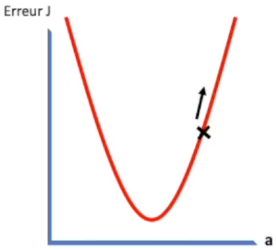
Le paramètre alpha correspond au learning rate ou taux d'apprentissage (on le revoie après)

A droite on a la dérivée de la fonction de coup en fonction du paramètre qu'on est en train de mettre à jour.

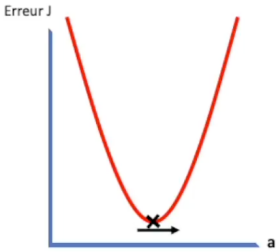
Rappel : la dérivée d'une fonction c'est la pente de la courbe à un endroit donné.



Si on se place sur la première partie de notre fonction de perte ici : si a augmente la valeur de j diminue et donc la courbe descend donc notre dérivé va nous donner une valeur négative.



Par contre si on se place plus loin sur la courbe alors quand a augmente j augmente également donc la pente est croissante donc notre dérivé va nous donner une valeur positive.

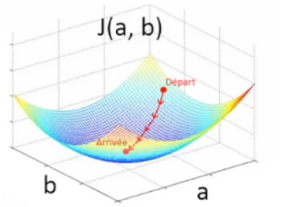


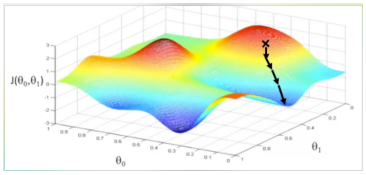
Enfin, si on est au plus bas de notre courbe à l'endroit où elle passe d'une pente descendante à une pente croissante on va être sur un point où la dérivée est égale à 0 et ce sera notre minimum local.

Donc si on reprend notre formule on voit qu'avant notre dérivé on a un signe moins et donc on comprend bien que si augmenter la valeur de notre paramètre diminue l'erreur notre dérivé va être négative et donc appliquer notre formule reviendra à augmenter un petit peu la valeur de notre paramètre :

$$\theta := \theta - \alpha * \frac{\partial}{\partial \theta} J(\theta_0, \theta_1 \ldots)$$

Si au contraire augmenter le paramètre augmente l'erreur alors dérivé va être positive et donc appliquer la formule va entraîner de diminuer la valeur du paramètre.
Enfin si on est arrivé sur un minimum local notre dérivé va être nul et donc notre paramètre ne changera pas de valeur et une fois que ce sera le cas pour tous nos paramètres les opérations sont terminées et l'algorithme nous aura donné les paramètres qui permettent d'avoir une erreur minimale.
Si vous avez compris ça vous avez compris le principe général de l'algorithme de descente de gradients et ce qu'il faut bien comprendre c'est que cet algorithme on peut l'utiliser pour un grand nombre de fonctions et pas seulement pour la régression linéaire ça va même au-delà du machine learning parce que ça peut s'appliquer à d'autres types de problèmes.
Notamment quand on fait de la reconstruction d'image.
Ce qu'il faut comprendre aussi c'est que si dans notre exemple de régression linéaire la fonction de coût présentait un minimum unique pour des valeurs uniques de a et b et donc prenez la forme qu'on a ici



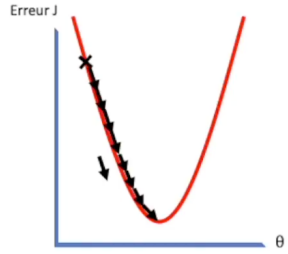


Dans d'autres situations il peut exister plusieurs minimums locaux possibles et donc en fonction des paramètres qu'on va utiliser au départ le résultat ne sera pas forcément le même.

En effet si on démarre notre algorithme à un endroit il va nous mener à ce minimum ici alors que si on démarre un autre endroit notre algorithme va nous mener à un autre minimum local et donc il n'existe pas toujours une solution unique à nos problèmes.
Pour imager un petit peu ce que fait la descente de gradient imaginez que vous soyez sur la croix qu'on a sur ce graphique et que le graphique soit une colline vous avez les yeux bandés et on vous demande de descendre de la colline pour aller le plus bas possible vous êtes debout comment vous allez faire pour descendre vous allez tâtonner de tous les côtés pour essayer de sentir le terrain pour essayer de trouver un endroit où la pente descend une fois que vous avez trouvé l'endroit où la pente descend le plus là vous allez faire un pas et vous allez donc changer de position ce que vous venez de faire c'est exactement ce que fait la descente de gradient à chaque itération et donc vous allez continuer jusqu'à ce que vous soyez sur un endroit où vous retrouvez à plat à partir de là vous allez considérer que vous êtes plus sur la colline et vous allez vous arrêter . Vous avez réussi à faire ce que vous avez demandé donc si vous avez compris ça vous avez compris comment fonctionne la descente de gradient.

VI. Quelques subtilités

Tout d'abord on va revenir sur le paramètre alpha de notre formule qui correspond donc au taux d'apprentissage => un facteur qui multiplie la dérive et donc si ce paramètre est petit quand on met à jour théta il va peu (pas beaucoup) être modifier alors que si on prend un paramètre alpha élevé quand on met à jour théta la variation va être plus importante.

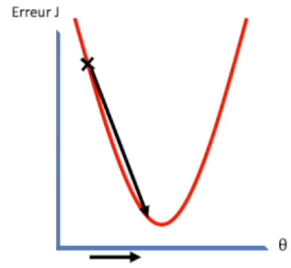


- $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 \ldots)$

α trop petit : petites variations de θ

Beaucoup de calculs pour arriver au minimum

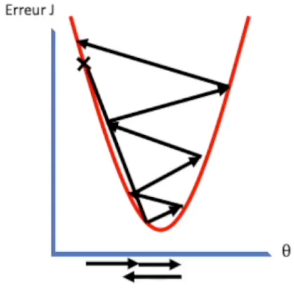
Cela peut avoir une influence sur l'algorithme si on reprend notre fonction de coup avec l'erreur en fonction du paramètre théta et qu'on se place là où j'ai (mis la croix si on prend un alpha très petit alors à chaque mise à jour on va avoir une petite variation du théta et donc on va arriver à notre minimum local mais ça va prendre beaucoup de mises à jour donc beaucoup de calcul.



- $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 \ldots)$

α trop grand: grandes variations de θ

Ensuite : Qu'est ce qui se passe si on prend un taux d'apprentissage trop élevé ?
Imaginons on a pris un alpha assez élevé on fait un premier calcul et notre valeur de théta sera proche de notre minimum pour l'instant tout va bien sauf qu'à la première chaîne mise à jour notre pente est toujours descendante et donc on va encore augmenter notre théta et donc à la prochaine mise à jour on risque de dépasser et même de s'éloigner du seuil minimal qu'on recherche et donc on arrive sur une partie de la courbe qui est ascendante donc avec notre formule on va retourner en arrière sauf qu'en plus comme on s'est éloigné par rapport à précédent mise à jour notre pente elle est plus forte et donc qu'est ce qui va se passer on va revenir en arrière encore plus loin qu'à l'itération précédente et ceci ainsi de suite et donc petit à petit en fait on va s'éloigner de plus en plus de notre objectif qui est ce minimum local .



- $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 \ldots)$

α trop grand: grandes variations de θ

Risque : pas de convergence voir même divergence

Et donc le risque d'avoir cet alpha trop élevé c'est que notre algorithme ne converge pas voire pire il peut diverger et s'éloigner de notre objectif.
Alpha est une valeur fixe pour converger effectivement il va falloir que les modifications apportées à teta au fur et à mesure qu'on se rapproche du minimum soit de plus en plus

fines mais ça peut se faire sans avoir à modifier alpha car c'est le paramètre de la dérivée de j et donc la pente de la courbe qui va diminuer petit à petit au fur et à mesure qu'on se rapproche de ce minimum et donc même avec un alpha constant les modifications qu'on va apporter vont être de plus en plus petites au fur et à mesure qu'on se rapproche de notre objectif donc si on résume concernant le taux d'apprentissage si on prend un taux d'apprentissage trop petit il va falloir faire un nombre important d'itération pour obtenir notre résultat notre minimum si par contre on prend un taux d'apprentissage trop élevé on risque de ne jamais converger voire de diverger et donc notre objectif ça va être de trouver un taux d'apprentissage qui nous permet de converger sans utiliser trop de calcul donc ça s'adapte en fonction des situations on peut faire un premier essai avec un temps un taux d'apprentissage défini si on se rend compte que l'algorithme diverge on diminue ce taux d'apprentissage si on trouve que l'algorithme prend énormément de temps on essaye avec un taux d'apprentissage un petit peu plus élevé

- Trop petit : long temps de calcul.

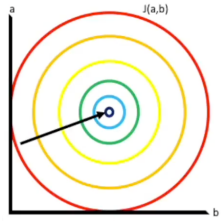
- Trop grand : risque de divergence.

- Objectif : le plus grand possible tant que ça converge.

VII. Le scaling des données ou la mise à l'échelle

C'est quelque chose qui concerne les données d'entrée qu'on va utiliser donc x je rappelle que on peut utiliser la descente de gradient pour minimiser la fonction de perte associée à tout un tas de fonctions qui peuvent être plus ou moins complexes et donc maintenant on va se mettre dans la situation où on essaye de minimiser la fonction de coût associé à la fonction : y = k (constante)+ a*x1 a étant le paramètre x1 étant notre première donnée +b x2 b étant un deuxième paramètre et x2 une deuxième donnée.
Si x1 et x2 ont la même dimension par exemple s'il s'agit pour les deux cas de dosage biologique pour lesquels les valeurs sont comprises entre 10 et 100 millimoles par litre alors la fonction de perte associée au paramètre a et b va prendre l'aspect qu'on avait tout à l'heure (5-ème graphique en partant du bas) et la descente de gradient va fonctionner correctement .

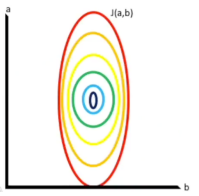
Si on représente cette fonction de perte sur une vue 2d donc ici vue du haut pour le graphique précédent ça va nous donner cette forme de cible et dans ce cas-là l'algorithme va rapidement trouver le minimum local. On peut imaginer que c'est comme si on laissait tomber une bille depuis le rebord d'un saladier celle-ci irait alors rapidement vers le point le plus bas du saladier.



$Y = k + a \cdot x_1 + b \cdot x_2$

x_1 et x_2 ont des dimensions similaires :
Le minimum est trouvé rapidement

Maintenant si nos données x1 et x2 présente des dimensions très différentes par exemple si x2 est beaucoup plus grand que x1. On peut imaginer par exemple faire un modèle pour prédire un événement cardiovasculaire chez des patients en fonction de leur glycémie agent et de leur âge alors nos données âge vont avoir une dimension beaucoup plus importante que nos données glycémie à jeun. Ici on imagine par exemple que dans nos données la glycémie à jeun varie entre 0,8 et 1,8 gramme par litre et nos patients ont entre 30 et 85 ans.

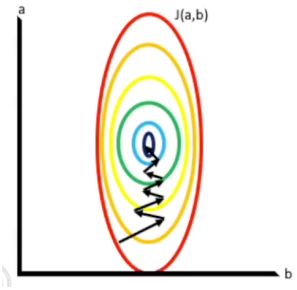


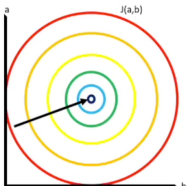
$Y = k + a \cdot x_1 + b \cdot x_2$

Ex. x_1 : glycémie à jeun (0,8 – 1,8)
x_2 : âge (30 – 85)

x_2 a une dimension beaucoup plus grande que x_1 :
Le minimum est beaucoup plus long à trouver

Si on représente la fonction de coût associé au paramètre A et B en fonction de ces paramètres on va voir que celle-ci va prendre une forme complètement différente comme représenté.
On voit que dans la zone verticale centrale de la figure la pente menant au minimum sera beaucoup plus faible et donc si l'algorithme se retrouve dans cette zone là à un moment il va mettre beaucoup plus de temps pour trouver notre minimum donc ce qu'on va vouloir faire c'est modifier nos données d'entrée donc x1 et x2 pour faire en sorte qu'elles aient une dimension similaire et donc retrouver une forme plus proche de celle qu'on avait au début donc ➔ et pour ça ce qu'on va faire c'est qu'on va faire un scaling des données ou une mise à l'échelle on peut aussi dire en français : on va centrer et réduire les données .





Donc ça, qu'est-ce que ça veut dire ?
Si on reprend nos données x1 et x2, le centrage va avoir pour but que la moyenne de x1 et la moyenne de x2 soient la même, et pour ça, on va mettre les moyennes de x1 et x2 à 0.
Pour ça, c'est simple, on va prendre les données de x1 et on va soustraire la moyenne de x1 à chacune des données.
Donc, notre nouvelle moyenne sera égale à l'ancienne moyenne, moins celle-ci, donc zéro.

Et ensuite, la réduction, ça va consister à faire en sorte que les valeurs de x1 et x2 soient réparties sur la même fourchette, et donc la même dimension.
Et pour ça, on va diviser chacune des valeurs pour nos deux variables, par une valeur qui représente la dispersion de nos variables.

En mathématique, il existe différentes mesures de cette dispersion, et donc on peut réduire en utilisant ces différentes mesures, on peut diviser par la variance, par l'écartype ou par l'intervalle de distribution entier de notre variable, qu'on appelle en anglais : range.

Et donc, la combinaison du centrage et de la réduction va nous donner la formule qu'on a à droite, où x_i représente une valeur pour une variable, μ_i représente la moyenne de cette variable et S_i représente l'écartip, la variance ou le range de cette variable.
Le centrage et la réduction, c'est quelque chose qui est utilisé en statistiques dans de nombreuses situations, pas seulement dans le cadre de la descente de gradient, et il faut savoir en quoi ça consiste.

Si on reprend nos exemples de données, d'abord la glycémie à jeun, donc initialement on a des données qui varient entre 0,8 et 1,8, avec une moyenne qu'on donne à 1.
Donc si on fait un centrage et une réduction en utilisant le range qui sera donc ici de 1 : 1,8-0,8, on obtiendra des nouvelles valeurs pour nos données qui varieront entre -0,2 et 0,8.
Donc pour la glycémie, on n'a pas eu une grosse variation de la dimension des données après centrage et réduction.

Maintenant si on prend l'âge qui varie ici de 30 ans à 85 ans, avec une moyenne qui est de 60 ans, si on refait la même chose, cette fois notre range c'est 85 - 30 donc 55, et donc après centrage et réduction, les nouvelles valeurs que prendront la variable âge vont aller de -0,54 à 0,45.
Et donc pour la variable âge, le scaling a vraiment diminué la dimension et a fait en sorte que maintenant nos deux variables, la glycémie à jeun et l'âge, ont des dimensions similaires.
Et ça, ça va faciliter la réalisation de la descente de gradient.

VIII. Pourquoi il y a un intérêt à créer des modèles qui prennent en compte plusieurs variables explicatives ? (Point important du machine learning)

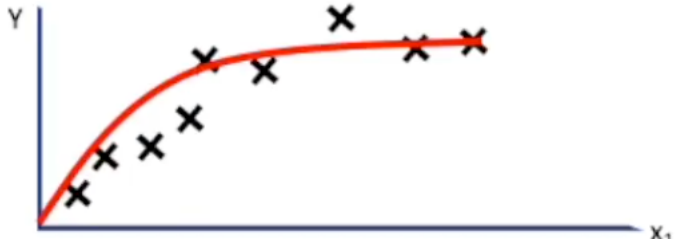
Donc pour ça on va reprendre notre distribution de données avec une variable Y exprimée en fonction d'une variable explicative X1. On peut reprendre l'exemple de la prédiction du prix de maison fonction de leur surface en mètre carré.

Donc on a vu tout à l'heure La régression linéaire à une seule variable et ça nous a servi à expliquer un petit peu comment fonctionne la descente de gradient.
On a donc vu comment on peut utiliser cette descente de gradient pour trouver les paramètres Thêta0 et thêta1 tel qu'on les a ici pour tracer la droite de régression qu'on a en rouge ici qui permettrait de prédire le prix de la maison en fonction de la surface en mètre carré.

$$Y=\theta_0 + \theta_1 x_1$$

Cependant, on voit que ce modèle est imparfait et en effet, on sait que le prix d'une maison ne va pas simplement se résumer à cette surface. ➔Donc il va falloir affiner notre modèle.

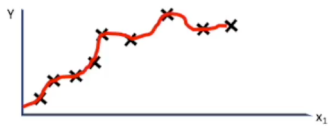
Pour ce faire, on va pouvoir par exemple introduire de nouvelles variables explicatives.
Par exemple, on peut intégrer une variable x2 qui pourrait par exemple représenter la distance du bien immobilier par rapport au centre-ville ou la surface d'une éventuelle terrasse et pour cette nouvelle variable x2, il va falloir trouver un nouveau paramètre thêta2.



$$Y=\theta_0 + \theta_1 x_1 + \theta_2 {x_1}^2$$

Pour affiner le modèle, on peut aussi choisir de prendre en compte des variables supplémentaires qui correspondraient par exemple à X1 porté au carré ou porté au cube ou des racines carrées de X1 et on ferait à ce moment-là plus de la régression linéaire mais de la régression polynomiale *qu'on n'abordera pas en détail.*

Mais ce qu'il faut comprendre c'est qu'intégrer de **nouvelles variables** à notre modèle, ça peut permettre de **l'affiner** et donc d'en **augmenter** la pertinence.



$$Y=\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1 \cdot x_2 + \dots$$

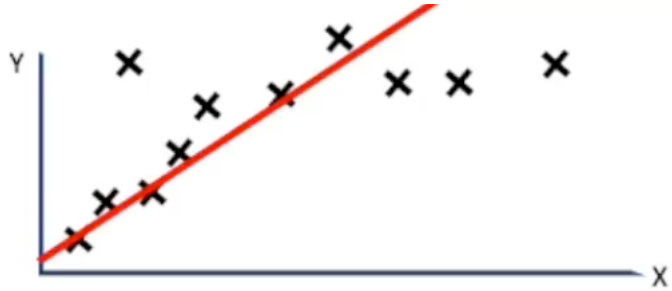
Modèle basé sur un grand nombre de données $x_1, x_2, x_3 \dots$

On peut créer de nouvelles données en combinant certaines variables :

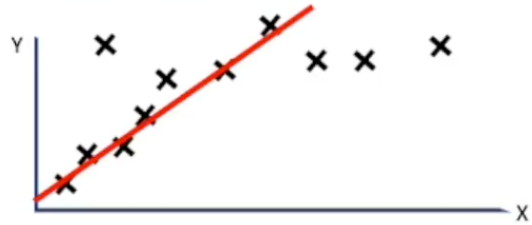
Ex : $x_3 = x_1 \cdot x_2$

Et ainsi, si on est encore plus loin qu'on intègre d'autant plus de données que ce soit de nouvelles variables explicatives ou des dérivés des variables explicatives qu'on a déjà, on peut arriver à un modèle tellement complexe qu'il passera par l'ensemble des points de nos données existantes.
Cependant, on va voir qu'avoir un modèle **trop complexe, c'est parfois délétère.**

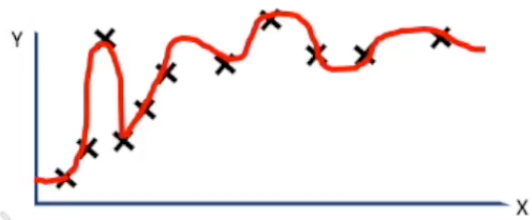
Si on reprend un jeu de données avec Y le prix de nos maisons et X la surface en mètre carré, avec cette fois-ci des variations un peu plus importantes.



On peut encore une fois utiliser la régression linéaire à une seule variable, ce qui va nous donner cette courbe qui comme on l'a vu est imparfaite et ne prend pas en compte d'autres paramètres, comme par exemple la présence d'une terrasse et donc même si la descente de dernière nous a permis de trouver le prix de la maison en fonction de la surface en mètre carré, l'erreur du modèle va rester significative.
Plusieurs données vont être assez éloignées de notre modèle.
Le modèle est trop simple pour représenter la distribution de nos données, donc il (ne colle pas assez à nos données et on va dire qu'il y a **underfitting**.
Et donc si on veut réutiliser ce modèle, ce qui reste quand même l'objectif en machine learning pour l'appliquer à de nouvelles données, par exemple si on a une nouvelle maison qui a une surface assez importante, l'application de notre modèle va nous donner un prix qui sera beaucoup plus élevé que la réalité et donc on n'arrivera pas à vendre cette maison.
Donc on ne va pas pouvoir utiliser correctement ce modèle.

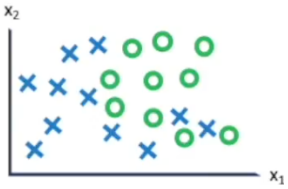


Modèle trop simple : « underfitting »
Faibles performances sur nos données d'entrainement,
Non applicable



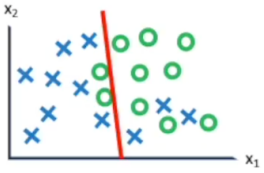
Modèle trop compliqué : « overfitting »
Très bonnes performances sur nos données d'entrainement,
Mais non applicable

Maintenant si au contraire on utilise un modèle très complexe, on peut arriver à une erreur qui est quasi nul et qu'un modèle qui au final colle parfaitement à nos données, même celle qui correspond au final à des exceptions. Et donc encore une fois si on veut réutiliser ce modèle, par exemple pour une maison qui aurait une surface comme ici, on risque de prédire un prix de vente qui encore une fois est trop éloigné de la réalité et on ne pourra donc pas utiliser ce modèle correctement. Et donc dans cette situation où le modèle colle trop bien à nos données d'entraînement, même à toutes les petites exceptions, ce qui le rend **inutilisable** pour de nouvelles données, eh bien on dirait qu'il y a **overfitting**.



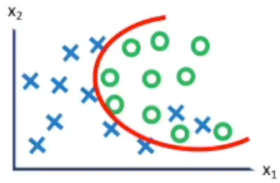
Ensuite si on prend un **problème** de **classification**, donc ici un problème avec deux classes représentées par des ronds et des croix, dont on a représenté la distribution en fonction de deux variables X1 et X2, on pourra avoir le même type de problème.

UNDERFITTING



En effet, comme on l'a vu, le but va être de séparer ces deux classes et pour ce faire, on peut utiliser une **simple droite** comme ici, où on considérera que tout ce qui est à gauche de la droite appartiendra à une première classe, donc ici les croix, et tout ce qui est à droite de la droite appartiendra à la classe des ronds.
Sauf qu'avec ce modèle simple, plusieurs de nos individus vont être **mal** classés.

BON INTERMÉDIAIRE



Donc on peut avoir envie d'utiliser un modèle plus complexe qui prend en compte d'autres variables.
Par exemple, un modèle comme celui-ci qui va mieux séparer nos deux classes, mais qui restera imparfait.

OVERFITTTING

Et si on complique encore plus le modèle, on peut arriver à quelque chose comme ça, où tous nos individus font partie de la bonne classe. Cependant, encore une fois, si on veut réutiliser ce modèle sur de nouvelles données, il risque de moins bien marcher que notre modèle intermédiaire.

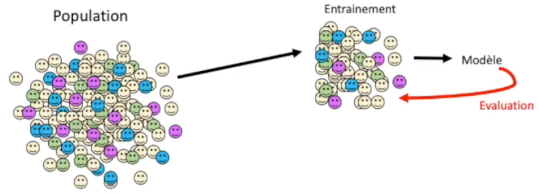
Donc comme pour la **régression**, on peut avoir de **l'underfitting** si notre modèle est trop simple, de l'**overfitting** si il est trop complexe, et ce qu'il va falloir trouver, c'est un **bon intermédiaire** qui va suffisamment bien séparer nos données d'entraînement tout en étant applicable pour de nouvelles données.
Donc **l'underfitting**, c'est le fait de créer des modèles qui sont trop simples pour expliquer nos données.
Et la solution à ça, ça va être de prendre en compte plus de variables explicatives.
Si on reprend notre exemple où on veut prédire le prix des maisons, si on se rend compte que l'utilisation de la surface en mètres carrés ne suffit pas à avoir un modèle pour correctement prédire le prix de la maison, on peut rajouter des variables comme la présence d'une terrasse ou la présence d'une piscine.

D'autres choses qu'on sait peuvent influencer le prix de notre bien.

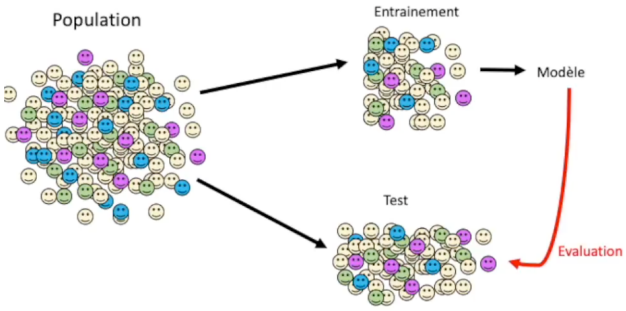
Mais il faudra garder en tête que rajouter des données, ça rajoute aussi un risque **d'overfitting**. Et pour limiter ce risque d'overfitting, il faudra tout de même limiter le nombre de variables explicatives qu'on utilise.
Ceci peut se faire à la main, donc par l'utilisateur.
Donc si on reprend l'exemple de la vente immobilière, l'utilisateur pourra choisir de ne pas prendre par exemple la couleur des tuiles comme variable explicative car a priori ce n'est pas quelque chose qui devrait influencer le prix de la maison.
Et l'utilisateur ne gardera que des variables qui sont pertinentes.
Cependant dans d'autres applications, on ne sait pas forcément à l'avance qu'est-ce qui est pertinent.
Notamment en biologie ou en médecine, on peut travailler avec des données type omiques comme des données de génétique ou de génomique où on a énormément de variables et à l'avance on ne sait pas forcément lesquelles vont être pertinentes. On pourra dans ce cas-là utiliser des méthodes qui permettent de réduire le nombre de données qu'on prend en compte en essayant d'éliminer les variables qui apporte une information redondante. On ne verra pas en détail comment ça fonctionne mais cela existe.
Une alternative si on veut garder toutes nos variables c'est d'utiliser une méthode qui s'appelle la régularisation. Cette méthode fonctionne en **limitant la magnitude des valeurs** que peuvent prendre les paramètres theta ce qui va permettre de limiter la complexité des modèles qu'on entraînera mais c'est un peu plus compliqué et on n'entrera pas là-dedans en détail. Enfin ce qu'il faut comprendre c'est que si l'underfitting et l'overfitting peuvent limiter la qualité des modèles qu'on crée et leur utilisation une autre chose qui est très importante c'est de travailler avec des données qui sont de **bonne** qualité.
En effet quand on entraîne un modèle celui-ci se base sur les données qu'on a entrées et donc si ces données contiennent des erreurs notre modèle va être basé sur des erreurs et donc si on veut l'appliquer à des nouvelles données il pourra se tromper.

Les algorithmes d'apprentissage comme la descente des gradients ne prennent pas en compte cette possibilité qu'il existe des erreurs ils prennent les données qu'on leur donne comme des **vérités absolues**.

Une autre chose qui va être importante quand on fait de l'apprentissage supervisé et qui se recoupe avec cette notion de qualité de données ça va être notre échantillonnage pour entraîner le modèle.
Donc notre objectif en machine learning ça va être de créer un modèle à partir d'un échantillon d'une population pour ensuite pouvoir utiliser ce modèle pour l'appliquer l'ensemble de la population et donc on va retrouver les mêmes problèmes d'échantillonnage que pour des tests statistiques descriptifs.



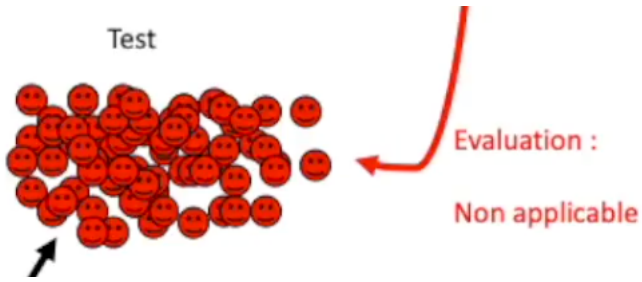
Donc à partir d'une population on va extraire une cohorte ou un échantillon qui va nous servir pour entraîner le modèle et on va faire une première évaluation du modèle sur cette cohorte on reverra ensuite comment on va appeler cette cohorte notre **cohorte d'entraînement.**



Ensuite on va extraire une deuxième partie de la population un deuxième échantillon qu'on va utiliser pour tester uniquement notre modèle pour l'évaluer on appellera cette cohorte notre **cohorte de test.**

Et c'est très important que la cohorte d'entraînement comme la cohorte de test soit bien représentative de notre population générale.
En effet si on prend une cohorte d'entraînement qui est non représentative par exemple si on entraîne notre modèle sur un sous-échantillon de notre population ici les patients bleus alors on va créer un modèle qui pourra être applicable à ce sous-échantillon au bleu mais qui ne sera pas forcément applicable pour l'ensemble de la population.

C'est aussi un peu ce qui se passe quand on est confronté à un problème d'overfitting dans ce cas-là même si on est sur un échantillon qui est représentatif de la population générale pour nous le modèle est tellement complexe qu'il prend en compte des variables qui sont en réalité très spécifiques à nos données d'entraînement et dont la distribution ne sera pas forcément la même sur le reste de la population.



De même notre population test sur laquelle on va évaluer notre modèle elle doit être représentative de notre population si on prend un échantillon de patients qui est extrait d'une autre population peut-être que notre modèle serait efficace sur notre population initiale sauf que sur cette nouvelle population il ne va pas forcément être applicable.
C'est aussi pour ça que si on entraîne un modèle sur des patients venant d'un pays occidental on ne va pas pouvoir forcément appliquer ce modèle sur des patients d'un pays du tiers monde.

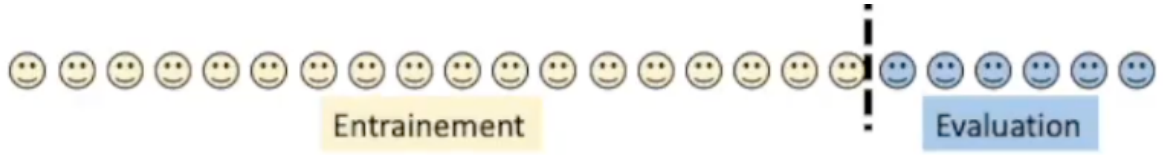
➔ ➔➔Donc il faudra faire bien attention que nos cohortes d'entraînement et de test soient toutes les deux représentatives de la population générale sur laquelle on veut travailler.

Donc comme on l'a vu on va avoir des données d'entraînement celle-ci vont nous permettre d'entraîner le modèle et on va faire une première évaluation du modèle sur ces données pour vérifier qu'il est suffisamment performant et ensuite pour vérifier que ces performances sont reproductibles et applicables à l'ensemble de notre population on va réévaluer notre modèle sur une population test

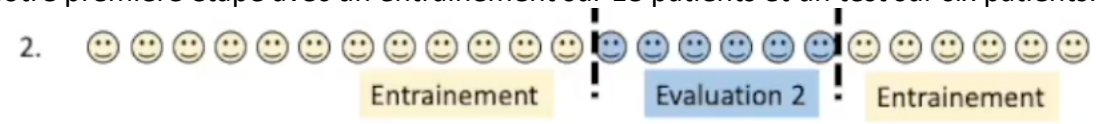
IX. Crossvalidation

= méthode qui est fréquemment utilisée pour l'évaluation des modèles si on constitue une corde d'entraînement avec 24 patients .

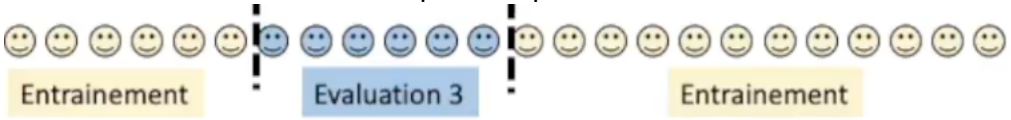
Sur ces patients on va vouloir donc entraîner le modèle et on va aussi vouloir évaluer ce modèle pour se faire ce qu'on pourrait faire c'est utiliser l'ensemble pour faire l'entraînement et ensuite appliquer le modèle pour l'évaluation mais si on fait ça comme c'est cette population qui a servi à l'entraînement on risque de surestimer les performances de modèle .

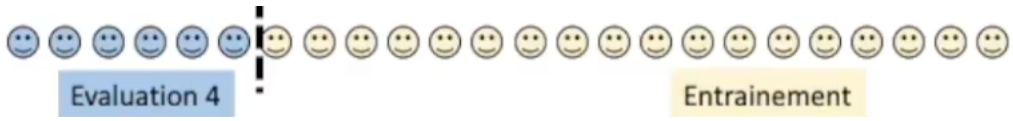


Donc ce qu'on va faire c'est qu'on va séparer notre population en mettant de côté des patients qu'on ne va utiliser que pour l'évaluation du modèle donc on va avoir la plus grosse partie de la population donc ici les trois quarts qui vont servir à l'entraînement du modèle et on va garder un quart de côté pour faire simplement l'évaluation et on n'aura pas utilisé ces patients là pour entraîner le modèle.
Maintenant ce qu'il faut savoir c'est que pour être robuste un modèle il doit avoir été entraîné sur un nombre suffisamment important de patients et surtout il faut le tester sur un nombre suffisant de patients.
Ici notre évaluation de la performance basée sur six patients ce n'est pas très robuste.
On va utiliser une astuce avec une méthode qui s'appelle la cross-validation donc on va faire notre première étape avec un entraînement sur 18 patients et un test sur six patients.



Ensuite on va recommencer mais cette fois-ci en prenant sur nos 24 patients 18 nouveaux patients pour l'entraînement et six nouveaux patients pour le test les six patients qui avaient été utilisés pour l'évaluation précédemment sont utilisés pour l'entraînement et on prend dans les 18 autres six nouveaux patients pour l'entraînement.





On va recommencer une troisième fois puis une quatrième fois et donc au final à la place d'évaluer notre modèle seulement sur un quart de notre population on a fini par évaluer notre modèle sur 4 quarts de la population donc l'ensemble de la population et ça ça va nous donner une première information sur la qualité de notre modèle sur cette population d'entraînement.

Dans ce cas précis comme on a coupé notre population en quatre on dit qu'on a fait la cross-validation à quatre **fold** mais on aurait aussi pu le faire en six en trois peu importe on peut faire cette cross-validation en utilisant un nombre de fold plus ou moins important.
Ce qui aura une petite influence sur le résultat final et dans tous les cas on aura évalué notre modèle sur l'ensemble des données de notre échantillon.

Ensuite il faudra tout de même évaluer ce modèle sur une deuxième population : une population test celle-ci étant indépendante de la population d'entraînement pour affirmer que notre modèle est bien reproductible.

- Tableau de contingence :

		Vérité	
		Y vrai = négatif	Y vrai = positif
Prédiction	Y prédit = négatif	Vrais Négatifs (VN)	Faux Négatifs (FN)
	Y prédit = positif	Faux Positifs (FP)	Vrais Positifs (VP)

Sensibilité (Se) = $\frac{VP}{VP+FN}$

Spécificité (Sp) = $\frac{VN}{VN+FP}$

Valeur Prédictive Positive (VPP) = $\frac{VP}{VP+FP}$

Valeur Prédictive Négative (VPN) = $\frac{VN}{VN+FN}$

Accuracy (Précision) Acc = $\frac{VP+VN}{VP+FP+VN+FN}$

Maintenant on va voir comment on fait pour cette évaluation.
Tout d'abord on va parler du cas de la classification dans ce cas-là l'évaluation va être proche des tests qu'on utilise pour évaluer la performance d'un examen diagnostique.
On va utiliser un tableau de contingence quand on fait cette évaluation dans les cohortes qu'on utilise on connaît le label que le modèle devrait prédire ici c'est ce qu'on va noter le Y vrai et donc on va faire tourner le modèle ce qui va nous donner des Y prédit par le modèle.

Ensuite on va comparer le Y prédit au Y attendu ou au Y vrai pour évaluer si le modèle fonctionne correctement donc si on travaille avec deux classes on va avoir un tableau de contingence comme celui au-dessus avec à gauche les Y prédit et en haut les Y vrai.

Quand le Y vrai est négatif et qu'il est correctement prédit en négatif ça va nous donner un vrai négatif de même si le Y vrai est positif et qu'il est correctement prédit en positif ça va nous donner un vrai positif.

Par contre si le Y vrai était positif mais qu'il a été prédit en négatif on va avoir un faux négatif le résultat du modèle est négatif alors qu'il aurait dû être positif.

Enfin si le modèle donne un résultat positif alors que ça aurait dû être négatif c'est un faux positif.

À partir de là on va pouvoir mesurer différentes métriques qui vont nous donner une information sur la qualité du modèle.
Tout d'abord on aura :

- la sensibilité = la capacité du modèle à prédire un résultat positif quand il doit prédire ce résultat positif
- la spécificité = la capacité du modèle à prédire un résultat négatif quand il doit être négatif

Ensuite on va pouvoir mesurer des valeurs prédictives avec la **valeur prédictive positive** qui correspondra à la probabilité qu'un résultat soit effectivement positif quand le modèle la prédit en tant que tel.

Puis la **valeur prédictive négative** qui correspondra à la probabilité qu'un résultat soit effectivement négatif quand le modèle la prédit en tant que tel.

Enfin en machine learning on va utiliser une autre métrique qui s'appelle
l'accuracy = pourcentage de fois où le modèle a correctement fait la prédiction et donc au nombre de patients qui ont été correctement prédits sur l'ensemble de la cohorte évaluée .

Cette métrique l'accuracy est assez pratique car elle peut aussi assez facilement être utilisée quand on utilise plus de deux classes par exemple on peut avoir une table de contingence comme ici ← avec trois classes dans ce cas-là si on veut parler de sensibilité ou de spécificité il faudra expliquer pour quelle classe on parle.

	Y vrai = 1	Y vrai = 2	Y vrai = 3
Y predit = 1	50	5	5
Y predit = 2	10	52	3
Y predit = 3	5	0	60

On pourra par exemple parler de la sensibilité pour la classe 1 qui est égale donc au nombre de patients effectivement de classe 1 qui sont prédits comme étant de classe 1 divisé par la totalité des patients qui sont effectivement de classe 1 ce qui va donc nous donner une information uniquement sur la qualité du modèle pour cette classe 1 alors que l'accuracy à ce moment-là correspondra au nombre de patients dont la classe prédite est correcte divisé par la totalité des patients.
Ce qui est donc une métrique qui représente l'ensemble du modèle.

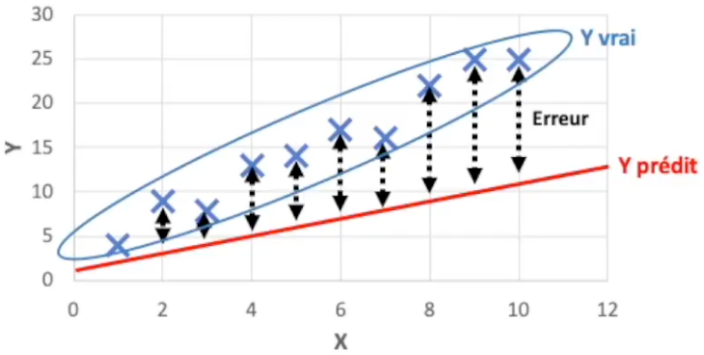
	Y vrai = 1	Y vrai = 2	Y vrai = 3
Y predit = 1	200	20	20
Y predit = 2	0	0	0
Y predit = 3	0	0	0

Accuracy = $\frac{bonnes\ prédictions\ (200)}{total\ (240)} = 83{,}3\%$

Cependant avec cette métrique il faudra faire attention entre la répartition des classes dans nos populations n'est pas équilibrée.
En effet si on prend une population avec 200 patients dans la classe 20 patients dans la classe 2 et 20 patients dans la classe 3 même si le modèle met tout le monde dans la classe 1 et est donc inutile.

L'accuracy sera de 200 sur 240 donc de 83% ce qui pourra donner l'impression que le modèle fonctionne correctement alors qu'il fait quelque chose qui est complètement inutile à savoir mettre tout le monde dans le même groupe.

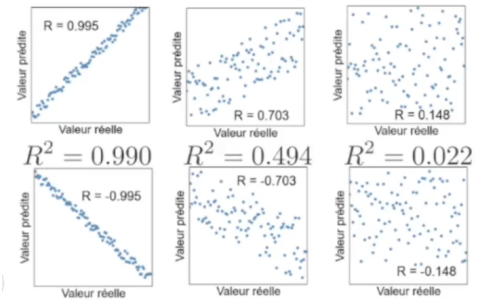
- Erreur : Distance entre la valeur prédite et la valeur réelle



Pour ce qui est de l'évaluation d'un modèle de régression comme notre label Y peut prendre tout un tas de valeurs on ne va pas utiliser une table de contingence on va prendre notre cohorte de test et on va y appliquer notre modèle.
Ainsi encore une fois pour tous les individus de notre échantillon on va avoir un Y prédit et un Y vrai et on va comparer ces deux Y pour obtenir l'erreur réalisé par le modèle qui va correspondre à la distance entre le Y prédit et le Y vrai.

Il existera différentes mesures de l'erreur on peut utiliser la somme des différences portée au carré ou la racine carrée de cette somme de différence portée au carré quoi qu'il en soit il se sera toujours une mesure de l'erreur.

- Coefficient de détermination



On pourra également utiliser une métrique qui s'appelle le R carré ou R2 ou coefficient de détermination ce coefficient de détermination correspond au carré du coefficient de corrélation entre les Y prédits et les Y vrais.
Ainsi il faut faire attention car si les Y prédits et les Y vrais sont inversement corrélés le R carré pourra tout de même être très enlevé et on pourra donc avoir faussement l'impression que notre modèle fonctionne très bien alors qu'il renvoie une valeur inverse à celle qui est attendue.
Mais on ne rentrera pas dans les détails concernant le coefficient de détermination et il ne fera pas l'objet d'une question pour l'examen

Résumé :
Pour l'évaluation des modèles en apprentissage supervisé on va faire une première évaluation par cross validation sur notre cohorte d'entraînement ce qui va nous donner une information sur le fit du modèle si les performances sont trop basses on pourra dire que le modèle est **underfit** et si les performances sont très élevées avec une **accuracy** proche de **100%** cela peut vouloir dire que le modèle est très bon mais cela peut aussi être un signe **d'overfitting**.

Ensuite faire une deuxième évaluation sur notre cohorte de test en appliquant directement notre modèle ce qui va nous permettre d'évaluer la reproductibilité de ce modèle pour le reste de notre population.
Si les performances sont équivalentes à celle mesurée sur la cohorte d'entraînement alors que cela veut dire que nos résultats sont **reproductibles**.
Cependant, si les performances sont bien moins bonnes que celle mesurée sur la cohorte d'entraînement alors cela peut vouloir dire que notre modèle est overfitté ou sinon il y a un problème d'échantillonnage et que notre cohorte de test est trop différente de notre cohorte d'entraînement.

Donc au final pour appliquer nos méthodes de machine learning que ce soit en médecine en biologie ou dans d'autres domaines il faudra tout d'abord bien définir notre problème.

1. Bien définir son problème :
   - Apprentissage Supervisé / Non supervisé.
   - Régression / Classification

2. Bien choisir ses données :
   - Bon échantillonage
   - Bien définir le label Y
   - Bien définir les variables explicatives x

On va définir la question à laquelle on va répondre et à partir de ça on saura si on a besoin d'un apprentissage supervisé et non-supervisé et s'il faudrait utiliser une méthode de classification de régression.
En parallèle la définition de notre problème on va nous permettre de choisir les données qui permettront d'y répondre pour se faire il faudra faire un bon échantillonnage dans notre population cible , bien définir notre label après dire y dont la méthode de mesure devra être fiable et de même il faudra bien définir nos variables explicatives X qu'on pourra sélectionner manuellement ou à l'aide d'un algorithme dédié ou un peu des deux et pour lesquels les méthodes de mesure devront être également le plus fiable possible.

3. Adapter son algorithme d'apprentissage
   - Comprendre le principe d'un algorithme comme la descente de gradient
   - Savoir si un scaling est nécessaire
   - Savoir adapter le taux d'apprentissage

4. Evaluer son modèle
   - Comprendre l'underfitting et l'overfitting
   - Comprendre la cross validation
   - Comprendre les mesures d'évaluation

On utilisera ensuite un algorithme d'apprentissage par exemple la descente de gradient même s'il en existe d'autres et pour ce faire il faudra comprendre le principe de fonctionnement cet algorithme pour pouvoir ajuster les paramètres.
Comme par exemple pour la descente de gradient savoir ajuster le taux d'apprentissage.
De même au moment où on aura analysé les données d'entrée il faudra savoir si un scaling est nécessaire.
L'application de notre méthode d'apprentissage va nous donner un modèle et il va falloir évaluer des performances de ce modèle. Pour ce faire il faudra comprendre et connaître ce qu'est l'underfitting et l'overfitting il faudra comprendre ce qui est la cross validation et il faudra comprendre quels sont les mesures qui permettent de faire cette évaluation.
En particulier pour la classification supervisée avec la sensibilité la spécificité l'accuracy …etc

Donc voilà ce sont je pense les notions à avoir pour pouvoir appréhender une méthode de machine learning le but du cours n'étant pas que vous sachiez au bout d'une heure appliquer directement un algorithme de machine learning mais que si un jour vous êtes amené à travailler avec des statisticiens sur ce type d'approche que vous puissiez avoir des bases pour comprendre comment ça marche et à quoi il faut faire attention

Les dédis :
Dédis au tutorat et à la merveilleuse expérience que ça a été et aux rencontres que j'ai pu y faire
Dédi à moi parce je suis la femme que je pense être = > Étudiante en Médecine /Tutrice et Pâtissière qui peut me test
Dédi à toi et à la force que t'a tu peux être fier/fière de toi pour tout ce que t'as accompli ➔ rappelle-toi la personne que t'était en aout et la personne que t'es maintenant et le beau chemin que t'as parcouru
C'est pas le moment de baisser les bras continue on est sur la dernière ligne droite et surtout oublie pas de me défoncer ces qru

RENDEZ MOI FIÈRE JE LE SUIS DEJA ET JE CROIS EN VOUS Force et honneur spartiate 👊👊💪💪💪💪