Pour utiliser un algorithme d'IA il est nécessaire de disposer de ……..… en général beaucoup de données. …….

Pour pouvoir traiter ces données, il peut être nécessaire qu'elles soient organisées sous une certaine forme. On peut par exemple identifier :

• les données structurées, dans une base de données par exemple;

• les données semi-structurées, dans un fichier csv, par exemple;

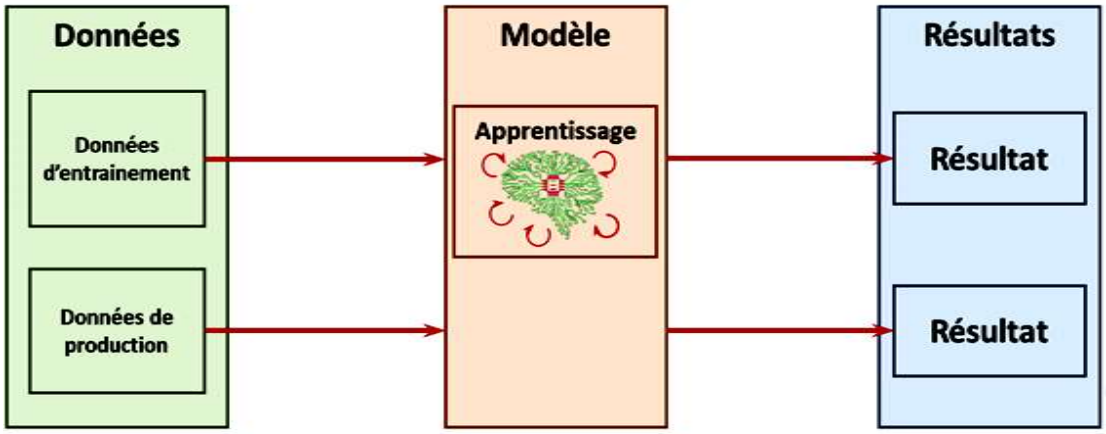
• les données non structurées comme une image, du texte, ou une vidéo.

L'apprentissage automatique est un champ de l'intelligence artificielle dont l'objectif est d'analyser un grand volume de données afin de déterminer des motifs et de réaliser un modèle prédictif.

L'apprentissage comprend deux phases :

• ……...……... est une phase d'estimation du modèle à partir de données d'observations;

• ……...……...….... est une phase pendant laquelle de nouvelles données sont traitées dans le but d'obtenir le résultat souhaité. L'entraînement peut être poursuivi même en phase de production.

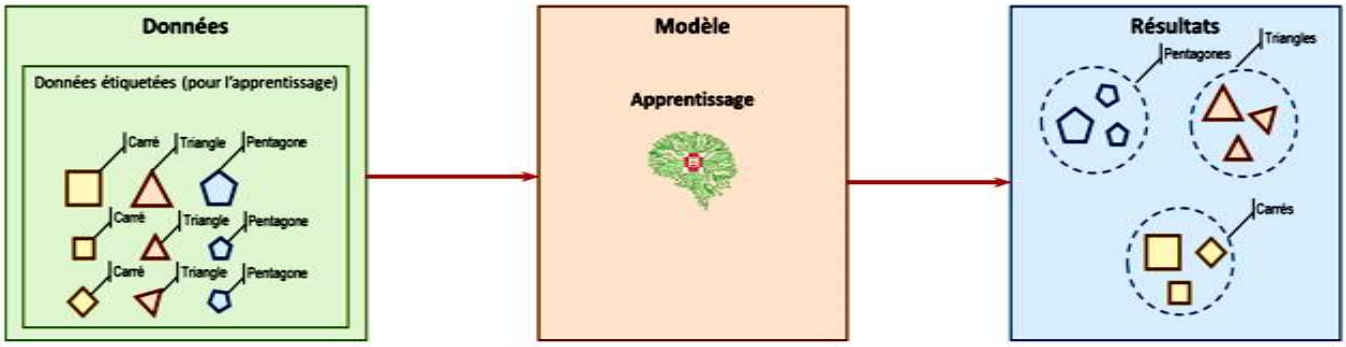


Il existe plusieurs types d'apprentissage automatique dont :

• L'**apprentissage supervisé** : on dispose d'un ……...……. à partir desquelles onva mettre en évidence une **tendance** qui nous servira à faire des **predictions** sur de nouvelles données ;

• L'**apprentissage non supervisé ou clustering :** Tâche d'apprentissage au cours duquel l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de ……...…… déterminer un lien entre les données (et les regrouper).

Dans ce cours **on se limite à l'apprentissage supervisé**. Ce type d'apprentissage peut permettre de résoudre aussi bien des problèmes de **r** ……... que des problèmes de **c** ……...……



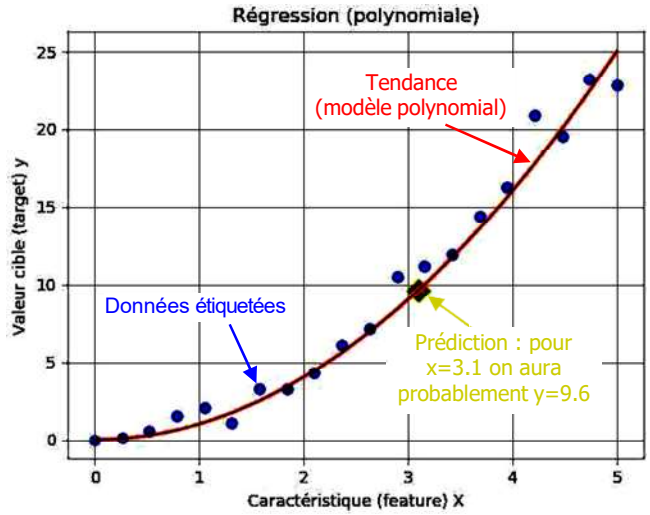
Pendant la ……...…… l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de données étiquetées, déterminer un lien entre un ensemble les données et les étiquettes. Dans l'exemple ci-dessus des formes sont étiquetées avec leur nom, le modèle va regrouper les formes par nom.

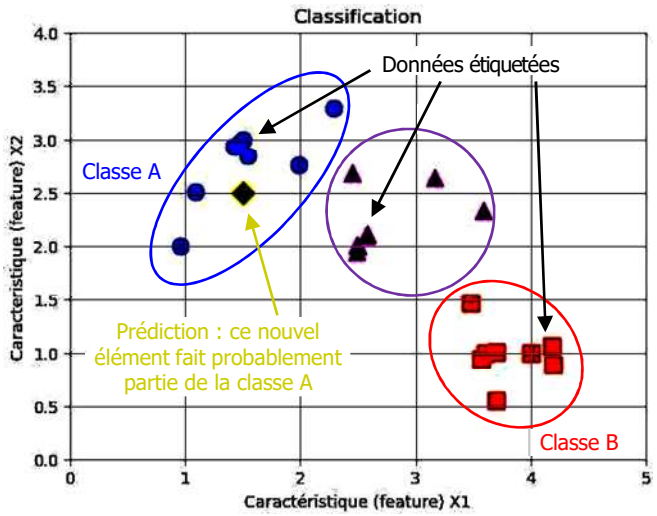
Puis pendant la ……...……l…. l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de données non étiquetées, prédire l'étiquette. Dans l'exemple ci-dessus on soumet au modèle une forme et le modèle indique son nom.

Régression et classification

- La **régression** est utilisée lorsque la sortie à prédire peut prendre des valeurs continues, il s'agit d'une variable réelle. Exemple : un algorithme prédisant la consommation électrique d'une installation ou un algorithme prédisant le cours d'actions en bourse.

- La **classification** est une tâche consistant à choisir une classe (valeur) parmi toutes celles possibles Exemple : Un algorithme prédisant le chiffre manuscrit sur l'image d'entrée ou un algorithme classifiant une tumeur comme « bénigne » ou « maligne ».



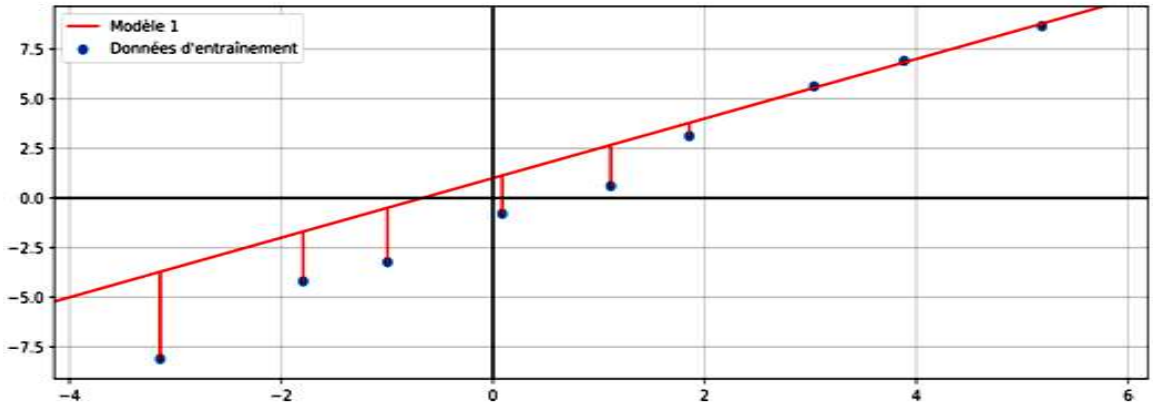


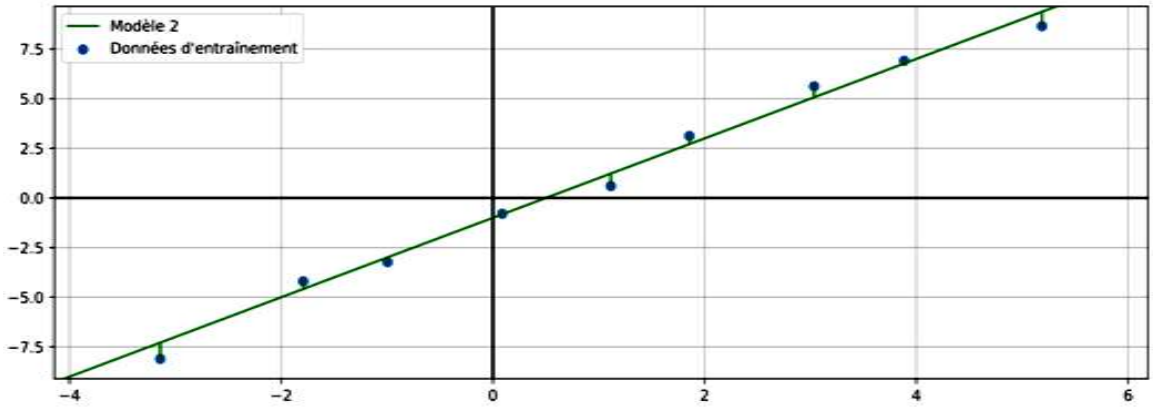
LA REGRESSION LINEAIRE

L'objectif de la régression linéaire est d'exprimer un……...……...…...
……...……...……...
Ce modèle a donc deux paramètres A et B, dont il faut trouver les valeurs optimales durant la phase d'apprentissage. Plusieurs techniques existent pour estimer ces paramètres, la plus répandue étant la méthode des moindres carrés.

La régression linéaire est un algorithme d'apprentissage supervisé, on dispose alors de N couples entrée-sortie constituant l'ensemble de données $\mathbf{D} = \{\mathbf{x_i}\}\ \mathbf{i} \in [1,\mathbf{N}]$. Ce sont ces données connues qui vont permettre d'estimer les paramètres du modèle.

Nous allons prendre ici l'exemple de la méthode des moindres carrés. Dans le cas de cette méthode des moindres carrés, on cherche les paramètres A et B permettant de minimiser une fonction coût . Cette fonction correspond à la somme des écarts au carré entre les prédictions et les valeurs attendues. Ces écarts à minimiser sont appelés résidus.

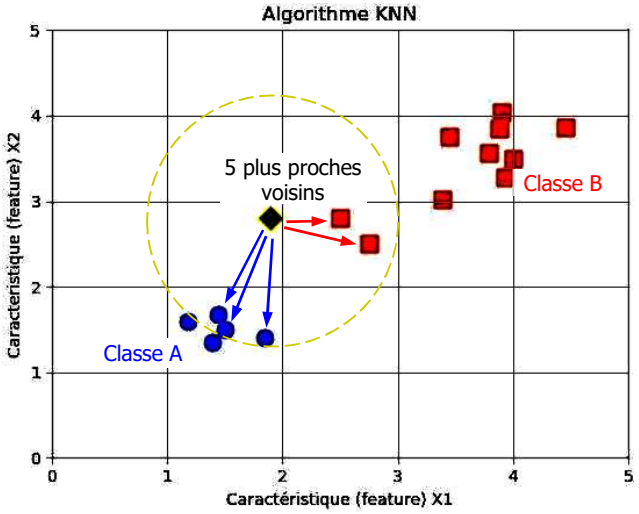




Ci-dessus on a deux modèles de régression linéaire Le premier modèle présente des écarts importants entre les valeurs prédites et attendues tandis que le second minimise les carrés de ces écarts

K PLUS PROCHES VOISINS ALGORITHME KNN (K NEAREST NEIGHBORS)

L'algorithme des K plus proches voisins est un algorithme d'apprentissage supervisé que l'on peut utiliser pour la classification de données discrètes. Le principe est extrêmement simple : si les K plus proches voisins d'un élément sont majoritairement d'une certaine classe, alors cet élément sera associé à cette classe.

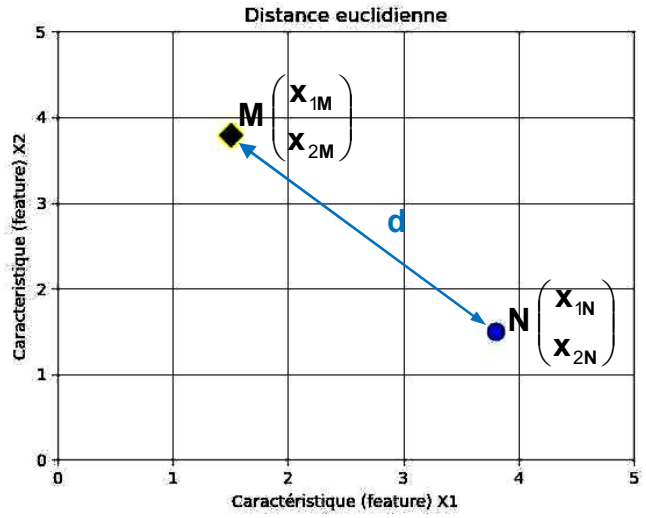


Par exemple, sur la figure ci-contre, les 5 plus proches voisins de l'élément que l'on cherche à classer sont majoritairement de classe A. On peut en déduire que cet élément sera probablement lui aussi de classe A.

K est appelé **hyper-paramètre**. Son choix conditionne l'efficacité de l'algorithme.

On peut remarquer que si on avait choisi K=3 sur l'exemple précédent, la conclusion aurait été différente.

La proximité avec des voisins est évaluée de grâce à la **distance euclidienne**.



En deux dimensions, cela donne :

$$\mathbf{d(M,N)} =$$

Plus généralement, dans un espace de dimension n :

$$\mathbf{d(M,N)} =$$

Mise en place de l'algorithme KNN

Conventions d'écriture :

- La machine reçoit des données dont les caractéristiques (features) sont regroupées dans le vecteur $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$;
- Ces données sont annotées par une étiquette (label/target) y.

Début Algorithme

Données en entrée :

- Un ensemble de données
- Un nombre entier K

Pour une nouvelle observation dont on veut prédire l'étiquette de sortie, Faire :

1- Calculer toutes les distances de cette observation avec les caractéristiques de l'ensemble des données.

2- Retenir les K étiquettes du jeu de données les plus proches de cette observation.

3- Choisir la valeur d'étiquette dominante parmi les K valeurs précédente.

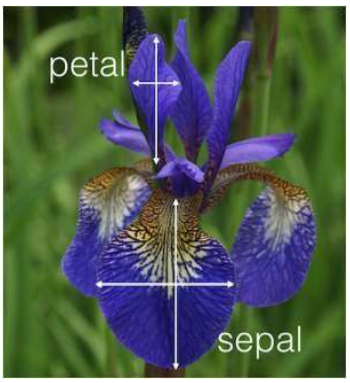
4- Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par KNN pour l'observation.

5- Fin Algorithme

Nous allons travailler sur une base de données établie par un botaniste pour distinguer trois type d'Iris différentes. Cette base de données contient 150 mesures différentes des caractéristiques de la fleur et les classe en trois variétés : Setosa, Versicolor et Virginia:

Chaque mesure contient 4 caractéristiques (features) et est étiquetée par une étiquette (label/target).

Caractéristiques				**Etiquette**
Longueur sépale	Largeur sépale	Longueur pétale	Largeur pétale	Variété
5.1	3.5	1.4	0.2	0 (Setosa)
4.9	3.0	1.4	2.0	0 (Setosa)
7.0	3.2	4.7	1.4	1 (Versicolor)
6.4	3.2	4.5	1.5	1 (Versicolor)
5.8	2.7	5.1	1.9	2 (Virginica)
7.1	3.0	5.9	2.1	2 (Virginica)

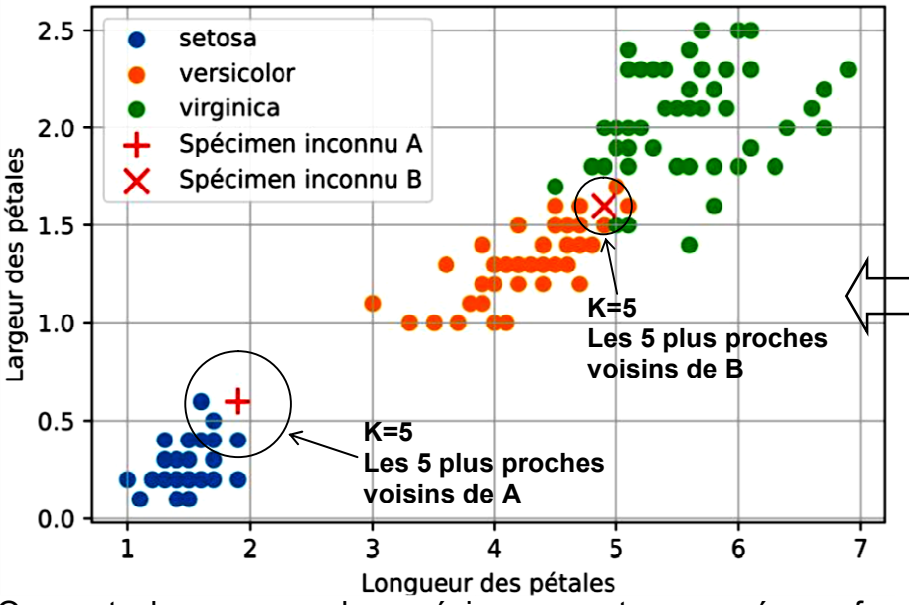


Les dimensions sont données en cm.

Les variétés Setosa, Versicolor, et Virginica sont respectivement étiquetée 0, 1, et 2.

Pour chaque spécimen, il a été mesuré la longueur et la largeur des pétales et des sépales.

Dans le cadre de cet exemple, nous allons nous intéresser uniquement à la longueur et largeur des pétales. Les entrées x_i sont donc les vecteurs contenant la longueur des pétales et la largeur des pétales. Les sorties y_i sont un entier 0, 1 ou 2, correspondant à la classe du spécimen (setosa, versicolor ou virginica).



Bases de données Iris, les 150 spécimens sont répartis en 3 variétés et positionnés en fonction de la longueur et largeur des pétales

On peut observer que les spécimens sont regroupés en fonction de leur variété. En effet, des spécimens d'une même variété ont des caractéristiques similaires. Il semble alors cohérent de prédire l'appartenance d'un spécimen à une variété selon les autres spécimens proches (au sens de la distance euclidienne).
Considérant pour cet exemple deux spécimens inconnus A et B et cherchons à prédire la variété à laquelle ils appartiennent.

• Le premier spécimen, appelé **A** est représenté par un '+' sur le graphique. Il s'agit d'une iris avec des pétales de 1,9 cm de longueur et 0,6 cm de largeur. Cette iris a des caractéristiques plus proches de celle des spécimens d'Iris Setosa que de celle des autres spécimens. On peut alors classer cette iris dans la variété Setosa (ses 5 plus proches voisins sont des Setosa).

• Le second spécimen, appelé **B**, a des pétales de longueur 4,9 cm et de largeur 1,6 cm et est représenté par un 'x' sur le graphique. On peut alors regarder les 5 spécimens d'iris connus les plus proches du spécimen B. On peut constater que parmi ceux-là il y a 1 Iris Virginica et 4 Iris Versicolor. On classe alors le spécimen B dans la variété Versicolor.

Applications et limites

L'algorithme des K plus proches voisins est fréquemment utilisé en classification comme en régression. On peut alors le retrouver dans le cadre de la reconnaissance de forme avec des entrées contenant la circonférence, l'aire ou encore la signature du contour de la forme et des sorties correspondant aux différentes formes possibles. Cet algorithme a l'avantage d'être relativement robuste si suffisamment d'exemple d'entraînement sont fournis.

Le choix de la valeur de K dans l'algorithme des K plus proches voisins a une influence forte sur la prédiction faite par l'algorithme :

- **K trop petit** : des éléments sortant de l'ordinaire influeront plus facilement la prédiction. La généralisation du modèle pour de nouveaux éléments sera donc moins bonne, il s'agit du problème de **sur-apprentissage** (overfitting). Par exemple, si l'on choisit K=1, un Iris Versicolor ayant des pétales anormalement petits sera pris en compte par le modèle qui risque de mal classifier une iris aux dimensions trop proche de cette valeur aberrante.

- **K trop grand** : Le modèle prendra en compte des données trop éloignées et la classe majoritaire sera prédite trop souvent. Il s'agit cette fois du problème de **sous apprentissage**, le modèle n'utilise que trop peu les données d'entraînement.

Ainsi, le choix de la valeur de K n'est pas aisé, il ne suffit pas de la prendre grande ou de la prendre petite.

Différentes méthodes existent pour avoir une idée de la valeur de K possible.

Il est également important de tester l'algorithme sur un ensemble de données connues afin de vérifier le fonctionnement de ce dernier.
On peut alors séparer l'ensemble de données connues D en deux ensembles :

- L'ensemble d'entraînement D_e
- L'ensemble de test D_t.

Le premier permettra de réaliser la prédiction et le second permettra de vérifier les performances de l'algorithme et d'éviter les phénomènes de sur-apprentissage et de sous-apprentissage. Un des principaux problèmes rencontrés lors de l'implémentation de l'algorithme des K plus proches voisins est le fléau de la dimension. En effet, pour que l'algorithme fonctionne de manière optimale, il faut un nombre suffisant de données d'entraînement afin que les points étudiés soient toujours proches d'exemples connus. Lors de l'étude de problèmes de plus grande dimension, il est donc primordial d'avoir un grand nombre de données d'entraînement. Pour cette raison, l'algorithme des K plus proches voisins devient rapidement inutilisable (au-delà de 4 ou 5 dimensions, le nombre de données nécessaires devient trop important).

COMMENT VALIDER LE MODELE ?

En d'autres termes comment savoir si le modèle est suffisamment précis

Une fois un algorithme d'apprentissage choisi, on se pose la question de la validation du modèle. Quels critères et outils vont nous permettre de considérer que notre apprentissage est « bon » ? Lors d'un problème de classification, il est par exemple possible de déterminer les écarts entre les valeurs prédites par l'algorithme et les valeurs cibles.

Critères de validation des problèmes de classification

- $\text{Justesse} = \dfrac{\text{Nombre de prédictions vraies}}{\text{Nombre de prédictions totales}}$

- La matrice de confusion est une métrique permettant de déterminer la qualité d'une classification. En abscisses sont indiquées les valeurs prédites, et en ordonnée les valeurs réelles.

Dans l'exemple ci-dessous, nous cherchons à classifier des iris, grâce à un algorithme, selon 3 familles : les Setosa, les Versicolor et les Virginica. Sur la diagonale, sont retrouvées les iris dont l'espèce a été correctement prédite. En revanche, 3 Versicolor ont été classées parmi les Virginica et 2 Virginica ont été classifiées dans les Versicolor

La matrice de confusion présentée est dite non-normalisée. Si la répartition des étiquettes n'est pas uniforme (en fonction des classes), il est probable qu'il y ait davantage d'erreurs pour les classes ayant beaucoup d'étiquettes. Pour palier ce problème on peut utiliser des matrices de confusions normalisées (on divise alors chaque terme de la matrice par la somme des éléments de la même ligne).

Dans l'exemple ci-dessous on a réellement 50 données par famille d'Iris

Valeurs réelles	Setosa	50	0	0
	Versicolor	0	47	3
	Virginica	0	2	48
		Setosa	Versicolor	Virginica
		Valeurs prédites		

On a Setosa vrais positifs
On a Versicolor vrais positifs
On a Virginica vrais positifs
On a Virginica faux positifs (ce sont des Versicolor)
On a Versicolor faux positifs (ce sont des Virginica)

Soit une population de 6000 individus séparés en trois classes A, B et C de 2000 individus chacune

		Valeurs prédites			
		A	B	C	Total
Valeurs réelles	A	1000	600	400	2000
	B	400	1200	400	2000
	C	200	200	1600	2000
	Total	1600	2000	2400	

Soit le repérage suivant dans la matrice $\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$

a) Qu'est ce que signifient les nombres 11, 22 et 33 ?

b) Qu'est ce signifie le nombre 23 ?

c) Calculer la justesse des prédictions.

On définit la précision et le rappel de la façon suivante :

$$\textbf{Précision}_i = \frac{\text{Nombre de prédictions vraies pour la classe i}}{\text{Nombre de prédictions totales attribuées à la classe i}}$$

$$\textbf{Rappel}_i \textbf{ ou sensibilité}_i = \frac{\text{Nombre de prédictions vraies pour la classe i}}{\text{Nombre réel d'individus appartenant à la classe i}}$$

d) Calculer la précision de chacune des classes.

	Classe A	Classe B	Classe C
Précision			

e) Calculer la sensibilité des prédictions.

	Classe A	Classe B	Classe C
Sensibilité			

LES RESEAUX DE NEURONES

Bref historique
Dans les années 1940, les chercheurs tentent de fabriquer une machine capable d'apprendre à partir de données fournies, de mémoriser des informations et de traiter des informations incomplètes. Pour cela, ils essayent de réaliser des modèles mathématiques de neurones biologiques. S'en suit alors la naissance des premiers modèles de perceptrons.

L'apprentissage automatisé va alors connaître des hauts et des bas, au gré des avancées scientifiques et technologiques. Dans les années 60 , un des premiers coups d'arêt fût provoqué par la non-capacité des réseaux de neurones à traiter des problèmes non linéaires. Dans les années 80 , la rétroproagation du gradient fut proposée. Mais devant le manque de capacité des ordinateurs, la recherche marquât un second coup d'arrêt. Dans les années $90/2000$, l'apparition des réseaux convolutifs et leur capacité à analyser les données des images relançât alors les recherches dans ce domaine.

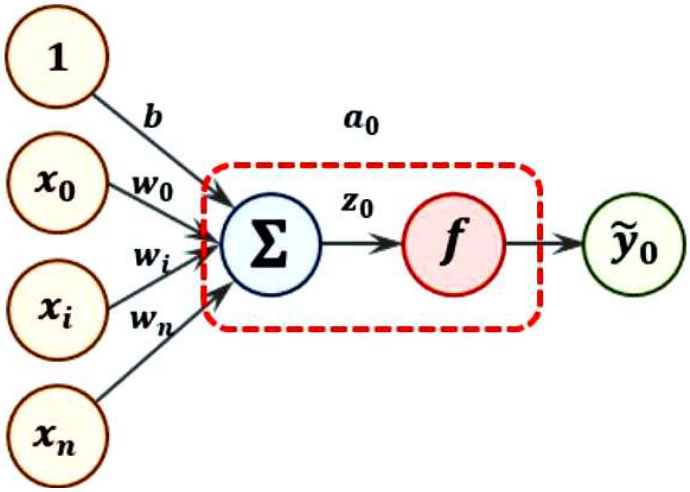
1.2 Exemples d'applications des réseaux de neurones
https://fr.wikiversity.org/

- Banque : prêts et scoring.
- Cartes de crédit : détection des fraudes.
- Finance : analyse d'investissements et de fluctuations des taux de change.
- Assurance : couverture assurantielle et estimation des réserves.
- Marketing : ciblage des prospections, mesures et comparaisons des campagnes et des méthodes.
- Archéologie : identification et datation de fossiles et d'ossements.
- Défense : identification de cibles.
- Environnement : prévisions de la qualité de l'air et de l'eau.
- Production : contrôles qualité.
- Médecine : diagnostics médicaux.
- Energies : stimations des réserves, prévisions de prix.
- Pharmacie : efficacité de nouveaux médicaments.
- Psychologie : prévisions comportementales.
- Immobilier : études de marchés.
- Recherche scientifique : identification de spécimens, séquençages de protéines.
- Télécommunication: détection des pannes de réseaux.
- Transport : maintenance des voies.
- Le neurone, les réseaux de neurones

Modèle de neurone
Définition - Neurone (ou perceptron).

Prenons la représentation suivante pour un neurone. On note:

- X le vecteur d'entrée et x_i les données de la couche d'entrée;
- w_i les poids (poids synaptiques);
- b le biais ou offset;
- z_0 la somme pondérée des entrées;
- f une fonction d'activation;
- $\tilde{y_0}$: la valeur de sortie du neurone.



On a donc, dans un premier temps :

$$z_0 = b + \sum_{i=0}^{n} w_i x_i$$

Après la fonction d'activation, on a donc en sortie du neurone :

$$\tilde{y}_0 = f(z_0) = f\left(b + \sum_{i=0}^{n} w_i x_i\right).$$

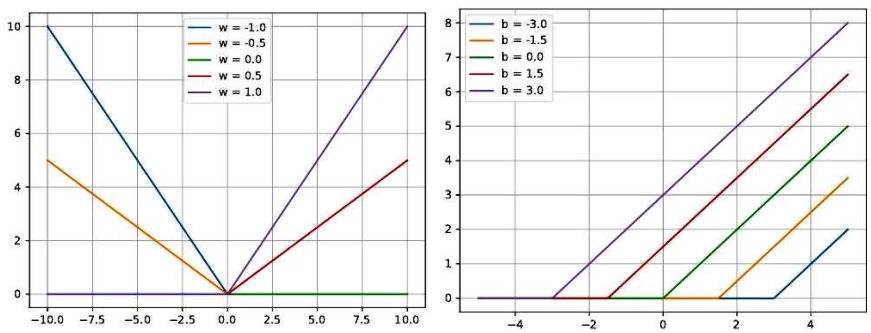
1. La notation tilde ($\tilde{y}_0$) vient du fait que la valeur de sortie d'un neurone est une valeur estimée qu'il faudra comparer à y_0 valeur de l'étiquette utilisée pour l'apprentissage supervisé.
2. Par la suite, dans la représentation graphique on ne fera pas apparaître la somme pondérée et la fonction d'activation, mais seulement la valeur de sortie du neurone (notée par exemple a_0).

Définition

- Fonction d'activation. Les fonctions d'activation sont des fonctions mathématiques appliquées au signal de sortie (z). Il est alors possible d'ajouter des non linéarités à la somme pondérée. On donne ci-dessous quelques fonctions usuelles:

Identité	Heaviside	Logistique (sigmoïde)	Unité de rectification linéaire (ReLU)
$f(x) = x$	$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$	$f(x) = \dfrac{1}{1 + \mathrm{e}^{-x}}$	$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$

Influence des poids et des biais sur la sortie du perceptron en utilisant une fonction d'activation ReLU.

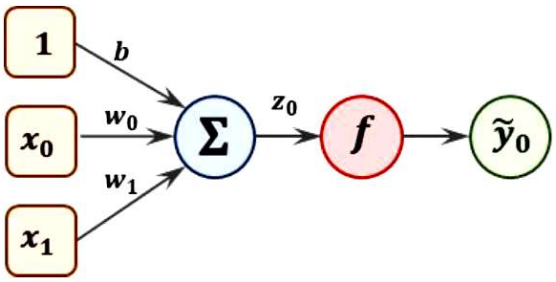


On peut ainsi voir qu'avec la fonction d'activation ReLU, plus le poids sera grand en valeur absolu, plus le neurone amplifiera le signal d'entrée. Le biais permettra de prendre en compte le «niveau » du signal d'entrée à partir duquel, le signal doit être amplifié, ou non.

- Exemple

Prenons un neurone à deux entrées binaires.

Initialisation les poids et le biais avec des valeurs aléatoires : $w_0 = -0{,}3, w_1 = 0{,}8$ et $b = 0{,}2$.



On peut donc évaluer l'ensemble des sorties calculable par le neurone.

x_0	x_1	z	Id.	H.	Sig.	ReLu
0	0	0,2	0,2	1	0.549	0,2
0	1	1	1	1	0.731	1
1	0	-0.1	-0.1	0	0.475	0
1	1	0.7	0.7	1	0.668	0.7

Réseaux de neurones

https://playground.tensorflow.org/

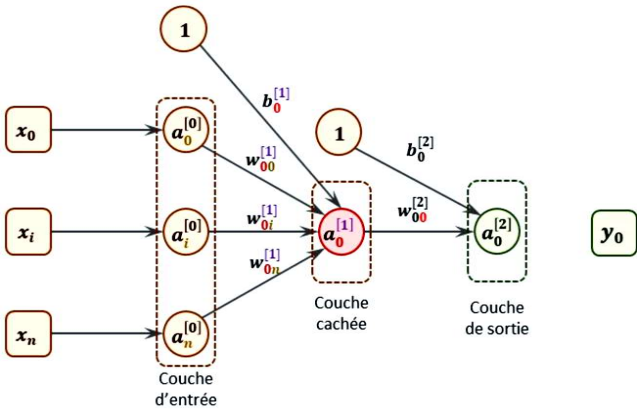
Modélisation d'un réseau de neurones

Définition

- Couches.

Un réseau de neurones est un ensemble de neurones reliés, par couches, entre eux. Dans un réseau de neurones dense tous les neurones de la couche i seront reliés à tous les neurones de la couche $i + 1$.

- Couche d'entrée : cette couche est une copie de l'ensemble des données d'entrées. Le nombre de neurones de cette couche correspond donc aux nombre de données d'entrées. On note $\mathrm{X} = (x_0, \dots, x_n)$ le vecteur d'entrées.
- Couche cachée (ou couche intermédiaire) : il s'agit d'une couche qui a une utilité intrinsèque au réseau de neurones. Ajouter des neurones dans cette couche (ou ces couches) permet donc d'ajouter de nouveaux paramètres. Pour une couche, la même fonction d'activation est utilisée pour tous les neurones. En revanche la fonction d'activation utilisée peut être différente pour deux couches différentes. Les fonctions d'activations des couches intermédiaires sont souvent non linéaires.
- Couche de sortie : le nombre de neurones de cette couche correspond au nombre de sorties attendues. La fonction d'activation de la couche de sortie est souvent linéaire. On note $\mathrm{Y} = (y_0, \dots, y_y)$ le vecteur des sorties.



En utilisant la loi de comportement du modèle de perceptron, on peut donc exprimer $\mathrm{Y} = \mathcal{F}(\mathrm{X})$ où $\mathcal{F}$ est une fonction dépendant des entrées, des poids et des biais.

Notations:

- on note $w_{jk}^{[\ell]}$ les poids permettant d'aller vers la couche ℓ depuis le neurone k vers le neurone j;
- $b_j^{[\ell]}$ le biais permettant d'aller sur le neurone j de la couche ℓ;
- $f^{[\ell]}$ la fonction d'activation de la couche ℓ;
- $n^{[\ell]}$ le nombre de neurones de la couche ℓ.

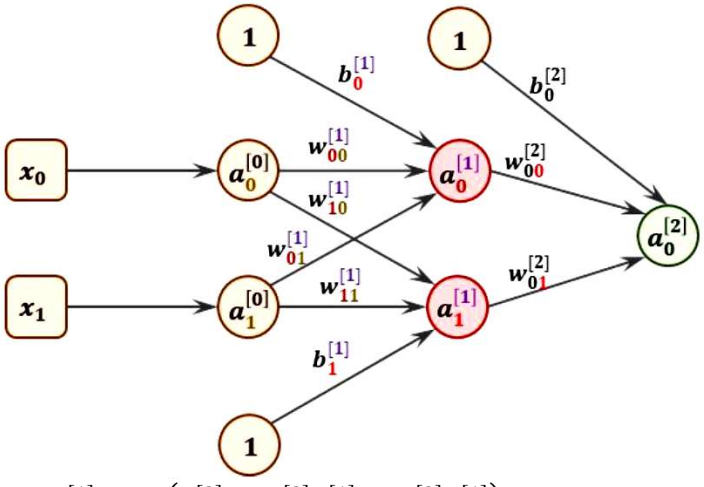
Définition - Équation de propagation. Pour chacun des neurones $a_j^{[\ell]}$ on peut donc écrire l'équation de propagation qui lui est associé :

$$a_j^{[\ell]} = f^{[\ell]}\left(\sum_{k=0}^{n^{[\ell-1]}} \left(w_{jk}^{[\ell]} a_k^{[\ell-1]}\right) + b_j^{[\ell]}\right) = f^{[\ell]}\left(z_j^{[\ell]}\right).$$

- Exemple

Prenons un réseau de neurones à 3 couches :

- 1 couche d'entrée à 2 neurones;
- 1 couche cachée à 2 neurones, de fonction d'activation f_1;
- 1 couche de sortie à 1 neurone, de fonction d'activation f_2;
- Initialisation les poids et le biais avec des valeurs aléatoires : $w_0 = -0{,}3, w_1 = 0{,}8$ et $b = 0{,}2$.



Il est possible d'écrire que $y_0 = a_0^{[1]} = f_2\left(b_0^{[2]} + w_{00}^{[2]} a_0^{[1]} + w_{01}^{[2]} a_1^{[1]}\right)$.

Par ailleurs : $a_0^{[1]} = f_1\left(b_0^{[1]} + w_{00}^{[1]} a_0^{[0]} + w_{01}^{[1]} a_1^{[0]}\right)$ et $a_1^{[1]} = f_1\left(b_1^{[1]} + w_{10}^{[1]} a_0^{[0]} + w_{11}^{[1]} a_1^{[0]}\right)$.

Au final, on a donc

$$y_0 = a_0^{[1]} = f_2\left(b_0^{[2]} + w_{00}^{[2]}\left(f_1\left(b_0^{[1]} + w_{00}^{[1]} a_0^{[0]} + w_{01}^{[1]} a_1^{[0]}\right)\right) + w_{01}^{[2]}\left(f_1\left(b_1^{[1]} + w_{10}^{[1]} a_0^{[0]} + w_{11}^{[1]} a_1^{[0]}\right)\right)\right)$$

Définition - Paramètres. Les paramètres du réseau de neurones sont les poids et les biais, autant de valeurs que l'entraînement devra déterminer.

Méthode - Calcul du nombre de paramètres - à vérifier.
Soit un jeu de données étiquetées avec n entrées et p sorties.

On construit un réseau possédant ℓ couches et a_ℓ le nombre de neurones de la couche ℓ. Dans ce cas, la première couche est la couche d'entrée $(a_1 = n)$ et la dernière couche et la couche de sortie ($a_\ell = p$).

Nombre de poids : $n_w = \sum_{i=1}^{\ell-1}(a_i \times a_{i+1})$.

Nombre de bais : $n_b = \sum_{i=2}^{\ell}(a_i)$.

Au final, le nombre total de paramètre à calculer est donné par $N = n_w + n_b$.

Objectif Soit un jeu de données étiquetées. On note X le vecteur des données d'entrées. On note Y le vecteur des données de sorties. On note $\tilde{\mathrm{Y}}$ le vecteur de sortie calculé par le réseau de neurones.

L'objectif de la phase d'apprentissage du réseau de neurones est de déterminer les valeurs de l'ensemble des poids et des biais de telle sorte que l'écart entre Y et $\tilde{\mathrm{Y}}$ soit minimale.

3.2 Fonction de coût
Dans le but de minimiser l'écart entre la sortie du réseau de neurones et la valeur réelle de la sortie, on utilise une fonction coût (ou fonction de perte). Il est possible de définir plusieurs types de fonctions, notamment en fonction du type de problème à traiter (classification ou régression par exemple).

Définition - Fonction coût régression. Notons nb le nombre de données dans la base d'entraînement. Dans le cadre d'un problème de régression, on peut définir la fonction coût comme la moyenne des erreurs quadratique entre la valeur donnée par l'équation de propagation et la valeur de l'étiquette :

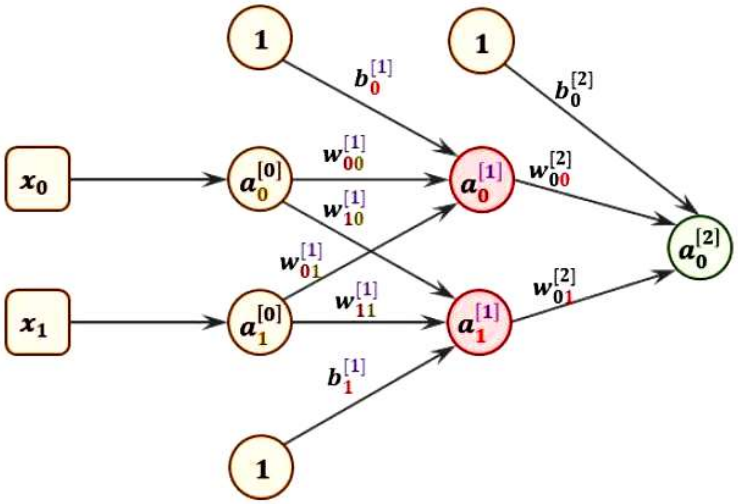
$$C = \frac{1}{nb}\sum_{i=1}^{nb}\left(\tilde{\mathrm{Y}}_\mathrm{i} - \mathrm{Y}_\mathrm{i}\right)^2$$

Objectif L'objectif est dès lors de déterminer les poids et les biais qui minimisent la fonction coût.

Notion de rétropropagation - Descente de gradient
En réutilisant l'exemple ci-contre, nous allons présenter succinctement comment est minimisée la fonction coût. Pour cela, il va falloir dériver la fonction coût par rapport à chacune des variables.

Cherchons uniquement à déterminer le coût que par rapport à un seul vecteur d'entrée du jeu d'entraînement. On a alors :

- $C = \left(\tilde{\mathrm{Y}}_\mathrm{i} - \mathrm{Y}_\mathrm{i}\right)^2 = \left(a_0^{[2]} - y\right)^2$;
- $a_0^{[2]} = f^{[2]}\left(z_0^{[2]}\right)$;
- $z_0^{[2]} = \sum_{k=0}^{1}\left(w_{0k}^{[2]} a_k^{[1]}\right) + b_0^{[2]}$.



Commençons par déterminer la dérivée partielle par rapport à un poids de la couche de sortie :

$\frac{\partial C}{\partial w_{00}^{[2]}} = \frac{\partial C}{\partial a_0^{[2]}} \frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} \frac{\partial z_0^{[2]}}{\partial w_{00}^{[2]}}$.

De même, on peut calculer la dérivée partielle du coût par rapport au biais: $\frac{\partial C}{\partial b_0^{[2]}} = \frac{\partial C}{\partial a_0^{[2]}} \frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} \frac{\partial z_0^{[2]}}{\partial b_0^{[2]}}$.

Calculons les dérivées nécessaires :

- $\frac{\partial C}{\partial a_0^{[2]}} = 2\left(a_0^{[2]} - y\right)$;
- $\frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} = f'^{[2]}\left(z_0^{[2]}\right)$;
- $\frac{\partial z_0^{[2]}}{\partial w_{00}^{[2]}} = a_0^{[1]}$.
- $\frac{\partial z_0^{[2]}}{\partial b_0^{[2]}} = 1$.

On a donc, $\frac{\partial C}{\partial w_{00}^{[2]}} = 2\left(a_0^{[2]} - y\right) f^{[2]}\left(z_0^{[2]}\right) a_0^{[1]}$ et $\frac{\partial C}{\partial b_0^{[2]}} = 2\left(a_0^{[2]} - y\right) f^{[2]}\left(z_0^{[2]}\right)$.

Prenons le cas ou la fonction d'activation est la fonction identité.

On a alors $\frac{\partial C}{\partial w_{00}^{[2]}} = 2\left(a_0^{[2]} - y\right) a_0^{[1]}$ et $\frac{\partial C}{\partial b_0^{[2]}} = 2\left(a_0^{[2]} - y\right)$ ...

On va ainsi pouvoir exprimer $\frac{\partial C}{\partial w_{00}^{[2]}}, \frac{\partial C}{\partial w_{01}^{[2]}}, \frac{\partial C}{\partial b_0^{[2]}}$, ... On pourrait ici écrire 9 équations en fonction des différents poids, des biais et des entrées x_0 et x_1.

À partir de cela, on va modifier les poids comme suit :

- $w_{00,i+1}^{[2]} = w_{00,i}^{[2]} - \eta \frac{\partial C}{\partial w_{00,i}^{[2]}}$;
- $b_{0,i+1}^{[2]} = b_{0,i}^{[2]} - \eta \frac{\partial C}{\partial b_{0,i}^{[2]}}$.

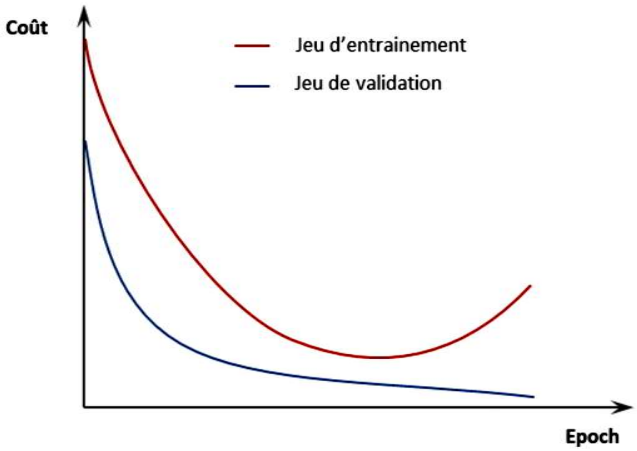
On réitère ensuite les opérations précédentes jusqu'à ce que la fonction coût ait été suffisamment réduite.
Définition - Taux d'apprentissage. On définit l'hyperparamètre $\eta \in [0,1[$ comme étant le taux d'apprentissage. Si ce taux d'apprentissage est très grand, l'algorithme d'apprentissage mettra beaucoup de temps à trouver le minimum. S'il est trop grand, le minimum peut ne jamais être trouvé.

Définition — Gradient.

Dés lors, dans le cas présenté, on peut définir le gradient comme le vecteur $\text{gradC} = \begin{bmatrix} \frac{\partial C}{w^{[1]}} \\ \frac{\partial C}{b^{[1]}} \\ \vdots \\ \frac{\partial C}{w^{[l]}} \\ \frac{\partial C}{b^{[l]}} \end{bmatrix}$.

3.4 Fin d'apprentissage
Définition - Epoch. On appelle epoch un cycle d'apprentissage où tous les poids et tous les biais ont été mis à jour en faisant passer toutes les données du jeu d'entraînement dans les algorithmes de propagation et de rétropropagation.

Les méthodes pour stopper l'apprentissage sont essentiellement empiriques. On pourrait en effet fixer un nombre d'epoch ou une valeur d'erreur admissible et s'arrêter à ce moment là. Dans la pratique, se focaliser sur l'erreur n'est généralement pas satisfaisant. En effet, il y a risque de «suraprentissage».



Dans la figure suivante, on réalise un entrâînement sur le jeu de données (une epoch) à la fin de l'epoch on dispose d'un premier modèle de réseau de neurones. On détermine alors l'erreur commise en utilisant le jeu d'entraînement puis l'erreur commise sur le jeu de validation. On calcule ensuite l'erreur commise par le modèle sur le jeu de validation. (On rappelle que le jeu de validation ne sert pas à modifier les poids et les biais.)

On réalise de même à la fin de l'epoch suivante etc...«Logiquement» l'erreur décroît toujours avec le jeu d'entraînement car la descente du gradient a pour objectif de réduire cette erreur.

Sur le jeu de validation, l'erreur décroît pendant un certain nombre d'epoch puis augmente. Il existe en fait un stade à partir duquel le réseau de neurones se spécialise sur le jeu d'entraînement et devient donc incapable de réaliser des prédictions fiables sur un nouveau je de données. On parle de surentraînement (ou d'overfitting).

- Exemple Wikipedia

La ligne verte représente un modèle sur-appris et la ligne noire représente un modèle régulier. La ligne verte classifie trop parfaitement les données d'entraînement, elle généralise mal et donnera de mauvaises prévisions futures avec de nouvelles données. Le modèle vert est donc au final moins bon que le noir.