

Auteur: Laby Damaro Camara

Email: Idamaro98@gmail.com

Github: <https://github.com/camara94>

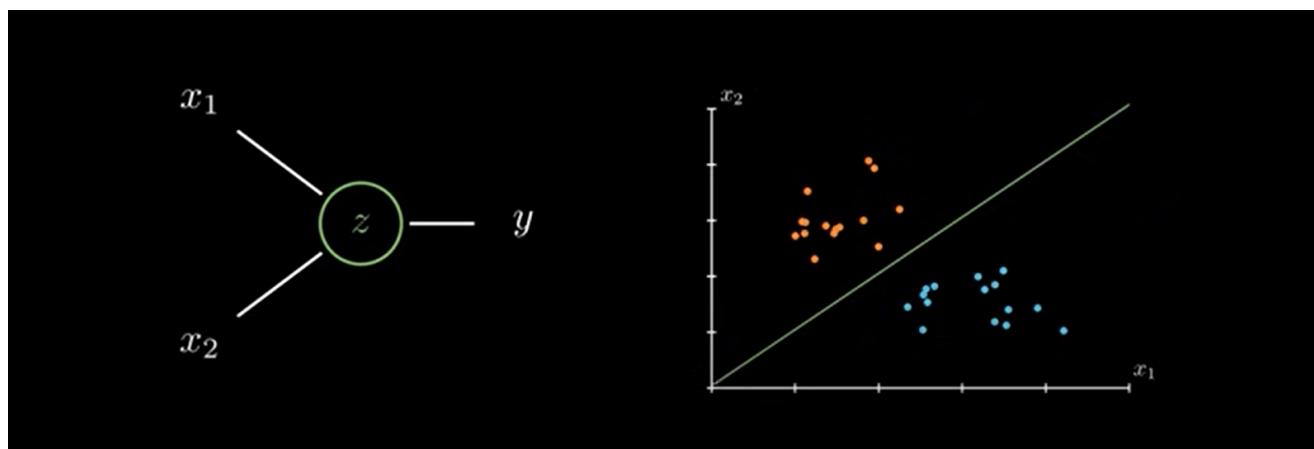
Les Bases de Deep Learning

Dans cette formation, nous allons apprendre les concepts de bases de Deep Learning et les calculs scientifiques

Perceptron

Définition

Le perceptron est l'unité de base des réseaux de neurones. Il s'agit d'un modèle de classification binaire, capable de séparer linéairement 2 classes de points

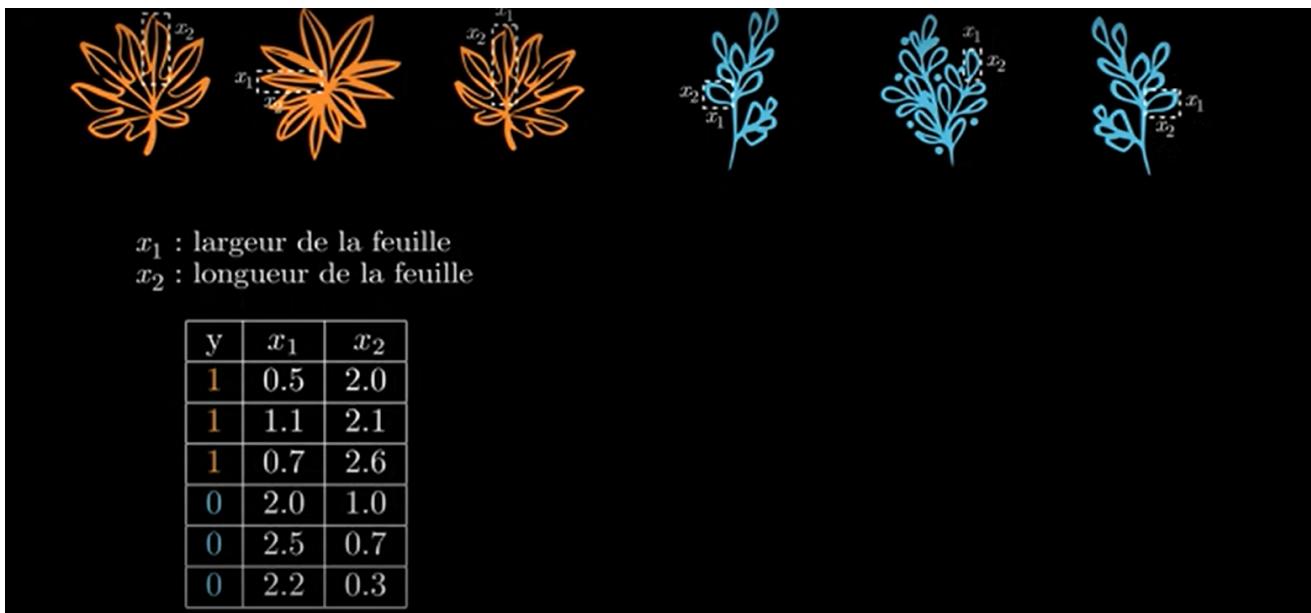


Exemple

Imaginer qu'on ait deux types de plantes(toxique: $y = 1$ et non-toxique $y = 0$).

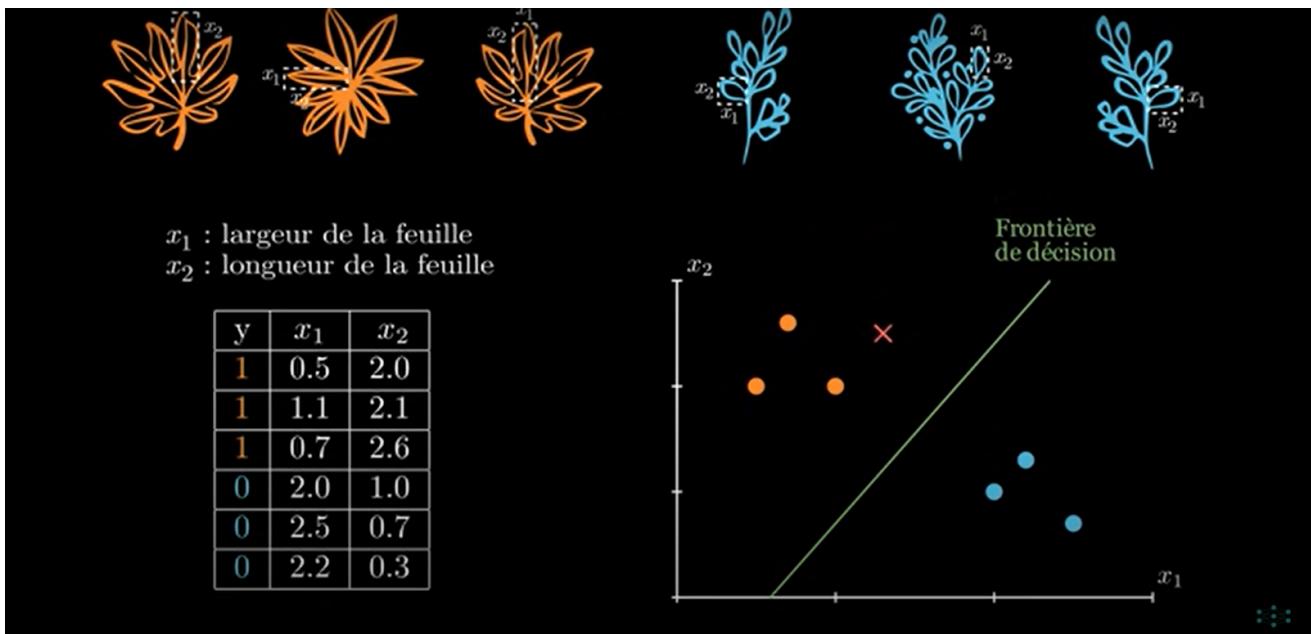


Si un jour, on décide de mesurer les longueurs de ces plantes (x_1 et x_2) .



En représentant les résultats d'un graphique, on constate que ces plantes sont regroupées en deux classes ou groupes.

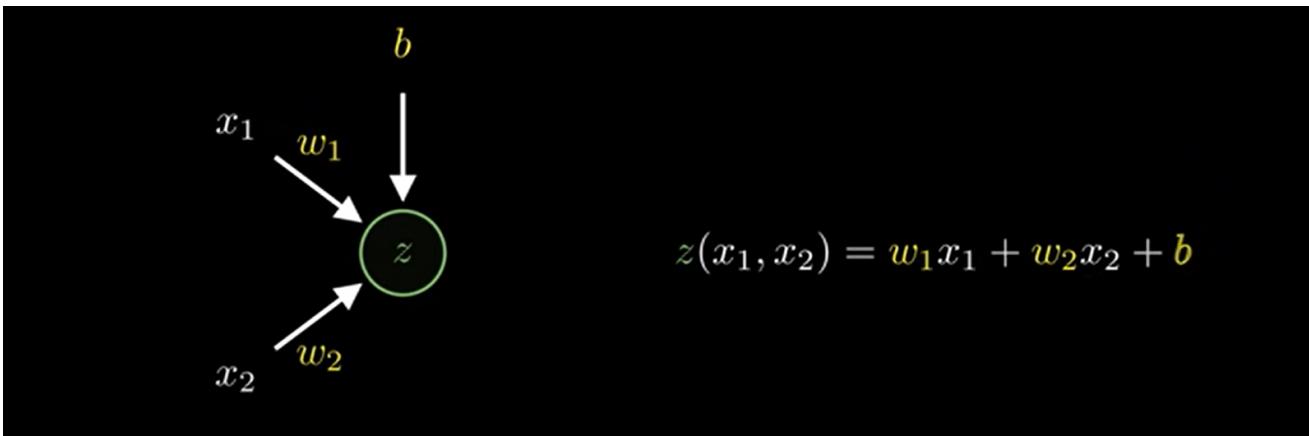
On peut donc développer un modèle capable de prédire à quelle classe appartient une future plante en se basant sur cette droite appelée la frontière de décision.



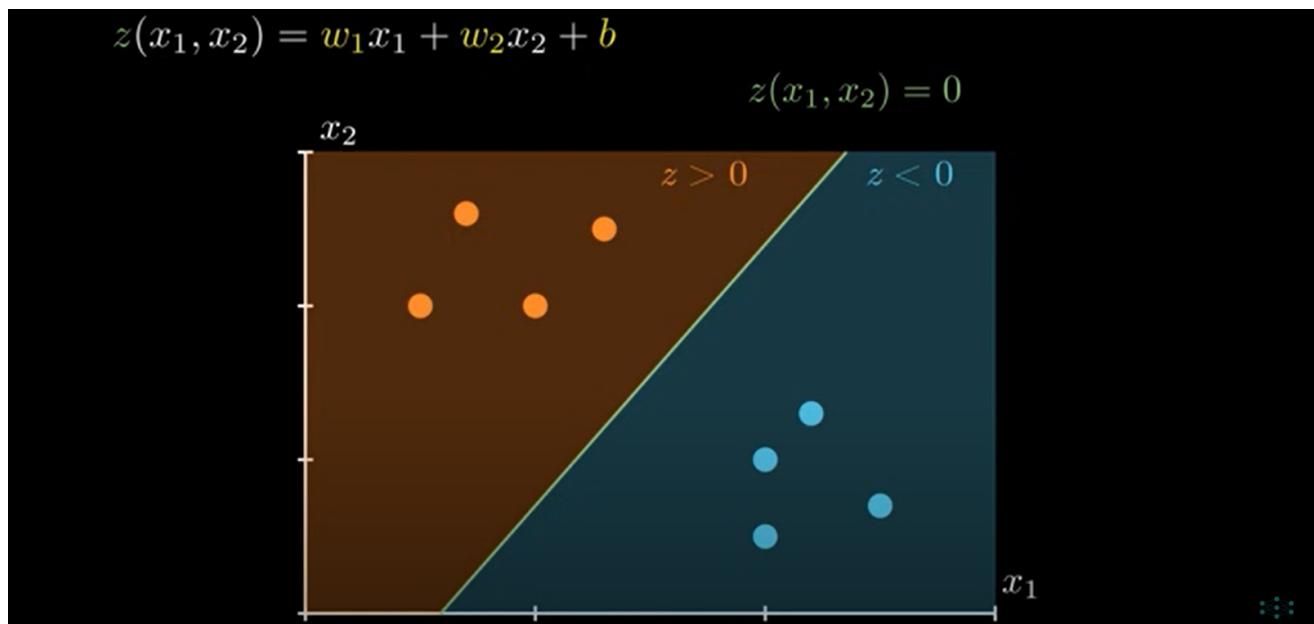
Modèle Linéaire

Pour développer ce modèle, il va falloir l'équation de cette droite.

Pour ça, nous allons développer un modèle linéaire en fournissant nos variables x_1 et x_2 à un neurone en multipliant chaque entrée du neurone par un poids w dans ce neurone, on va également faire passer un coefficient complémentaire b qu'on appelle le biais.

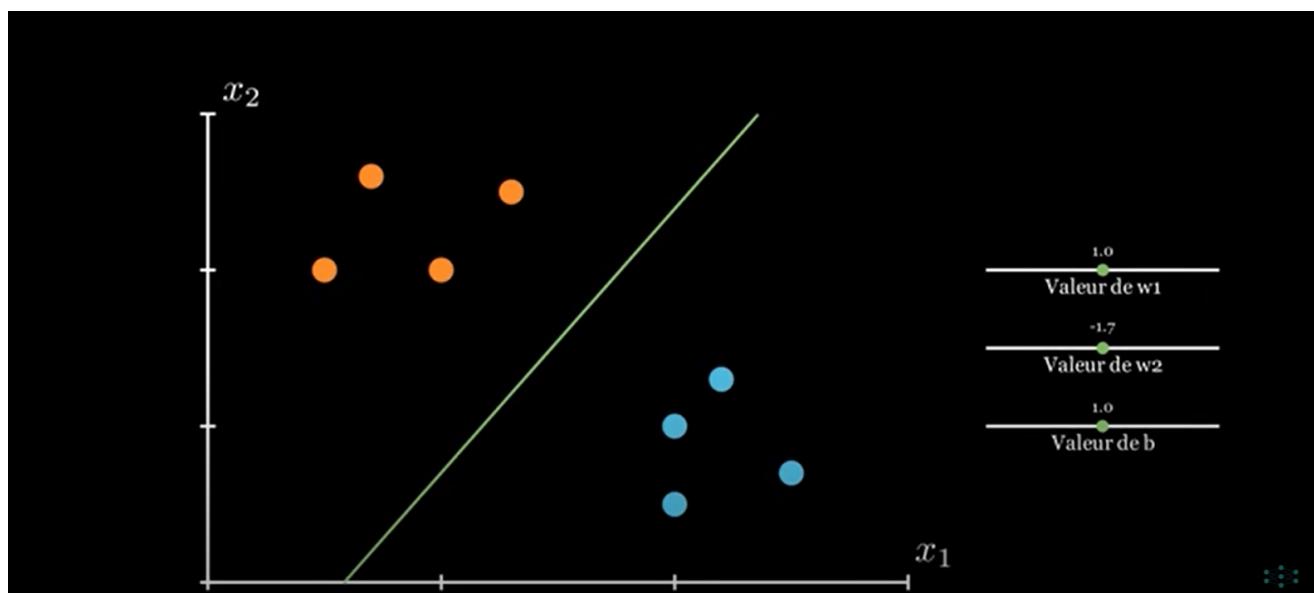


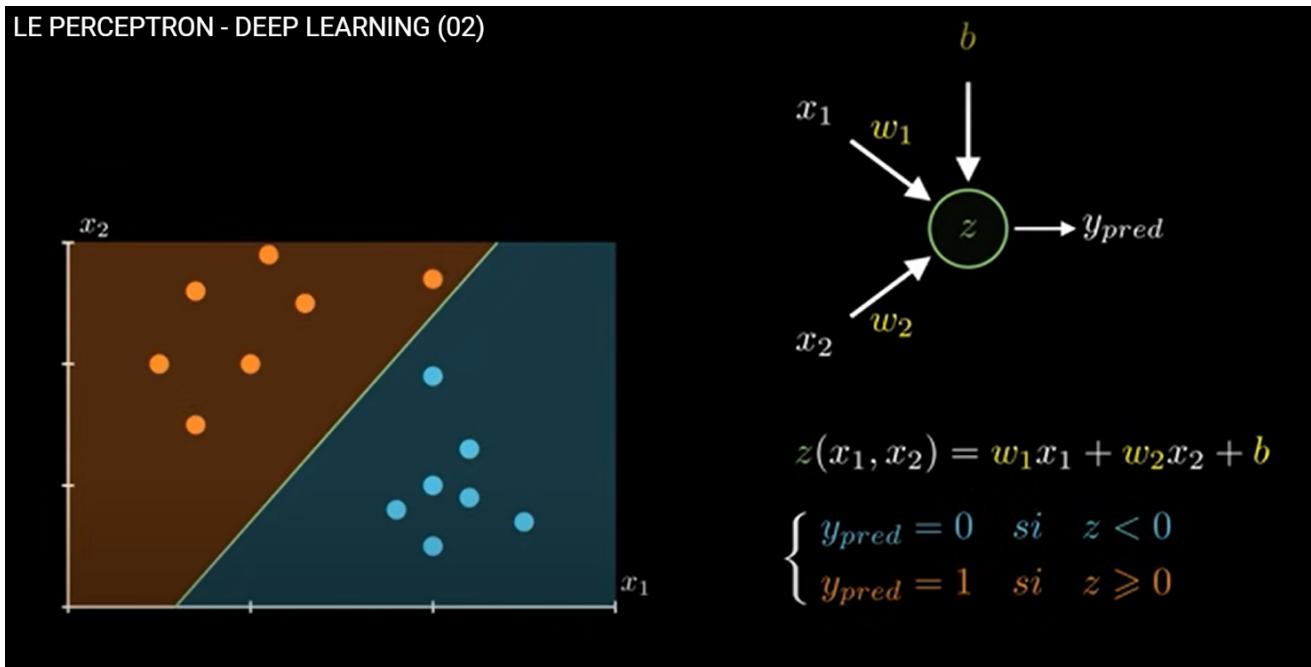
De retour sur notre graphique, on peut colorer les régions où cette fonction nous retourne une valeur positive $z>0$ et les régions où elle nous retourne une valeur négative $z<0$.



Reglage des paramètre

Du coup pour prédire à quelle classe appartient une plante il va falloir régler les paramètre w et b de façon à mieux séparer nos deux classes.

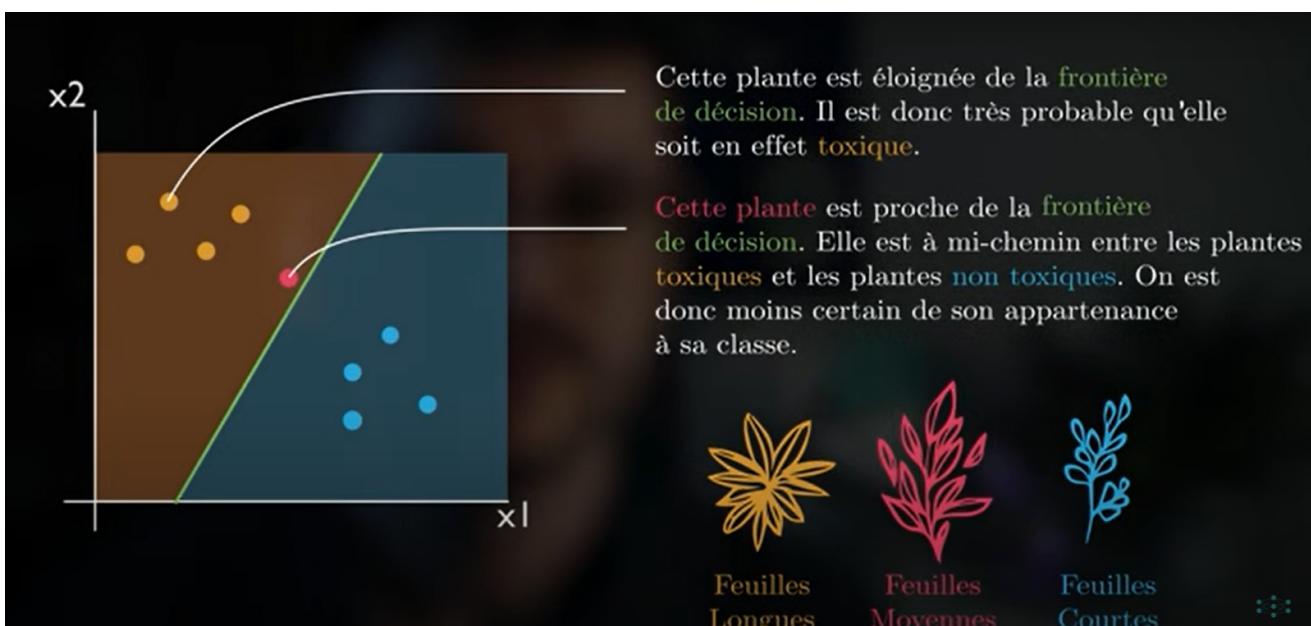




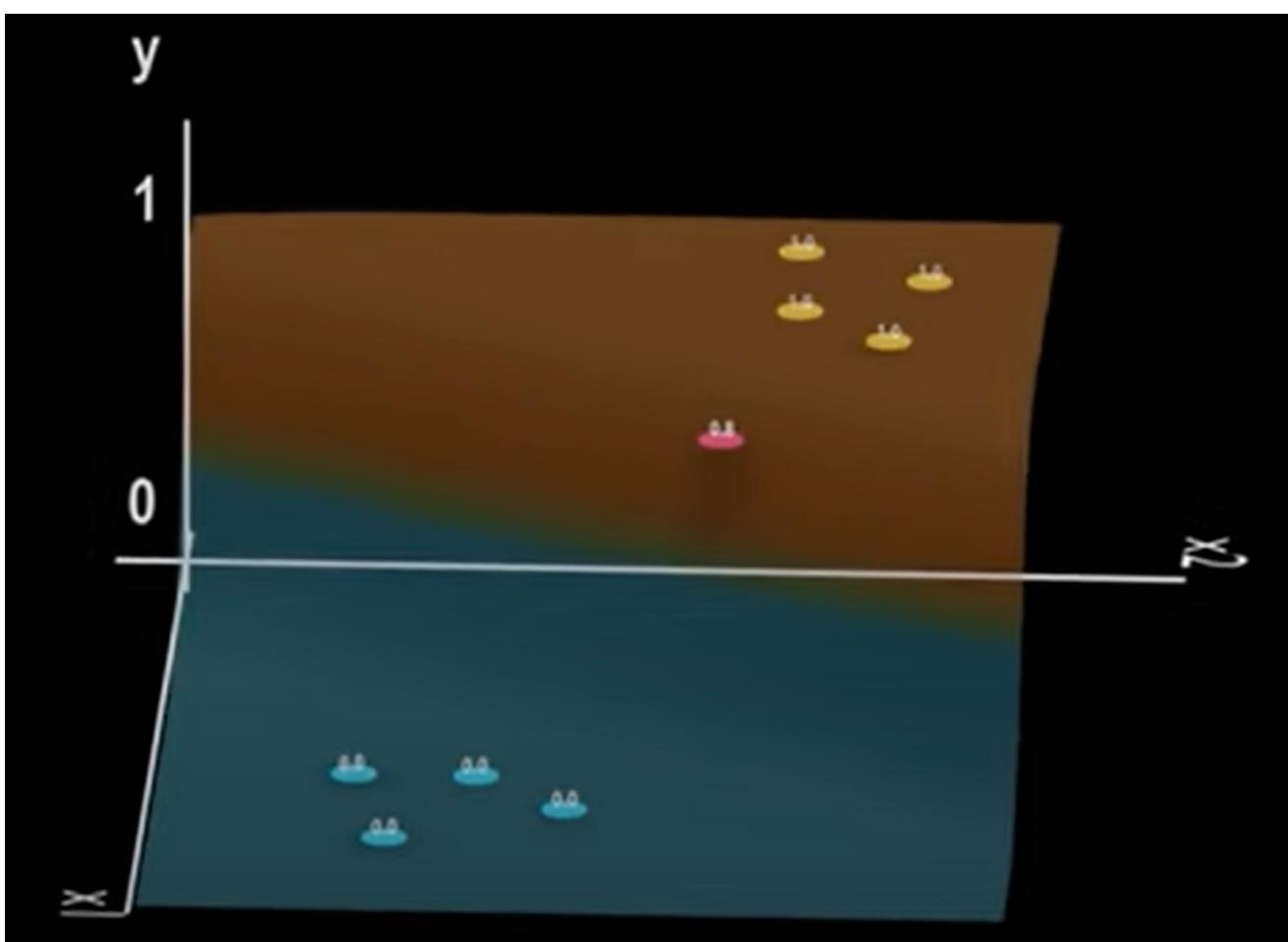
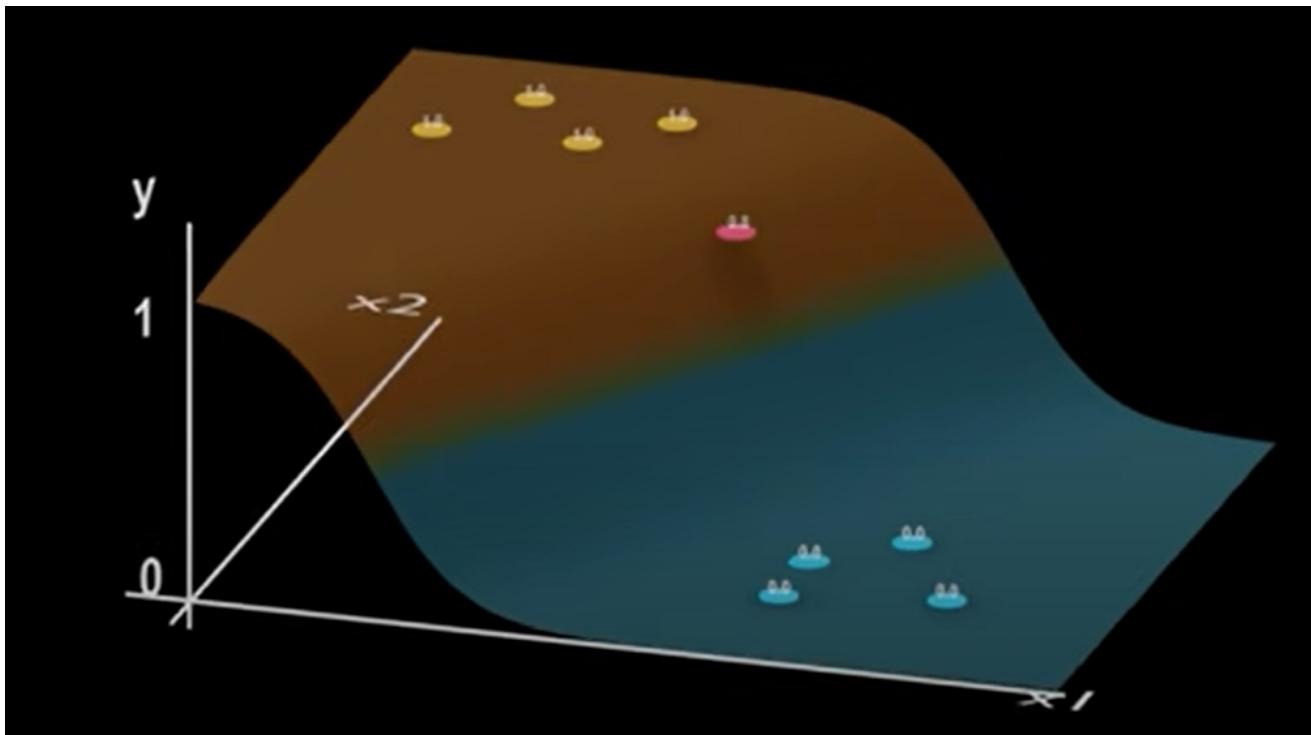
C'est ainsi que fonction un perceptron, le premier neurone de l'histoire de Deep Learning.

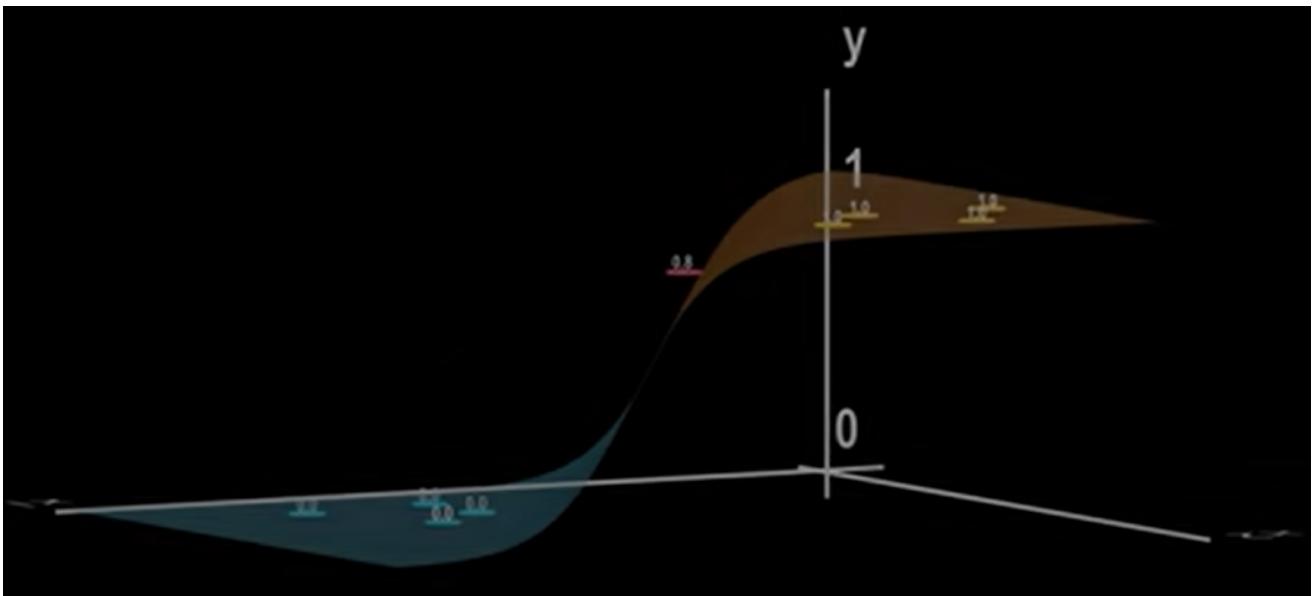
Amélioration

Pour améliorer ce modèle, il sera mieux d'accompagner chaque prédiction d'une probabilité. Plus une plante sera éloignée de la frontière de décision plus il sera évident qu'elle appartienne à sa classe.



Fonction d'activation

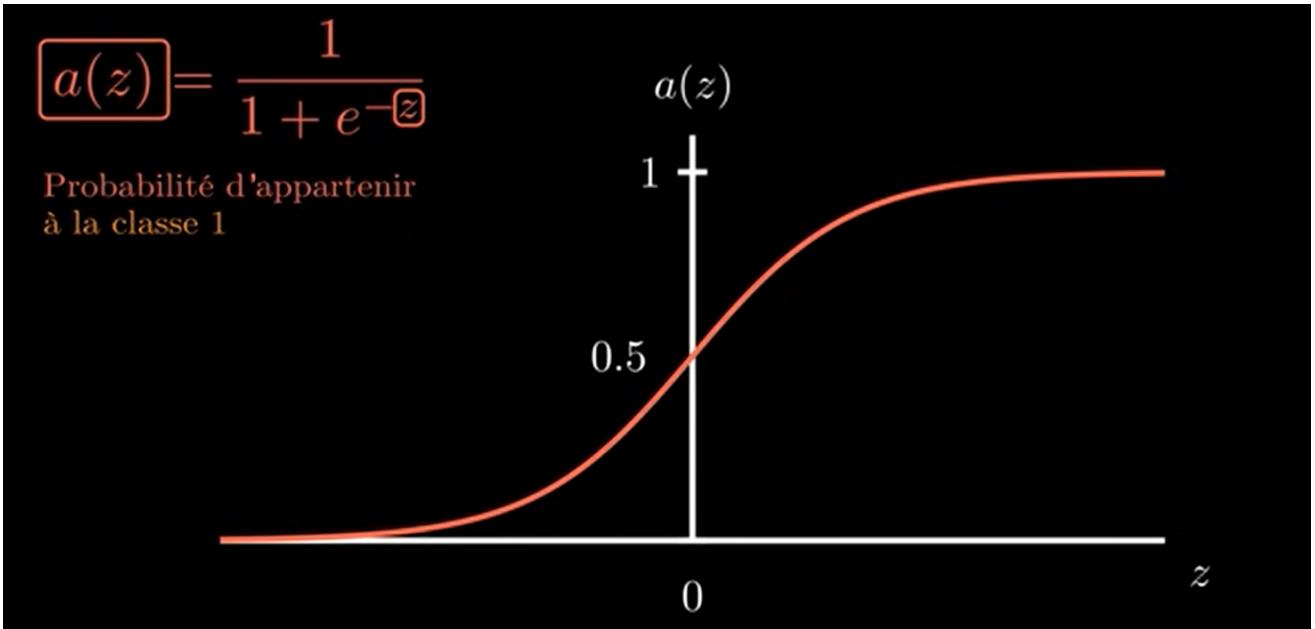




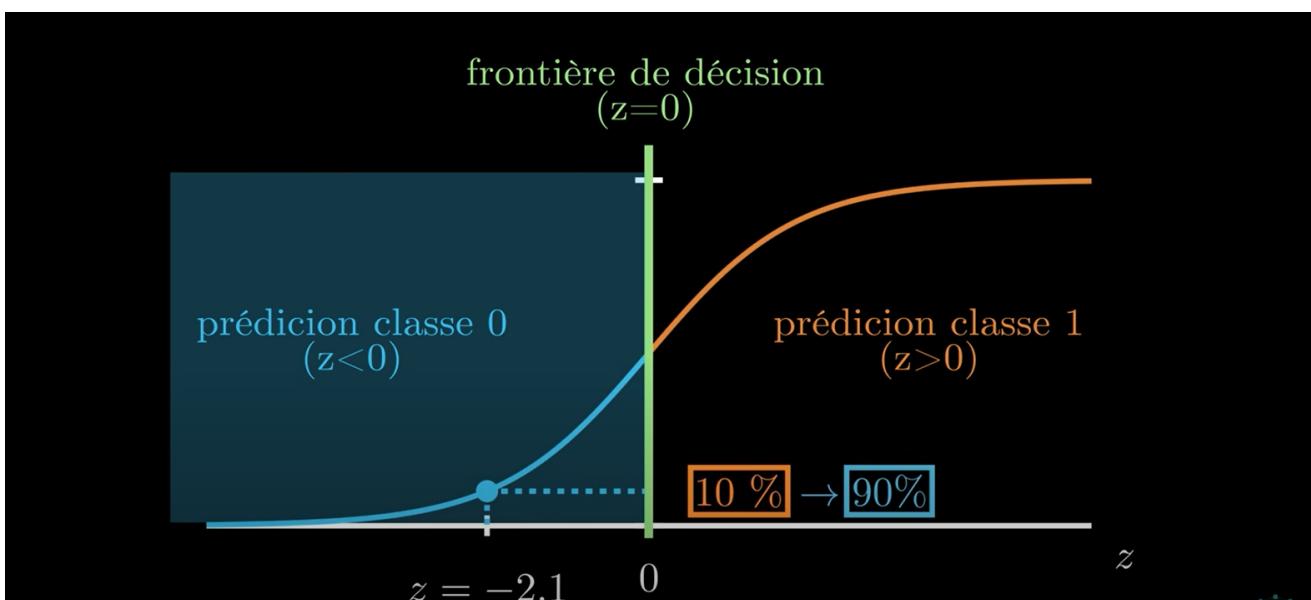
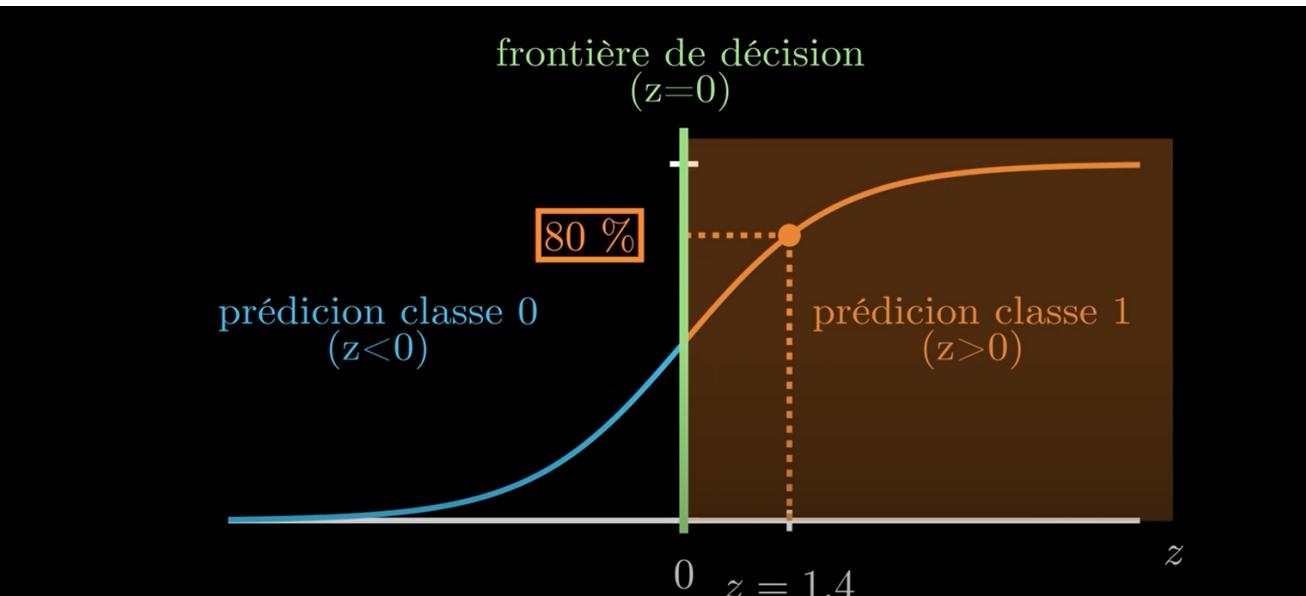
Fonction sigmoïde

Cette fonction qui permet d'effectuer cette action est appelée la fonction **sigmoïde(Logistique)**.

Cette permet de convertir la sortie z en une probabilité $a(z)$



Par exemple



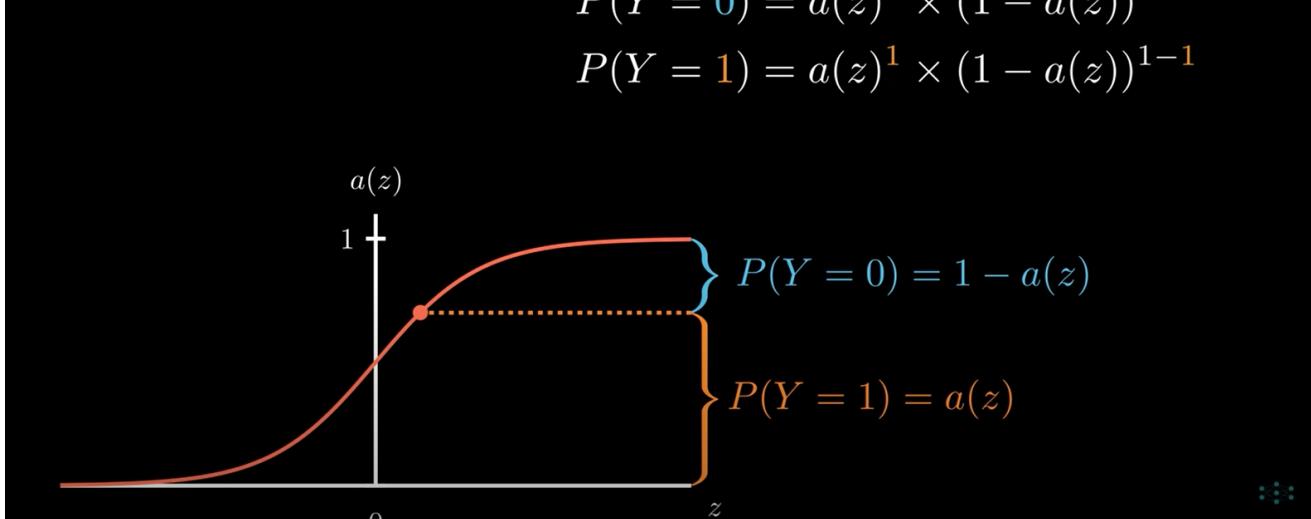
La loi de Bernoulli

Loi de Bernoulli :

$$P(Y = y) = a(z)^y \times (1 - a(z))^{1-y}$$

$$P(Y = 0) = a(z)^0 \times (1 - a(z))^{1-0}$$

$$P(Y = 1) = a(z)^1 \times (1 - a(z))^{1-1}$$



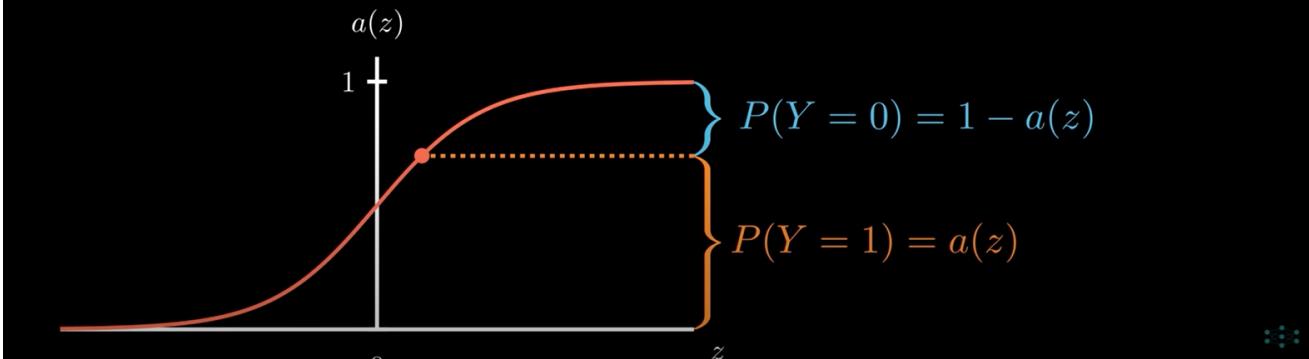
Loi de Bernoulli :

$$P(Y = y) = a(z)^y \times (1 - a(z))^{1-y}$$

$$P(Y = 0) = (1 - a(z))^{1-0}$$

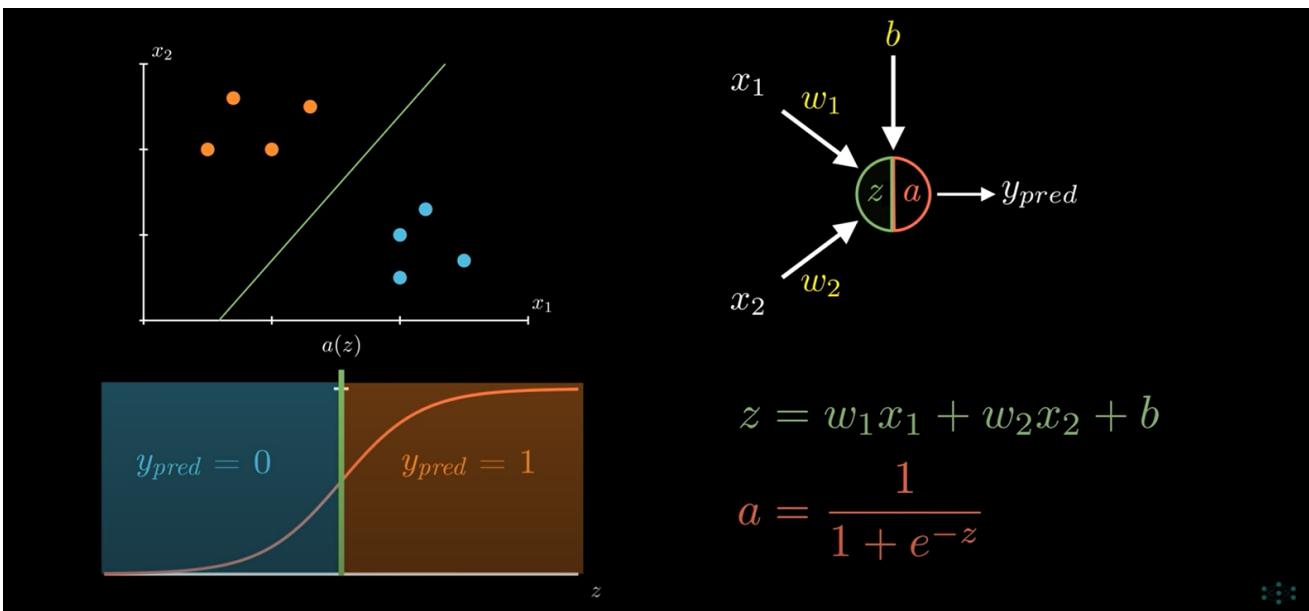
$$P(Y = 1) = a(z)^1$$

(rappel : $x^0 = 1$)



Résumé

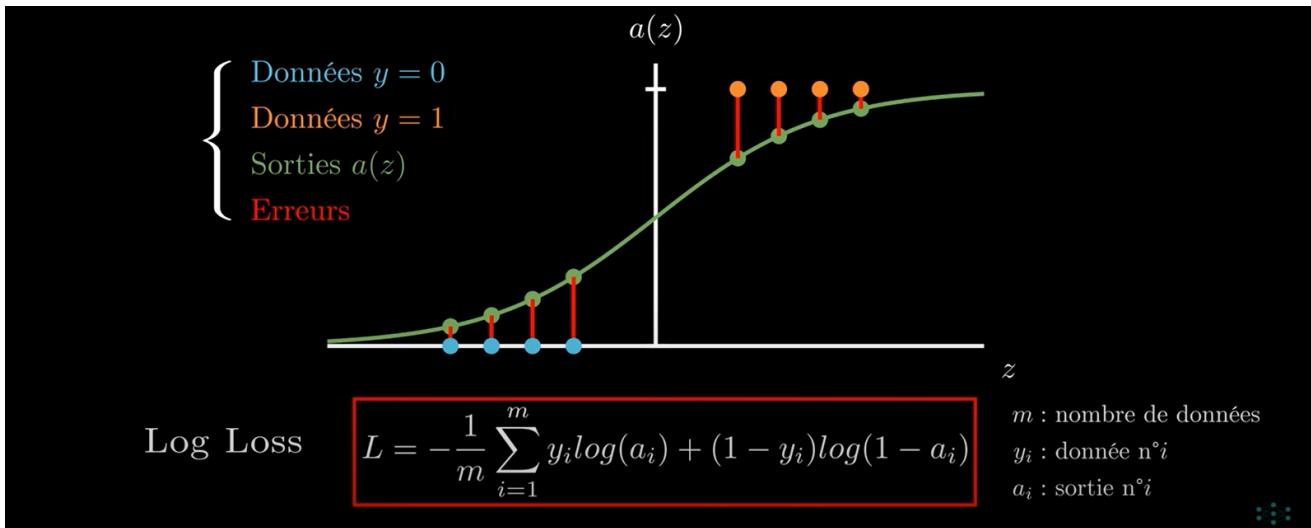
Pour résumer tout ce qu'on vient de voir, tout ce qui se trouve à l'intérieur d'un réseau de neurone ($z = w_1 * x_1 + w_2 * x_2 + b$), suivi d'une fonction d'activation $a(z) = 1 / (1 + e^{-z})$ qui nous retourne une probabilité suivant la loi de Bernoulli.



Fonction Coût (MAXIMUM DE VRAISEMBLANCE)

En machine Learning une fonction coût(Loss Function), c'est une fonction qui permet de quantifier les **erreurs effectuées** par un modèle.

Dans notre cas, c'est une fonction qui calcule la différence entre y réelle et y_{pred} de notre modèle.



Origine de la fonction coût(démonstration)

La Vraisemblance

Elle nous indique la plausibilité du modèle vis-à-vis de vraies données.

Analogie :

Une histoire est vraisemblable lorsqu'elle est en accord avec des faits qui se sont vraiment déroulés



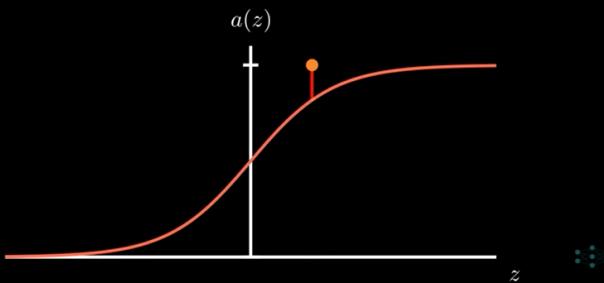
Par exemple

Si une plante est existe et elle est toxique c'est à dire $y = 1$



$$y = 1$$
$$P(y=1) = 0.8$$

Vraisemblance : 80%
i.e. Proche à 80% de la réalité



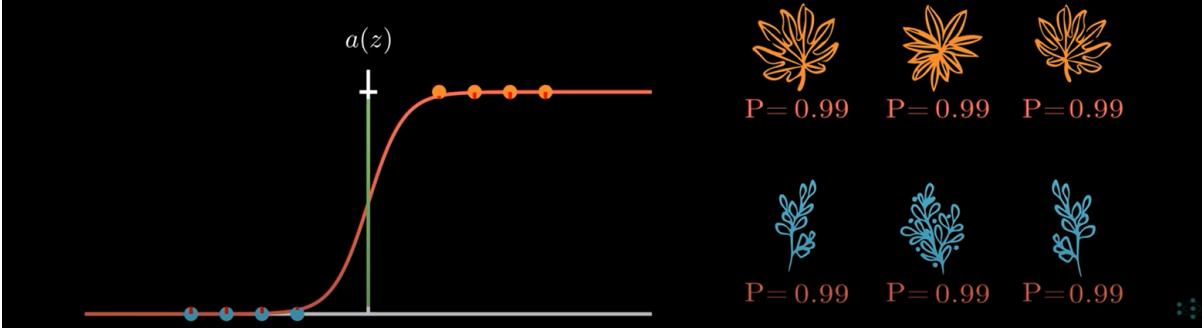
La Formule de Vraisemblance

Likelihood :

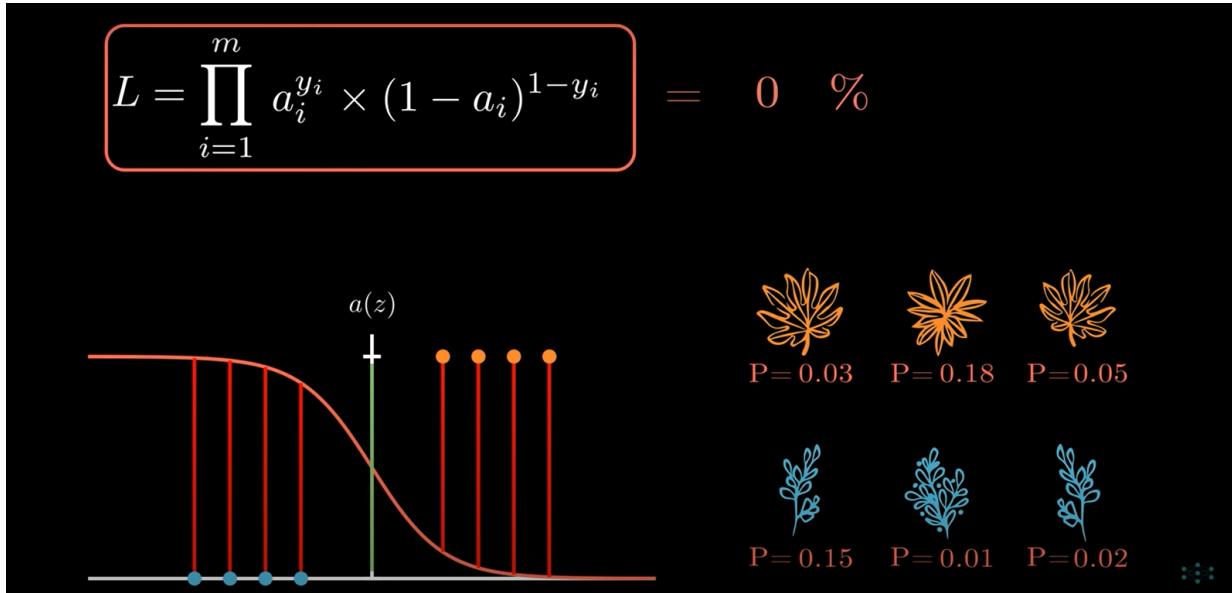
$$L = \prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}$$

- Si le résultat de cette formule est égale à 100% cela veut dire que notre modèle est vraisemblable à 100%

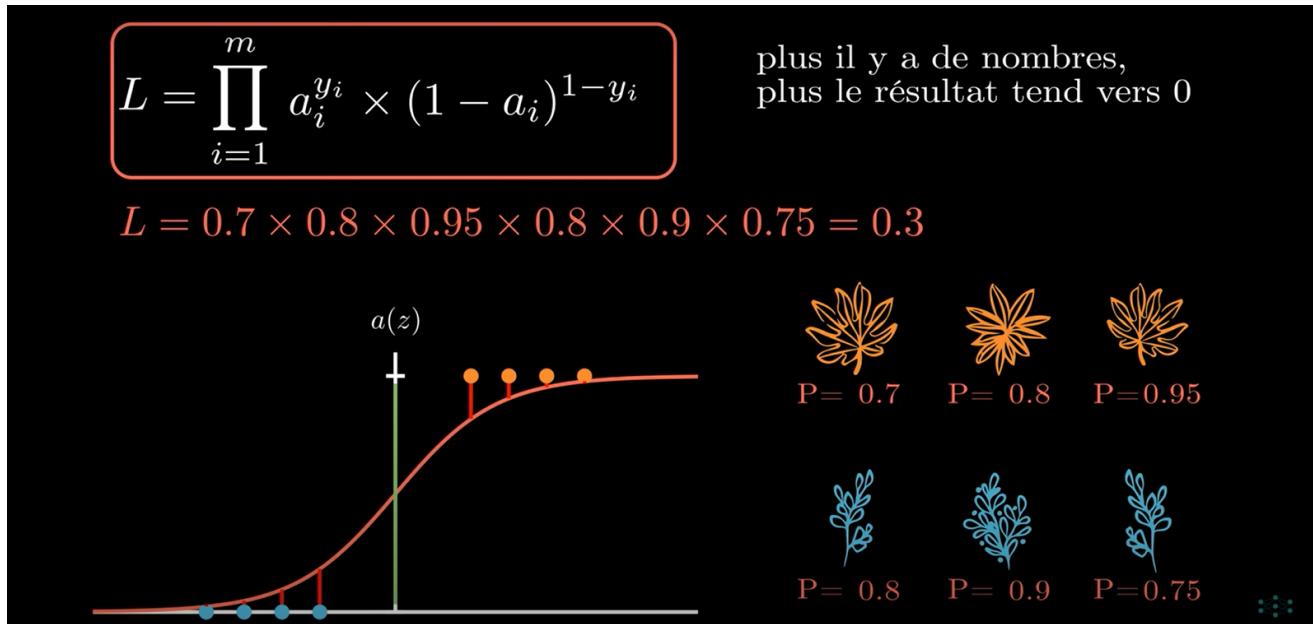
$$L = \prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i} = 99\% \quad \text{...}$$



- Si le résultat de cette formule est égale à 0% cela veut dire que notre modèle est fortement invraisemblable.



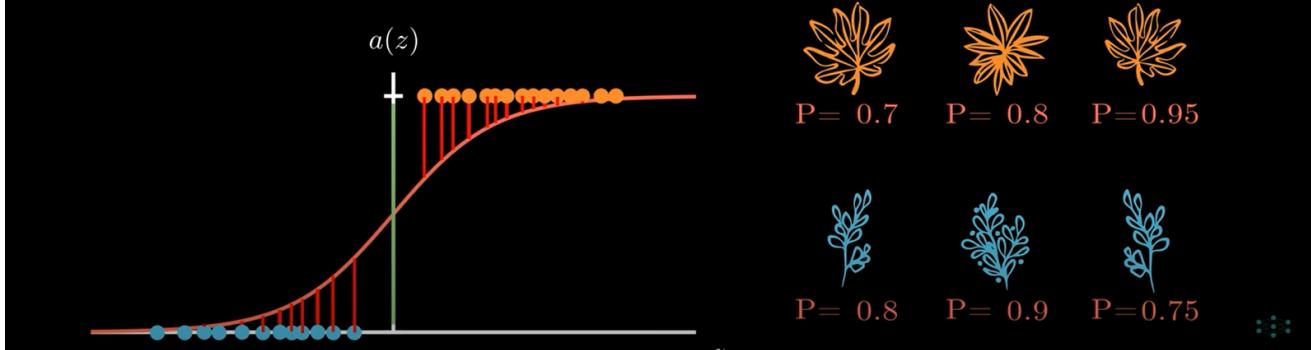
Problème de cette formule



En pratique lorsqu'on calcule la vraisemblance des volumes importants de données on se rend compte que le résultat quasiment presque égal à zéro (0). Et là ça devient un problème

$$L = \prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}$$

$$L = 0.7 \times 0.8 \times 0.95 \times 0.8 \times 0.9 \times 0.75 \times \dots \times 0.8 = 0.000\dots$$



Solution

Pour rémedier à ce problème, on applique la fonction logarithme à la formule de Bernoulli.

Et cela nous donne un résultat plus lisible au paravant.

$$L = \prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}$$

$$\log(ab) = \log(a) + \log(b)$$

$$L = 0.7 \times 0.8 \times 0.95 \times 0.8 \times 0.9 \times 0.75 \times \dots \times 0.8 = 0.000\dots$$

$$\log(L) = \log\left(\prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}\right)$$

$$= \log(0.7 \times 0.8 \times 0.95 \times 0.8 \times 0.9 \times 0.75)$$

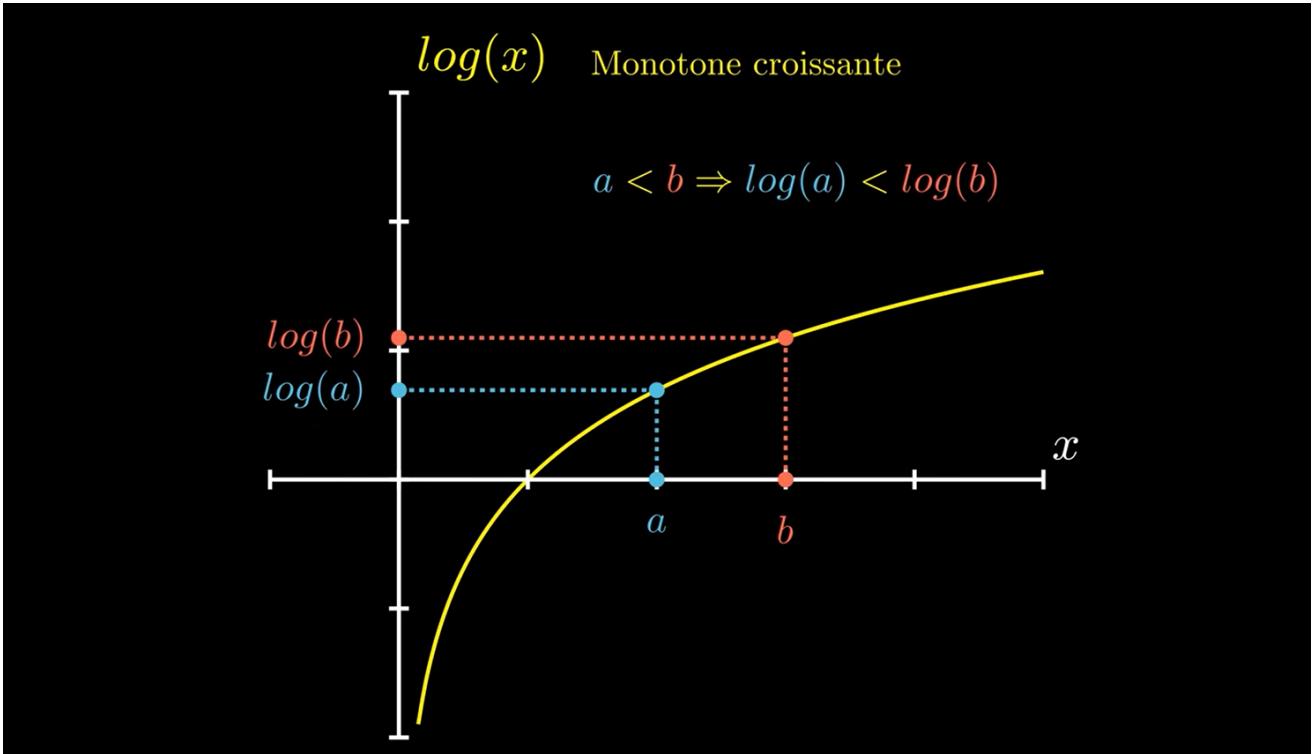
$$= \log(0.7) + \log(0.8) + \log(0.95) + \log(0.8) + \log(0.9) + \log(0.75)$$

$$= -0.35 - 0.22 - 0.05 - 0.22 - 0.1 - 0.35$$

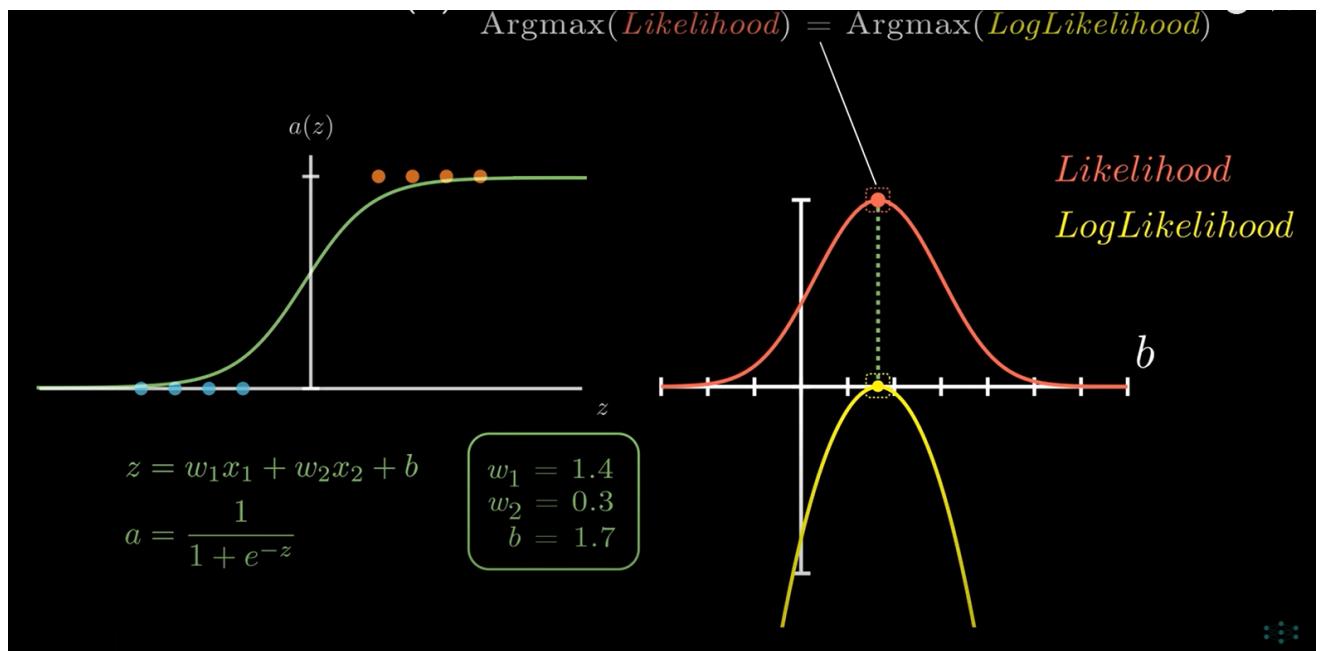
$$= -1.24$$

⋮

Ce pendant, on pourrait penser que la fonction logarithme donne un autre sens à notre calcul mais non ce n'est pas le cas car la fonction logarithme est une fonction monotone croissante donc elle conserve l'ordre de nos termes.



Cela signifie que lorsqu'on cherche le maximum de notre vraisemblance, il nous suffira de chercher le maximum de notre vraisemblance et comme vous pouvez le constater dans ce graphique ça retournera le même résultat. Alors par conclusion on peut utiliser le log de la vraisemblance pour continuer notre calcule.



Démonstration

$$\begin{aligned}
LL &= \log\left(\prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}\right) \\
&= \sum_{i=1}^m \log(a_i^{y_i} \times (1 - a_i)^{1-y_i}) \\
&= \sum_{i=1}^m \log(a_i^{y_i}) + \log((1 - a_i)^{1-y_i}) \\
&= \boxed{\sum_{i=1}^m y_i \log(a_i) + (1-y_i) \log(1 - a_i)} \quad \text{Rappel : } \log(ab) = \log(a) + \log(b) \\
&\qquad\qquad\qquad \text{Rappel : } \log(a^y) = y \log(a)
\end{aligned}$$

Ce résultat ressemble beaucoup au résultat de **Log Loss** à la différence de $-1/m$

$$\begin{aligned}
LL &= \log\left(\prod_{i=1}^m a_i^{y_i} \times (1 - a_i)^{1-y_i}\right) \\
&= \sum_{i=1}^m \log(a_i^{y_i} \times (1 - a_i)^{1-y_i}) \\
&= \sum_{i=1}^m \log(a_i^{y_i}) + \log((1 - a_i)^{1-y_i}) \\
&= \boxed{\sum_{i=1}^m y_i \log(a_i) + (1-y_i) \log(1 - a_i)} \quad \text{Rappel : } \log(ab) = \log(a) + \log(b) \\
&\qquad\qquad\qquad \text{Rappel : } \log(a^y) = y \log(a) \\
&\boxed{\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m y_i \log(a_i) + (1 - y_i) \log(1 - a_i)} \quad \text{Log Loss} \quad \dots
\end{aligned}$$

Mais ce qu'on a fait là c'est juste le calcul du log de la vraisemblance et une vraisemblance on cherche à la maximiser pour avoir le meilleur modèle possible le modèle le plus vraisemblable or en mathématique les algorithmes de maximisation n'existe pas vraiment à la place on utilise plutôt des algorithmes de minimisation

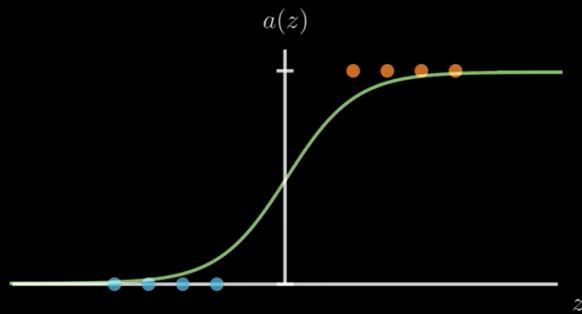
Alors

Maintenant vous savez l'origine de notre fonction **Log Loss**

Log Loss

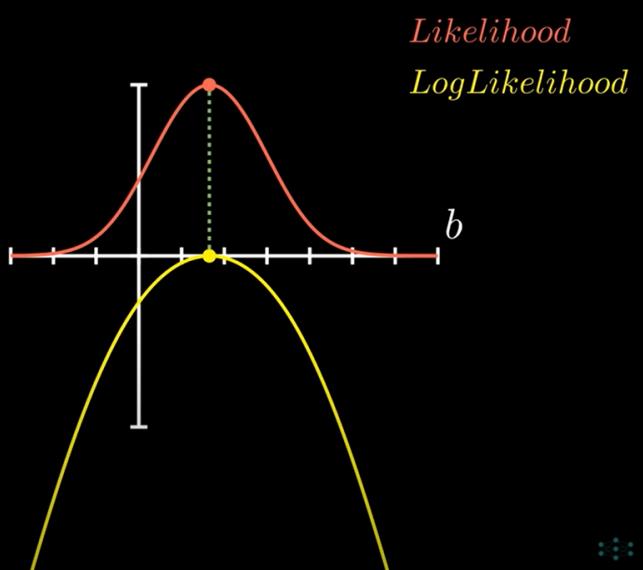
$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m y_i \log(a_i) + (1 - y_i) \log(1 - a_i)$$

Maximiser la vraisemblance L
en minimisant la fonction $-\log(L)$



Rappelez-vous :
On veut avoir le modèle
le plus vraisemblable possible !

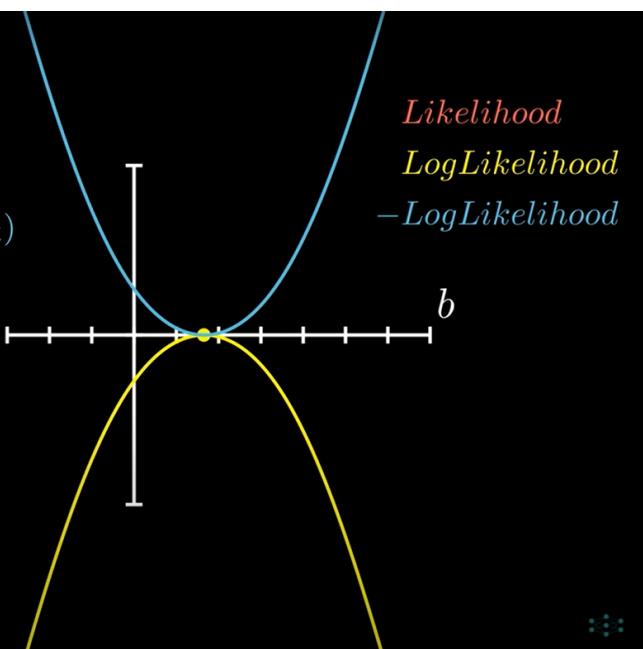
Problème : pas d'algorithme de
maximisation !



Mais le problème n'est pas là, car maximiser une fonction revient à la fonction $-f(x)$, c'est pourquoi pour maximiser notre fonction on revient minimiser sa fonction négative d'où le signe moins (-) et $1/m$ c'est un but de normalisation de notre fonction.

Maximiser $f(x) =$ Minimiser $-f(x)$

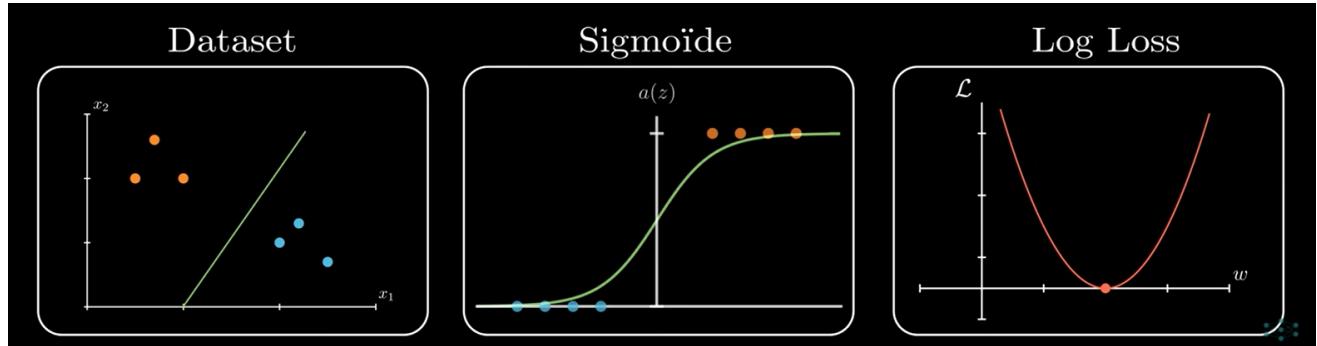
$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m y_i \log(a_i) + (1 - y_i) \log(1 - a_i)$$



Algorithme de la DESCENTE DE GRADIENT

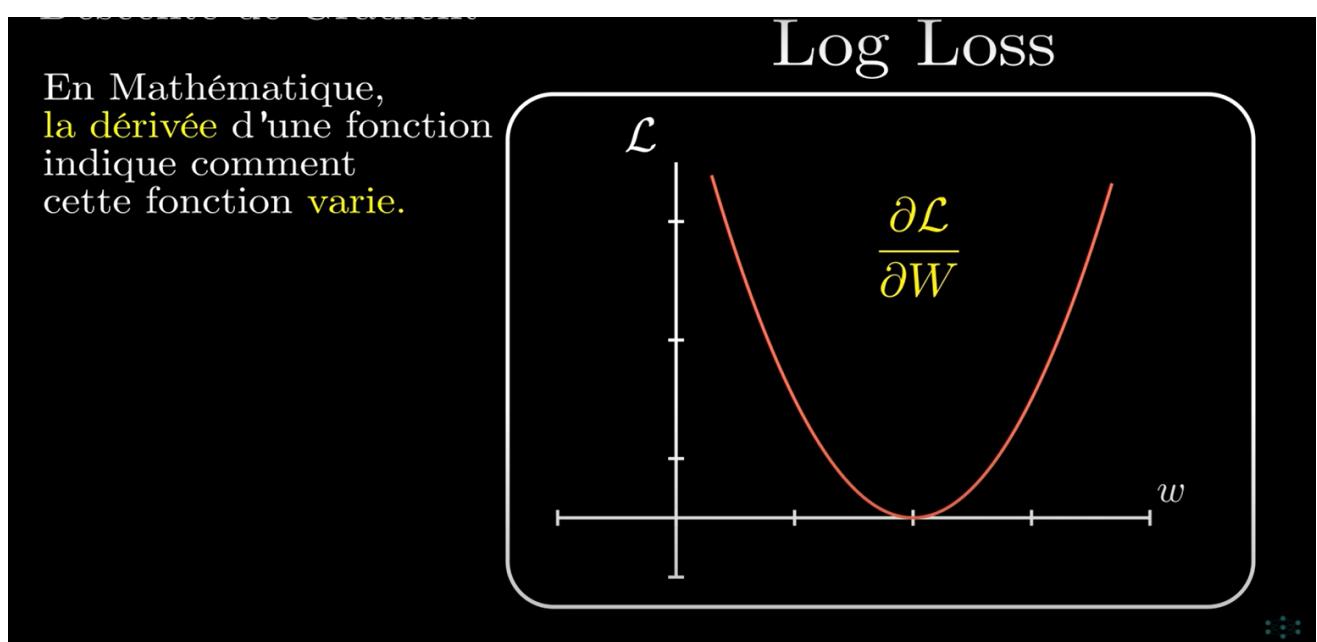
La Descente de gradient est l'un des algorithmes les plus utilisés en **Machine Learning** et en **Deep Learning**.

Il consiste à ajuster les paramètres W et b de façon à minimiser les erreurs du modèle et, c'est à dire à **minimiser la fonction coût (Log Loss)**.



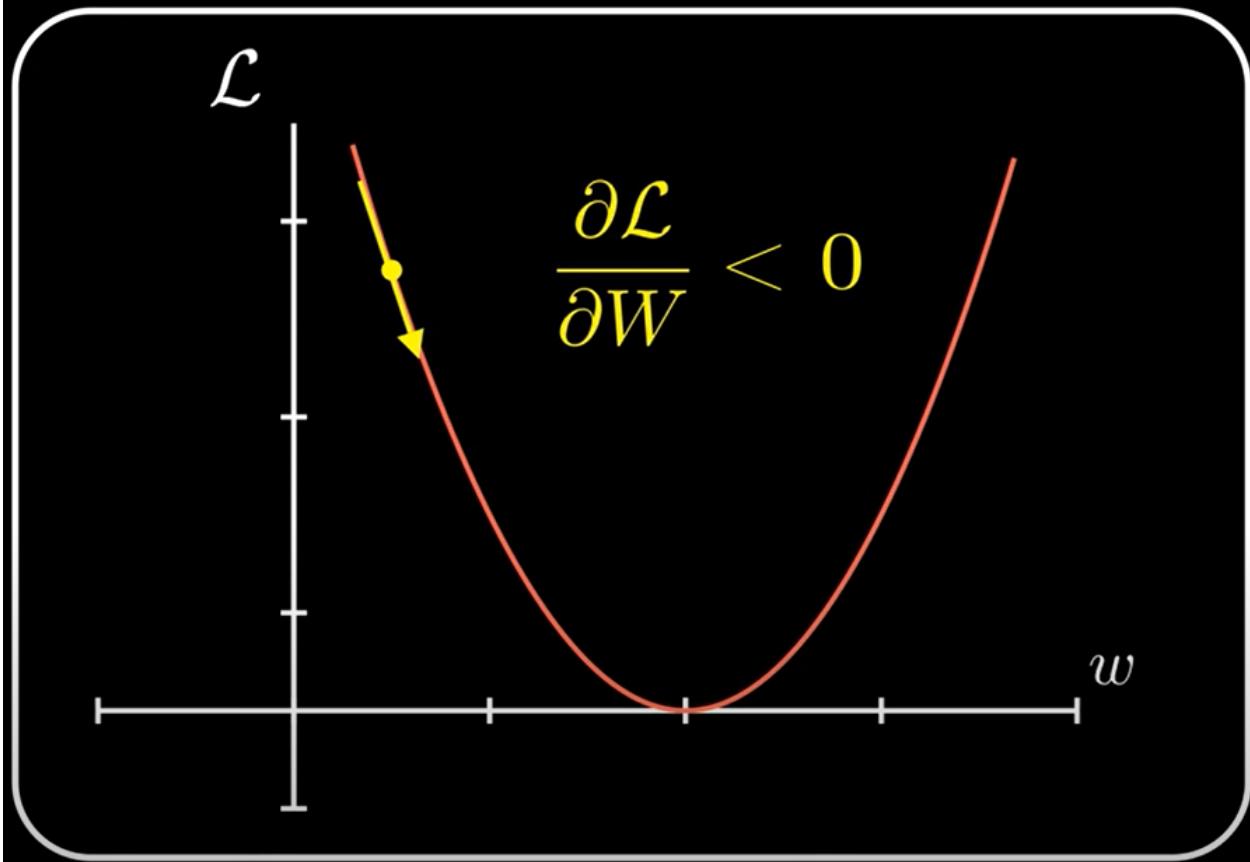
Pour ça, il faut déterminer comment est ce que cette fonction varie en fonction des différents paramètres.

C'est pourquoi on calcule le **Gradient** (ou la **dérivée**) de la **Fonction Coût**.



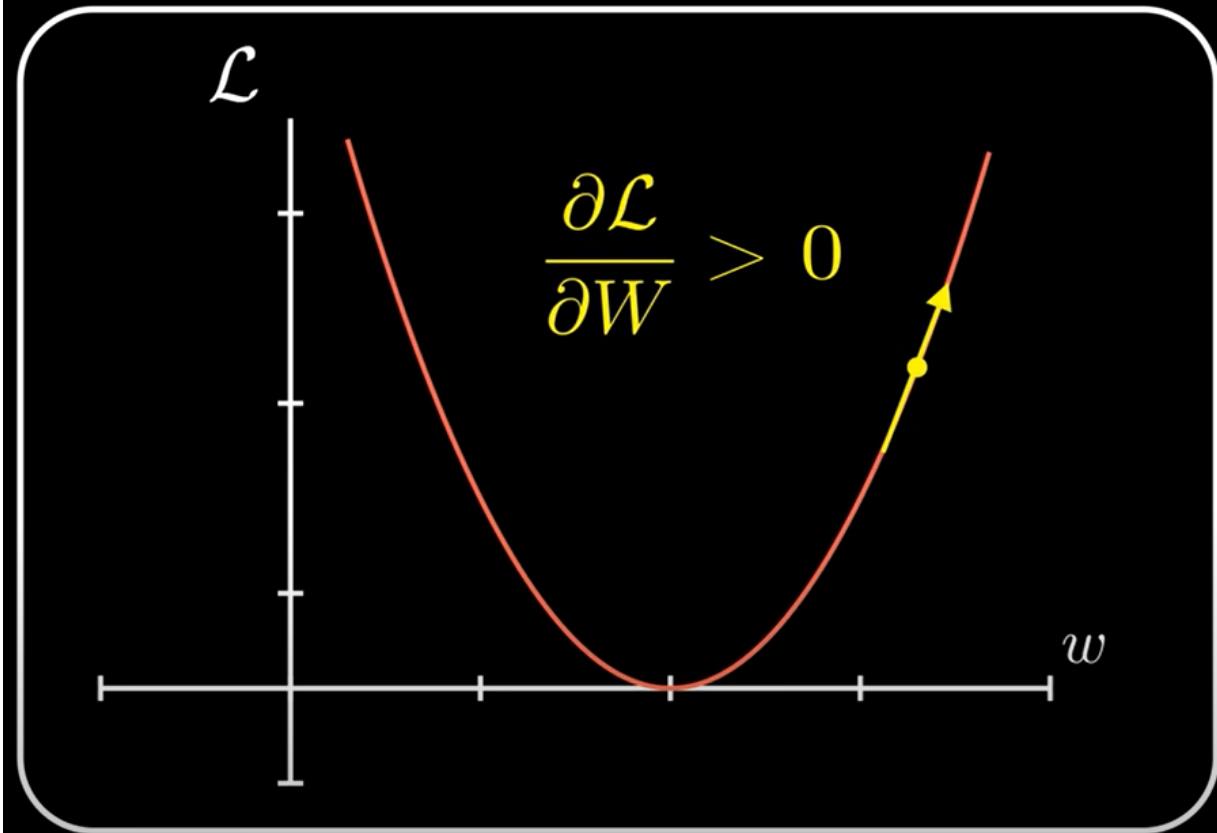
- Si la dérivée est négative ça nous indique la fonction diminue quand W augmente qu'il va falloir donc augmenter W si on veut réduire nos erreurs.

Log Loss



- Si la dérivée est positive ça nous indique la fonction augmente quand W augmente qu'il va falloir donc diminuer W si on veut réduire nos erreurs.

Log Loss



- Pour faire cela, on va utiliser la formule suivante:

Log Loss

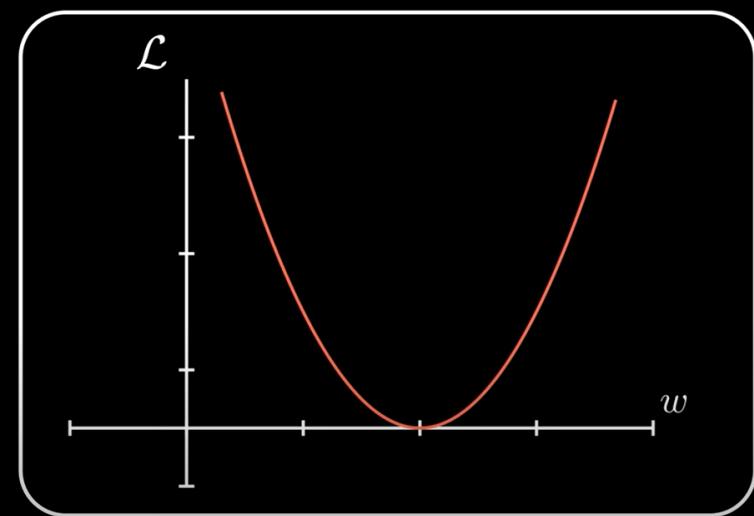
$$W_{t+1} = W_t - \alpha \frac{\partial L}{\partial W_t}$$

W_{t+1} : Paramètre W à l'instant $t+1$

W_t : Paramètre W à l'instant t

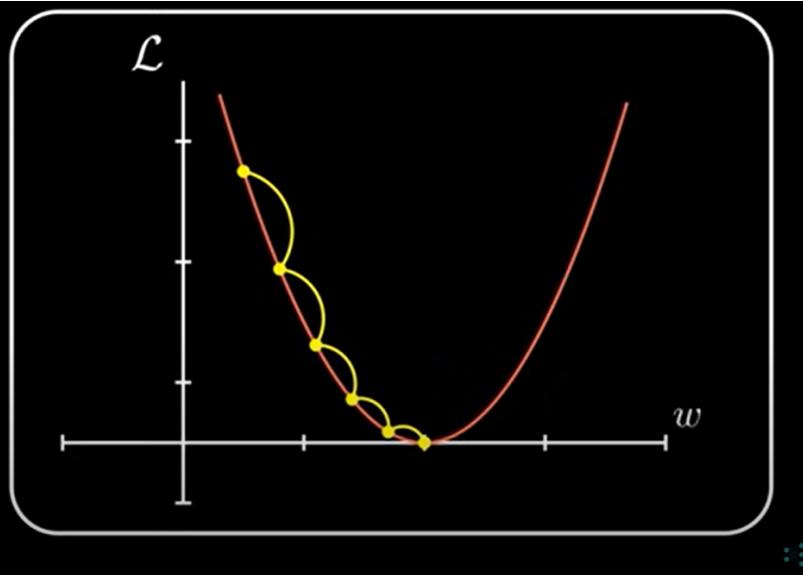
α : Pas d'apprentissage positif

$\frac{\partial L}{\partial W_t}$: Gradient à l'instant t



- En repétant cette formule on obtient cette figure, d'où le nom **Descente Gradient**

$$W_{t+1} = W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

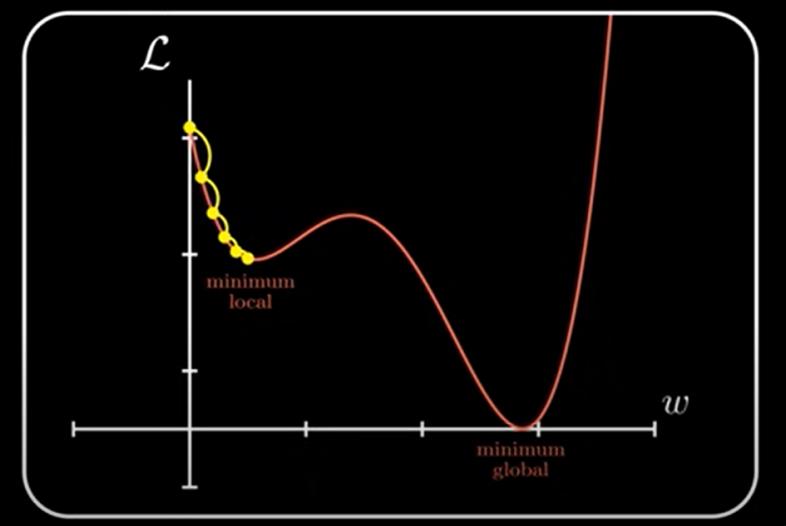


- Alors que ça marche il faut que notre fonction soit convexe.

$$W_{t+1} = W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

Fonction Convexe :
qui ne contient qu'un seul
minimum.

Par exemple, la fonction
à droite n'est pas convexe.



Exercice

Calculez les gradients suivants, qui nous seront indispensables pour coder notre premier neurone.

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i) x_1$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i) x_2$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i)$$

Cet exercice est DIFFICILE !
Ne vous démotivez pas
si vous ne trouvez pas la solution.

Indices :

1. le gradient peut être décomposé en une chaîne de dérivées partielles :

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w_1}$$

2. pour calculer la dérivée de $a(z)$, formulez-la comme une fonction composée :

$$h = g \circ f \quad g(x) = \frac{1}{x} \quad f(x) = 1 + e^{-x}$$

3. rappel dérivée d'une fonction composée :

$$h' = g' \circ f$$

$$h'(x) = g'(f(x)) \times f'(x)$$