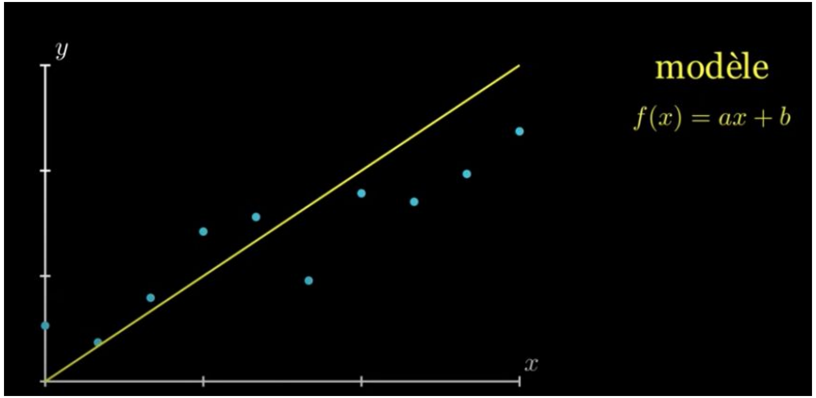
Deep Learning

les bases

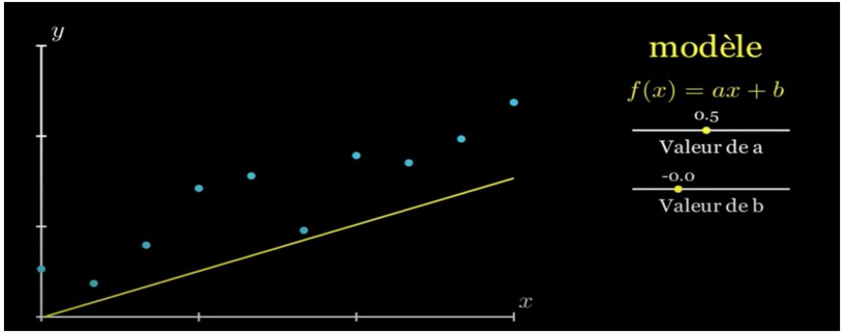
Machine Learning

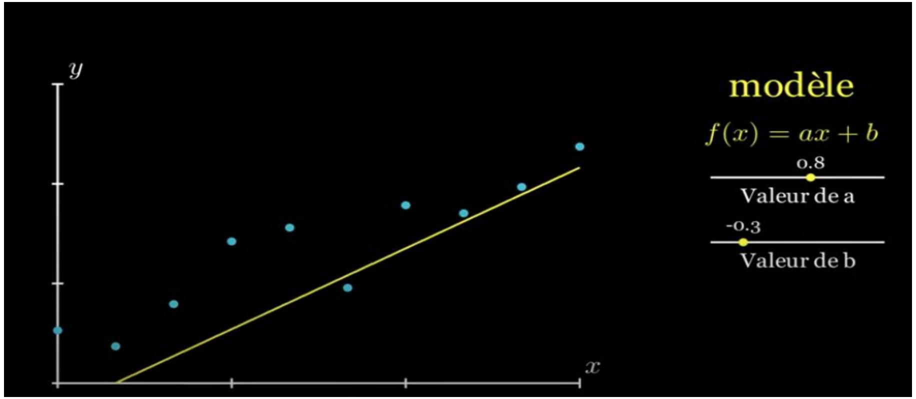
- Le Machine Learning est un domaine d’intelligence artificielle permettant aux machines d’apprendre sans avoir été préalablement programmées spécifiquement.
- Pour ça nous avons besoin d'exemples d'apprentissages (données)
- Les modèles sont donc entrainer à partir des exemples



Machine Learning

L'entraînement consisté à trouver les paramètres qui donnent le meilleur modèle

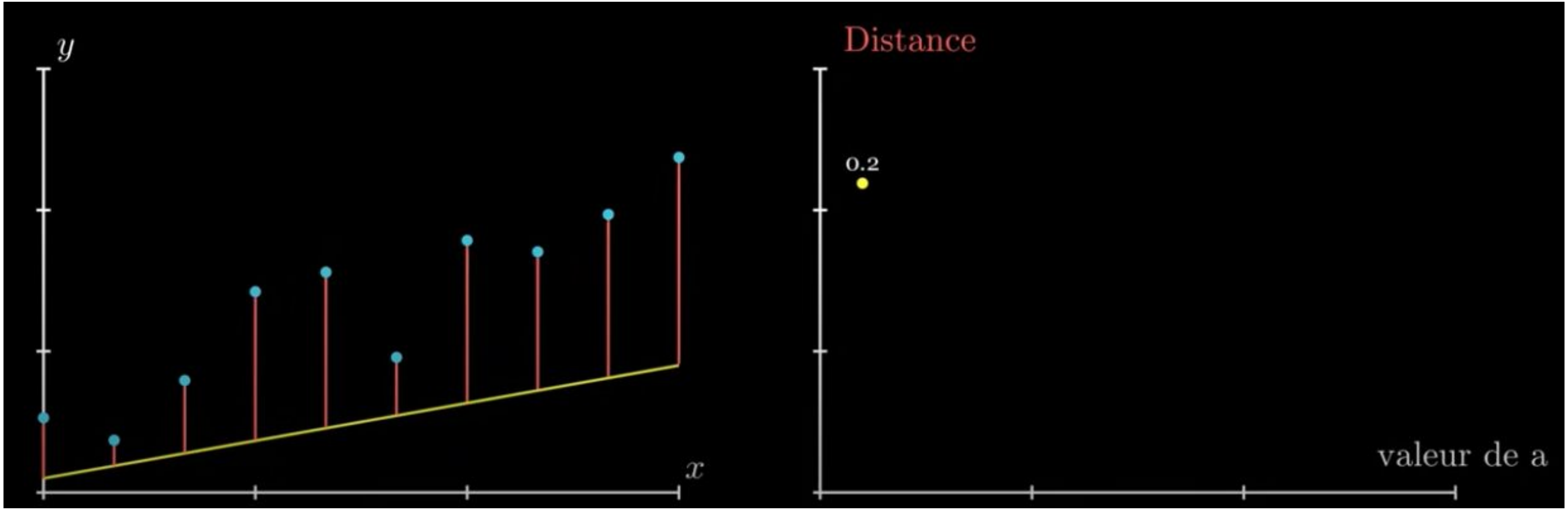




Le modèle qui s'ajuste meilleur à nous données

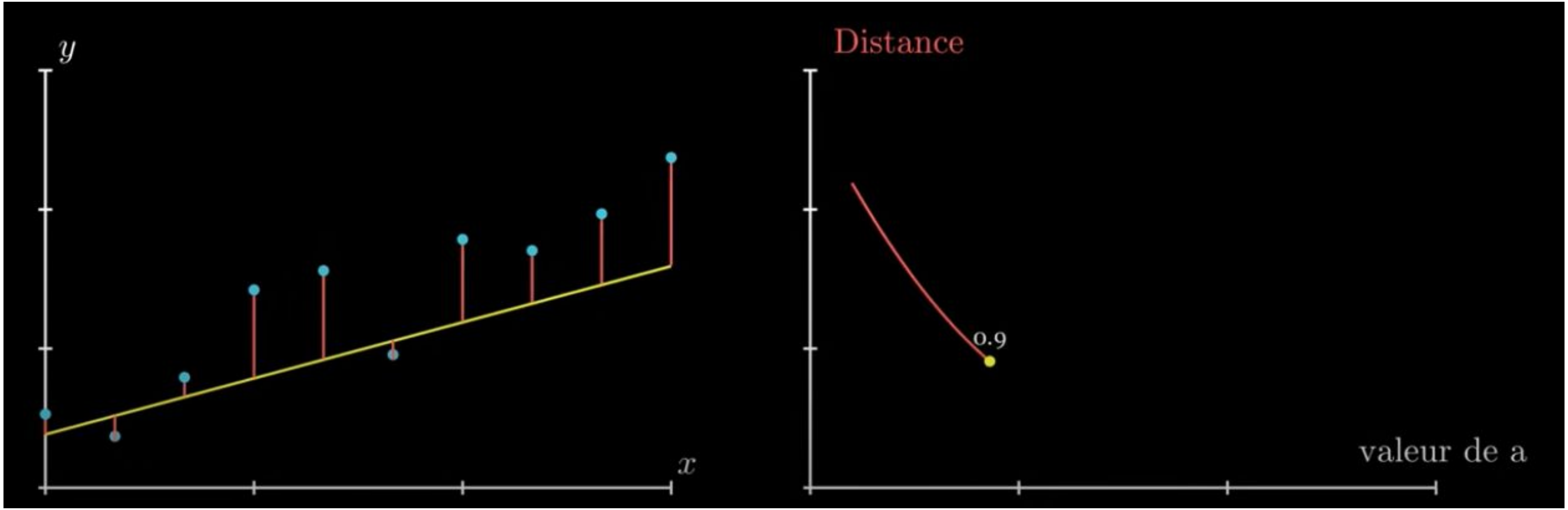
Machine Learning

Pour cela nous programmons un algorithme d'optimisation pour tester différentes valeurs de a et b jusqu’à trouver la combinaison qui minimise la distance entre le modèle et les exemples.



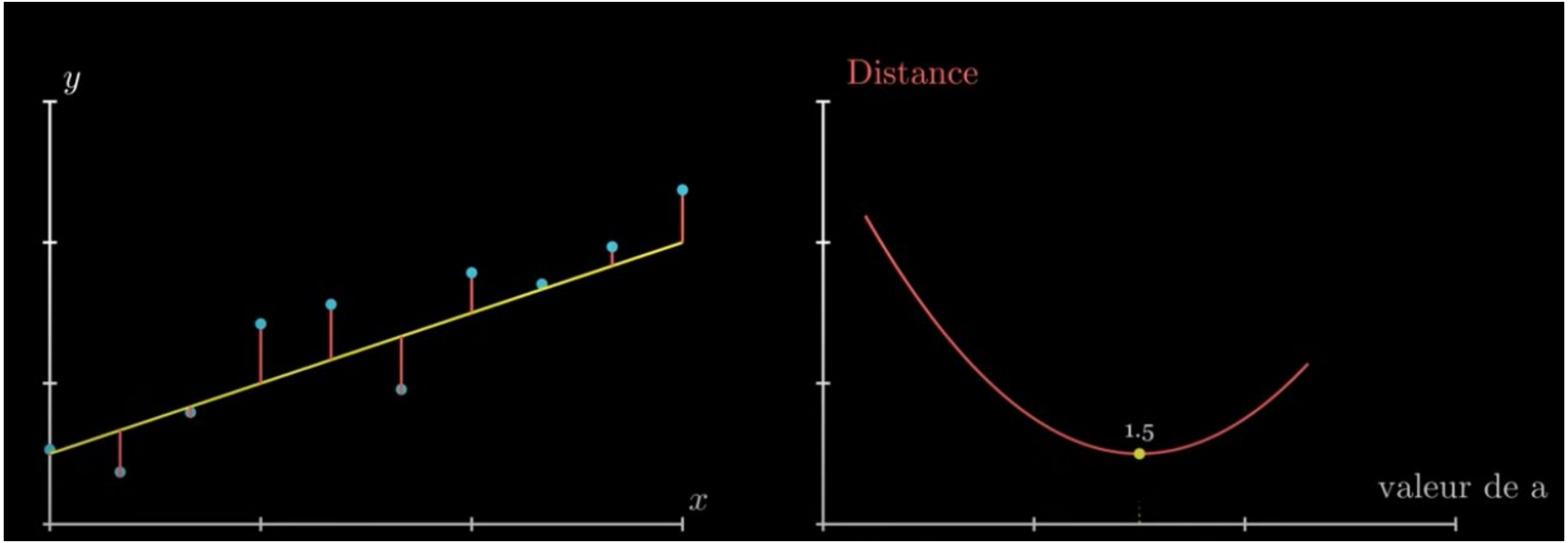
Machine Learning

Pour cela nous programmons un algorithme d'optimisation pour tester différentes valeurs de a et b jusqu'à trouver la combinaison qui minimise la distance entre le modèle et les exemples.



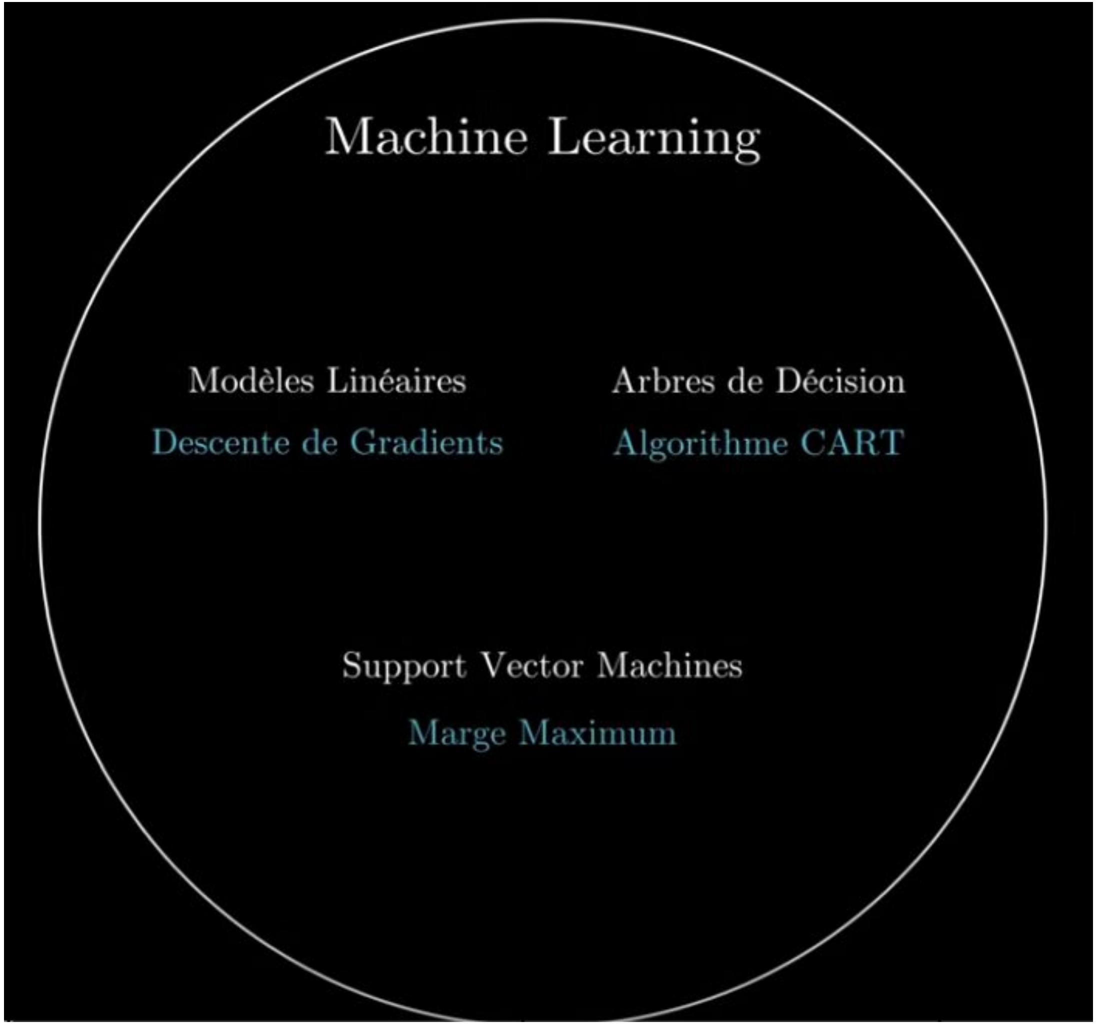
Machine Learning

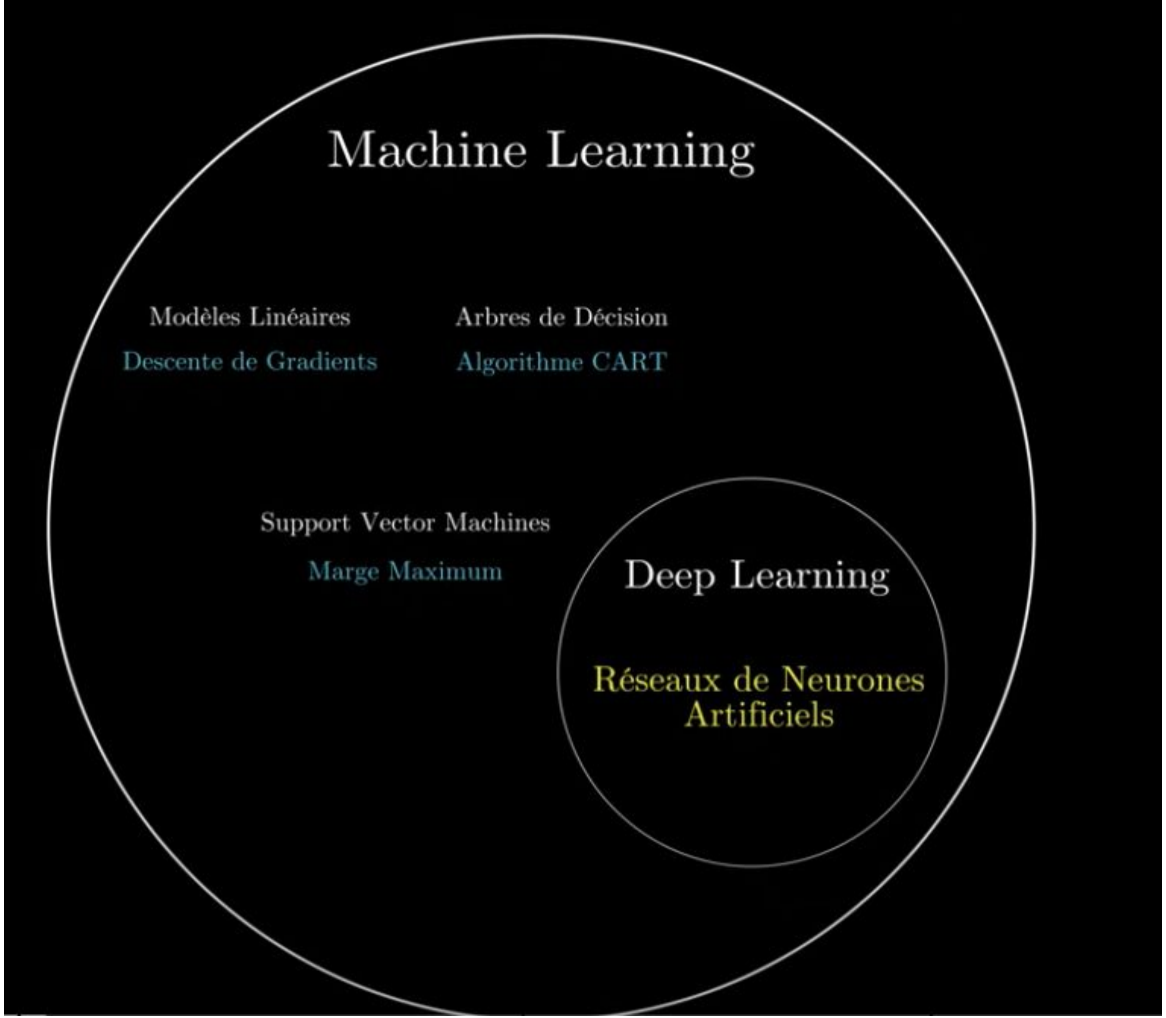
Pour cela nous programmons un algorithme d'optimisation pour tester différentes valeurs de a et b jusqu'à trouver la combinaison qui **minimise la distance** entre le modèle et les exemples.

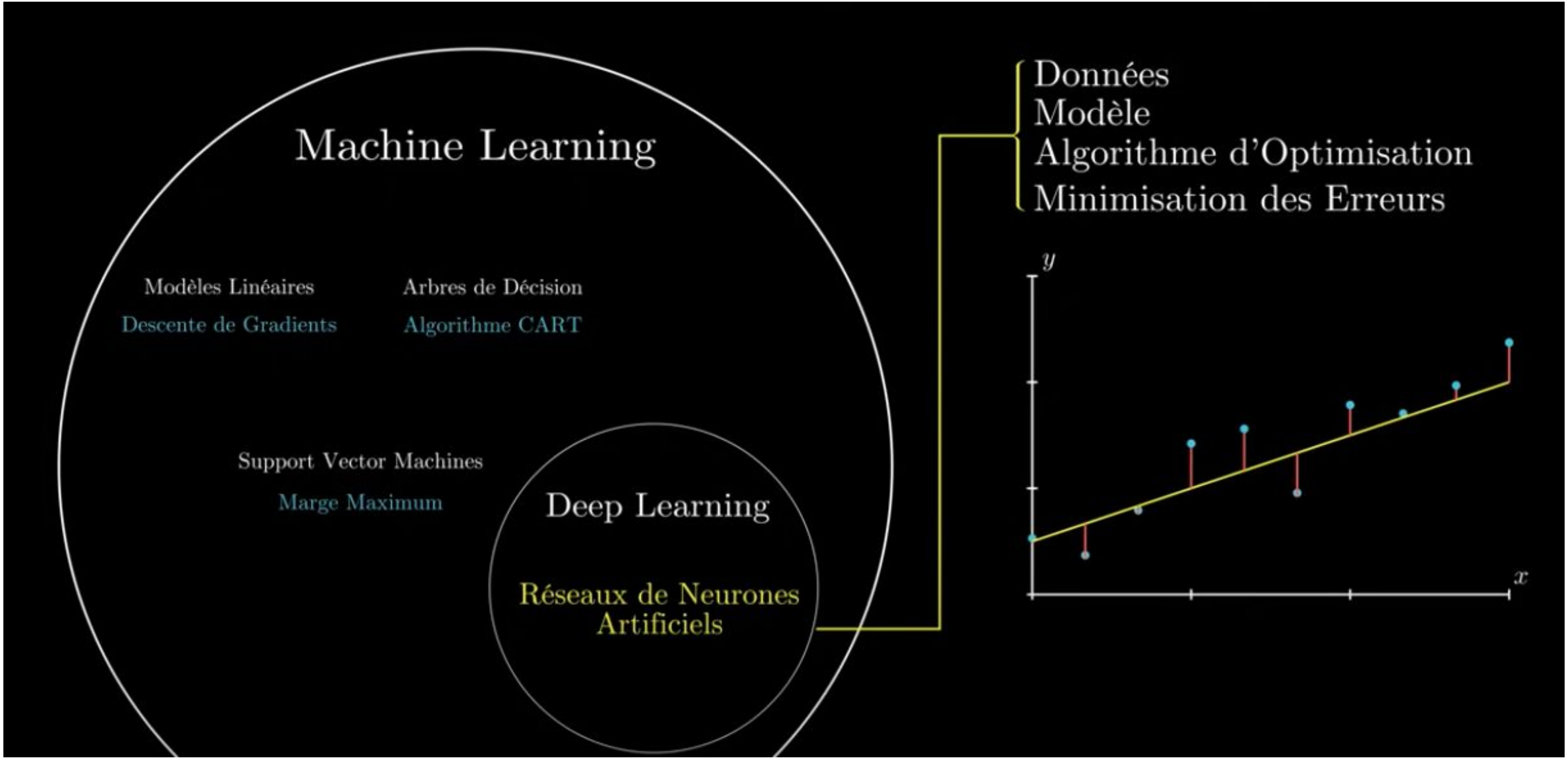


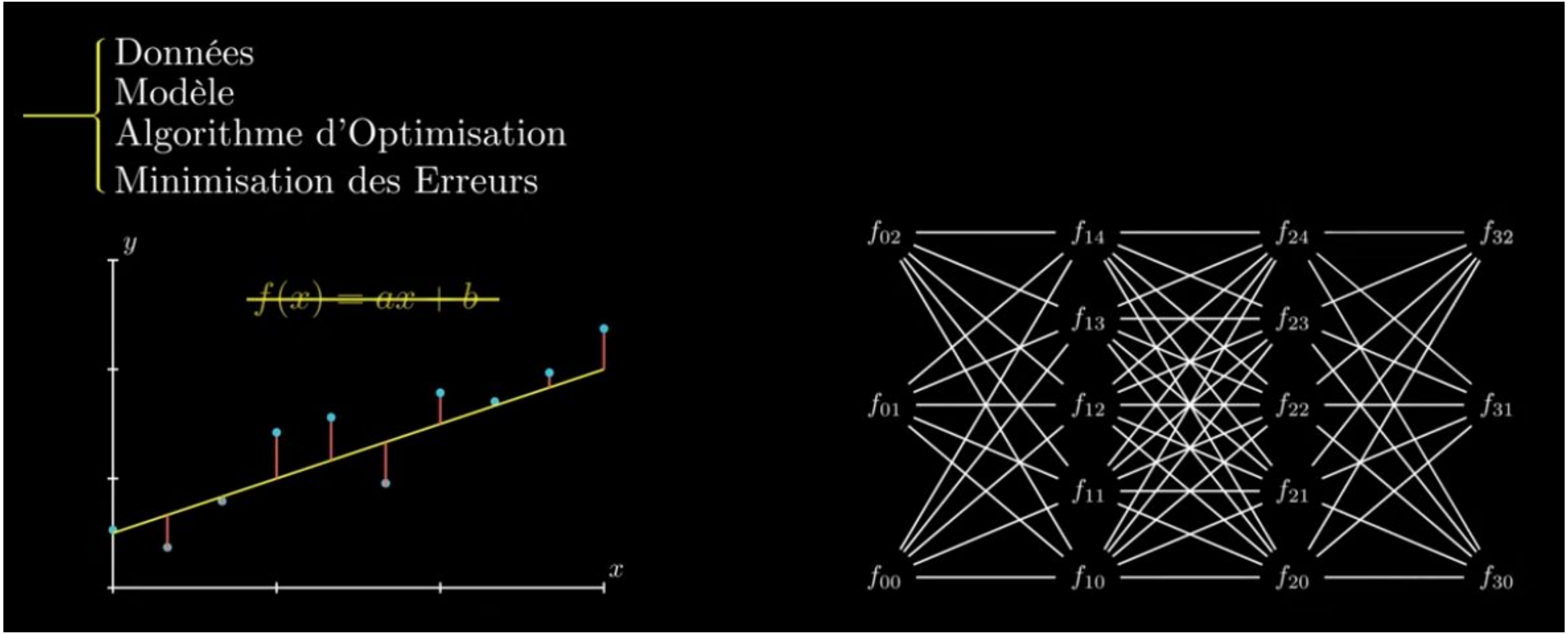
Machine Learning

Développer un **modèle**, utilisant un **algorithme d'optimisation** pour **minimiser les erreurs** entre le modèle et les données

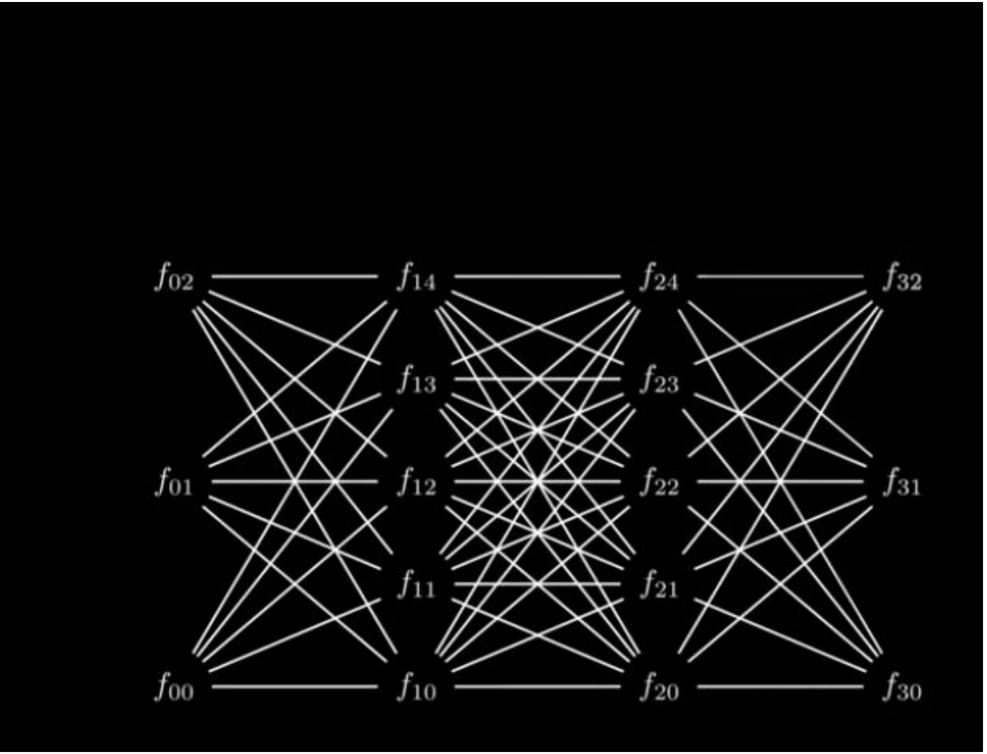

Machine Learning
Modèles Linéaires
Descente de Gradients
Arbres de Décision
Algorithme CART
Support Vector Machines
Marge Maximum


Machine Learning
Modèles Linéaires
Descente de Gradients
Arbres de Décision
Algorithme CART
Support Vector Machines
Marge Maximum
Deep Learning
Réseaux de Neurones Artificiels

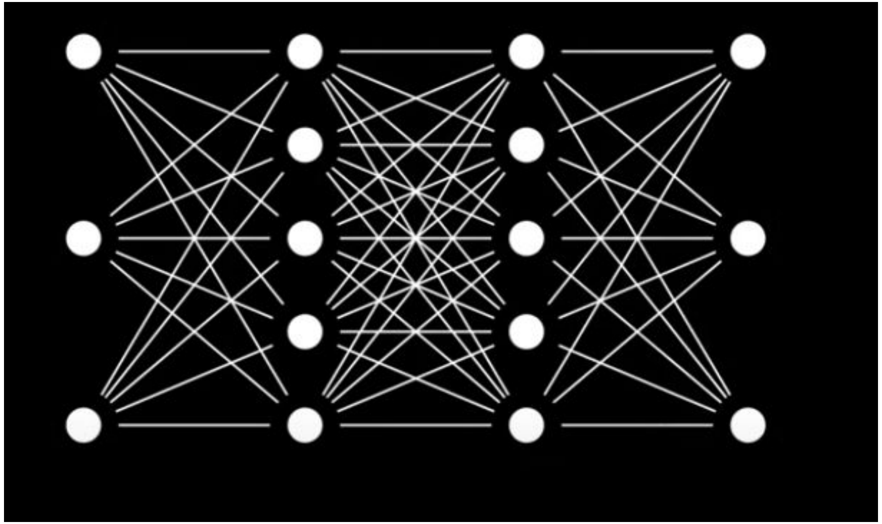

Machine Learning
Modèles Linéaires
Descente de Gradients
Arbres de Décision
Algorithme CART
Support Vector Machines
Marge Maximum
Deep Learning
Réseaux de Neurones
Artificiels
Données
Modèle
Algorithme d'Optimisation
Minimisation des Erreurs
y
x



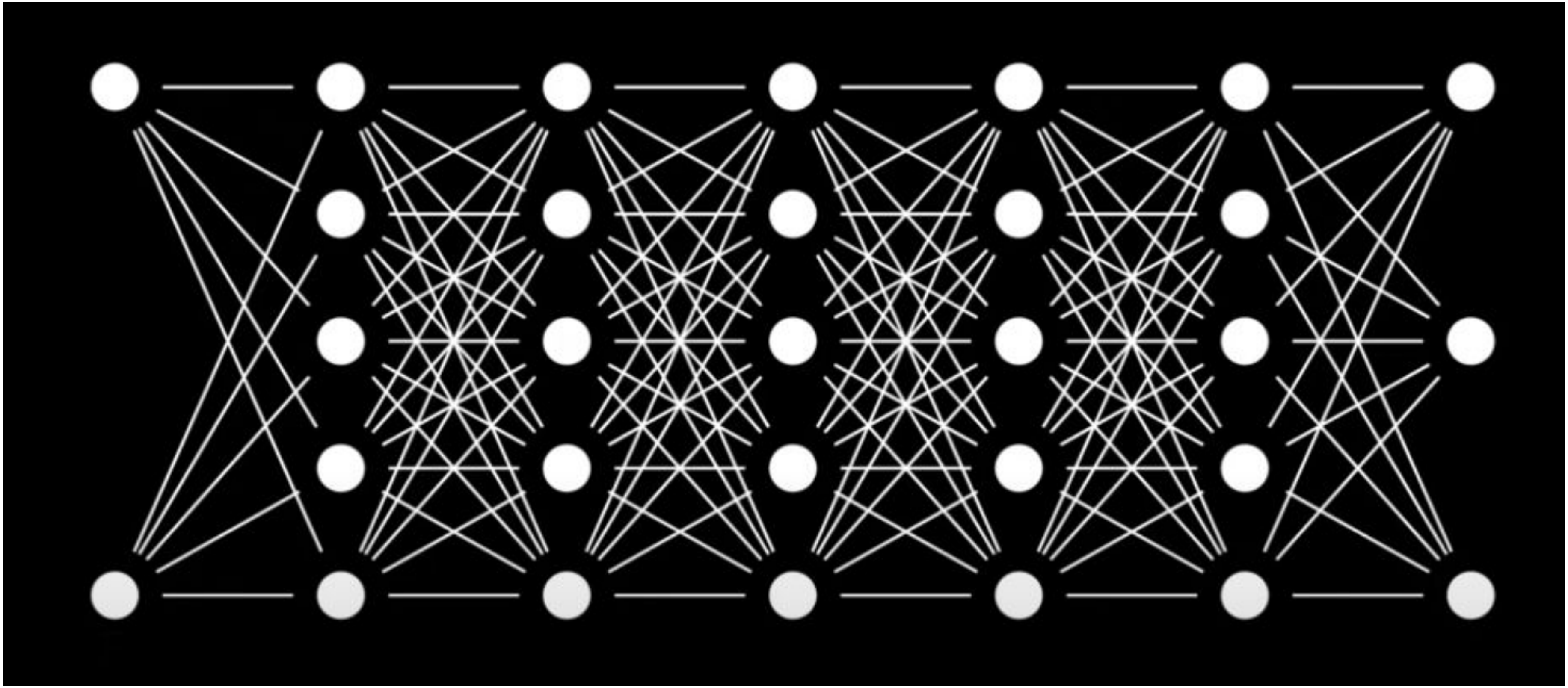
On remplace la fonction linear par un réseau de fonctions connecté les une aux autres



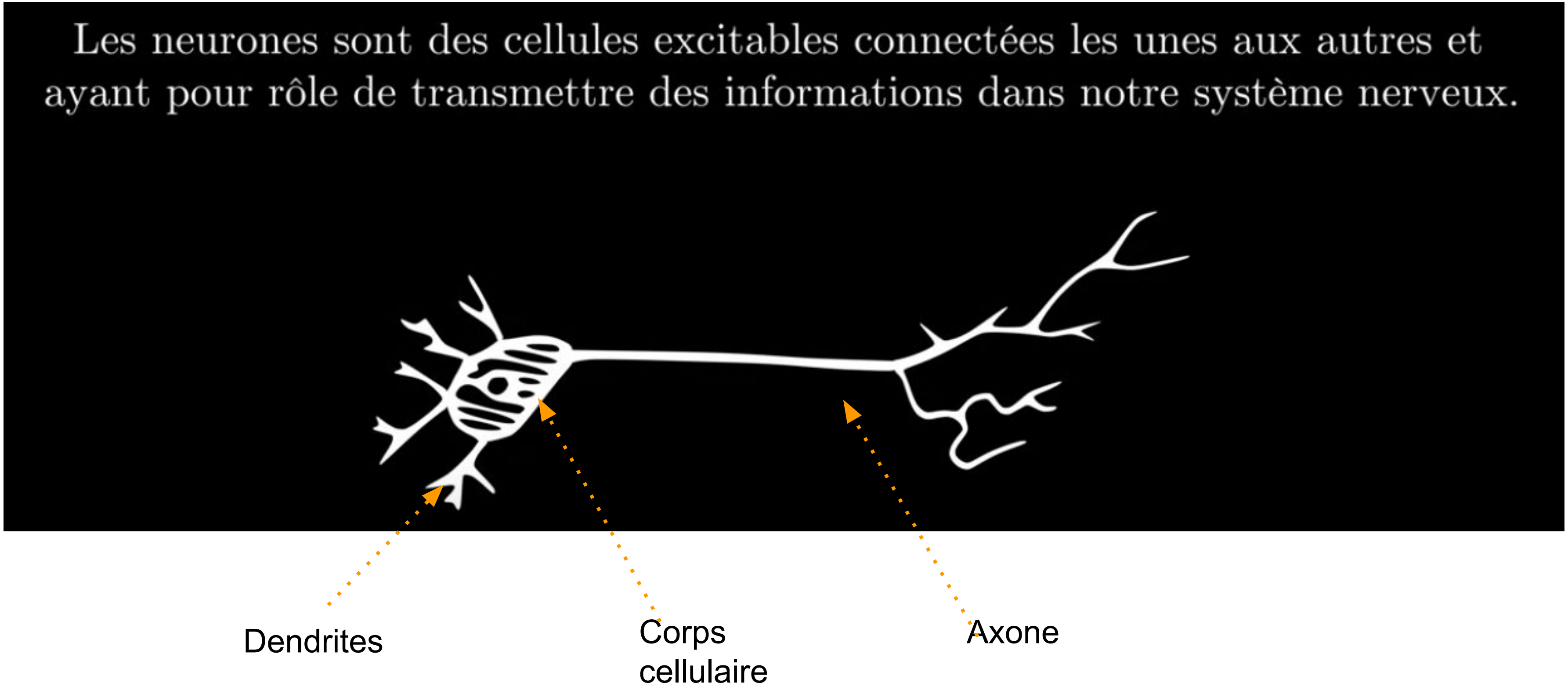
Un réseau de neurones



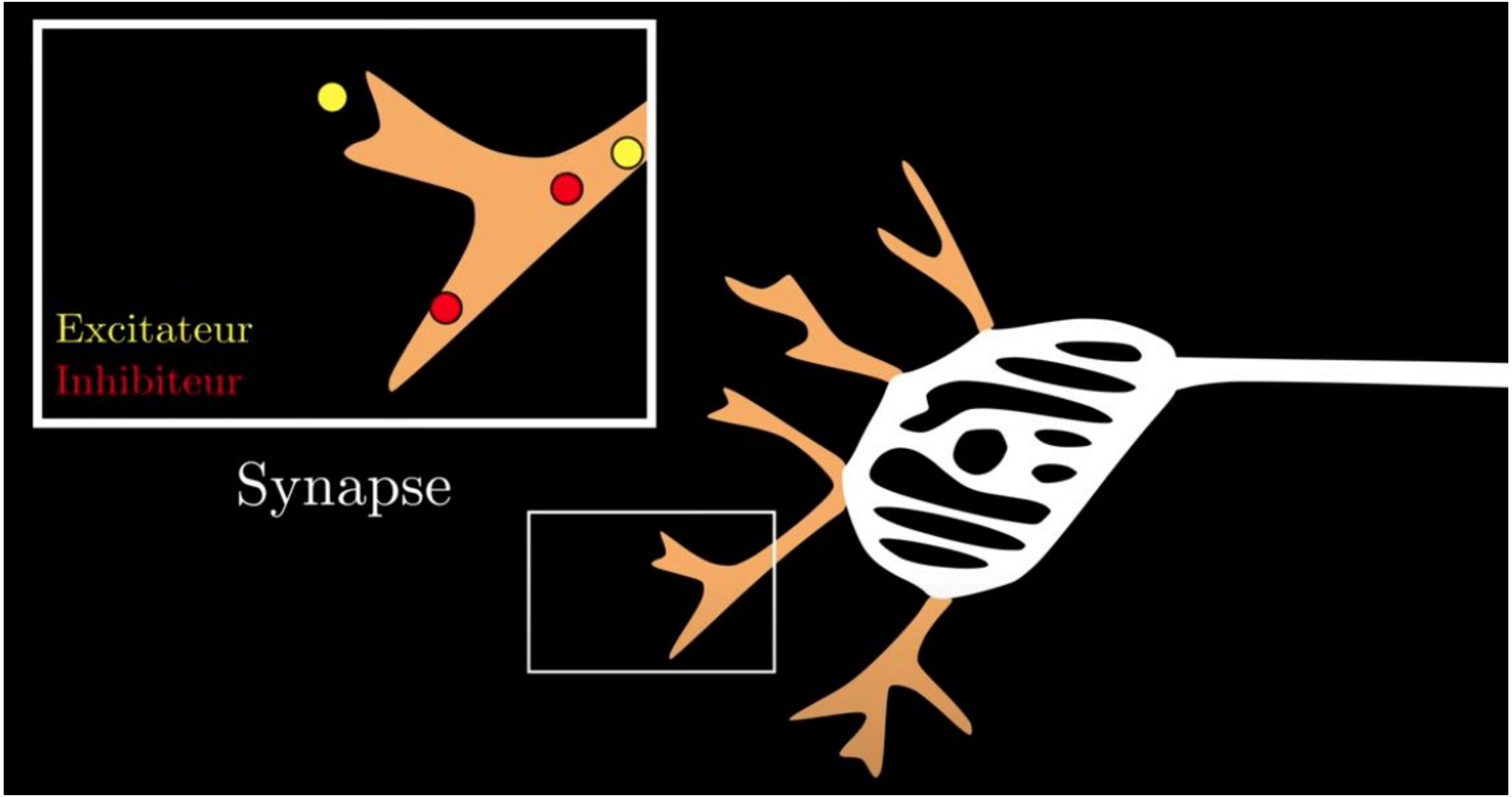
Plus ce réseau est profond plus la machine est capable d'apprendre de tâche complexe → Deep Learning

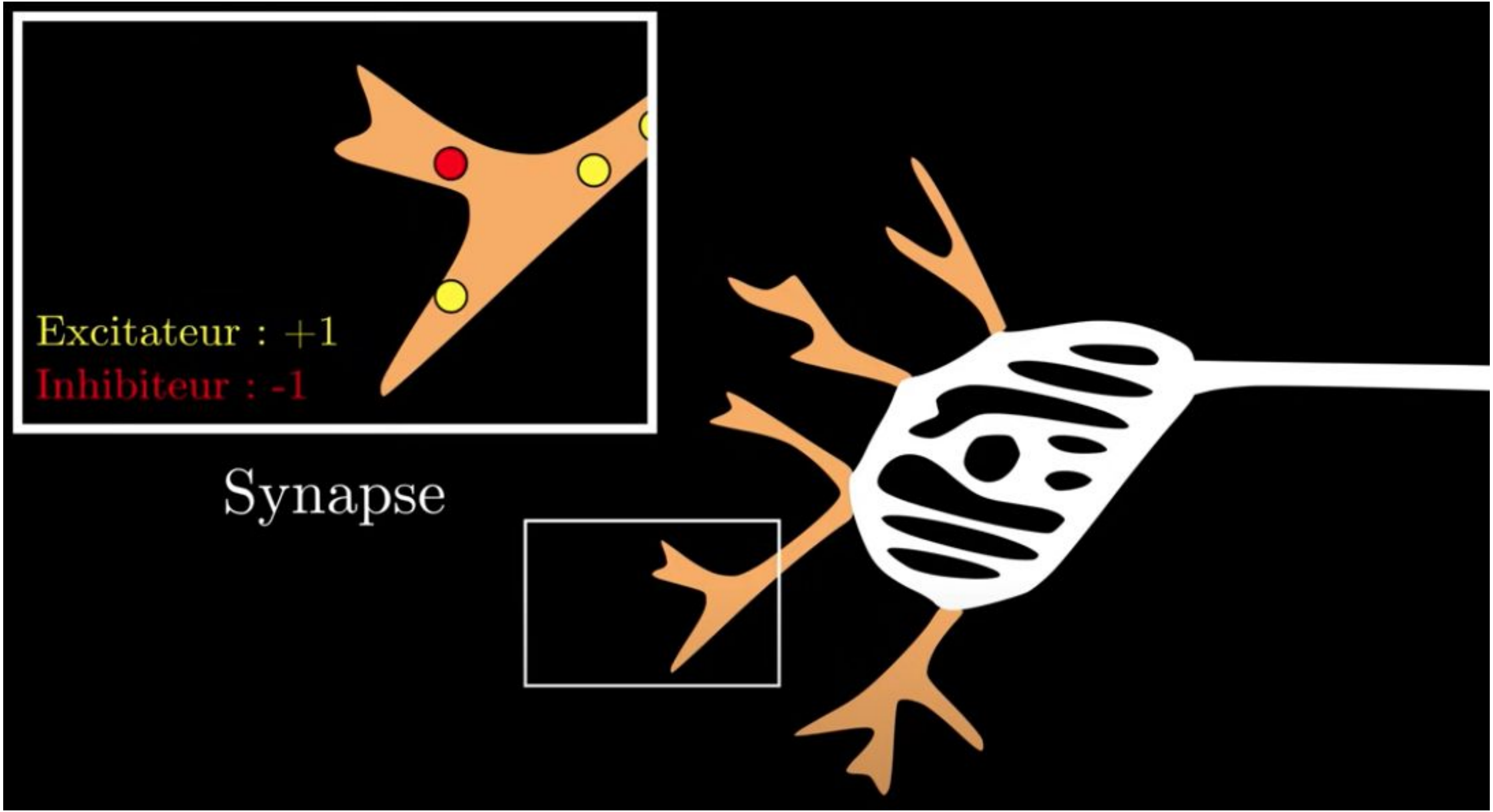


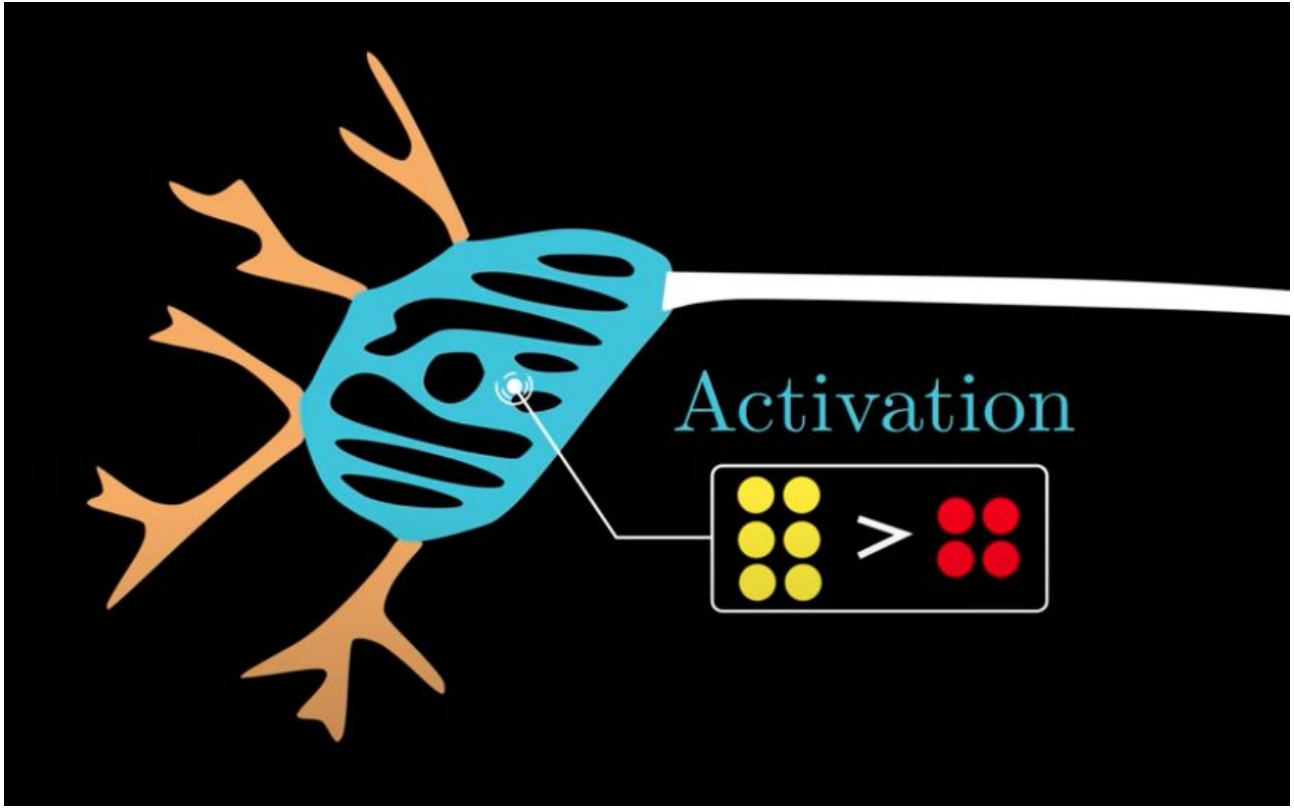
Les neurones



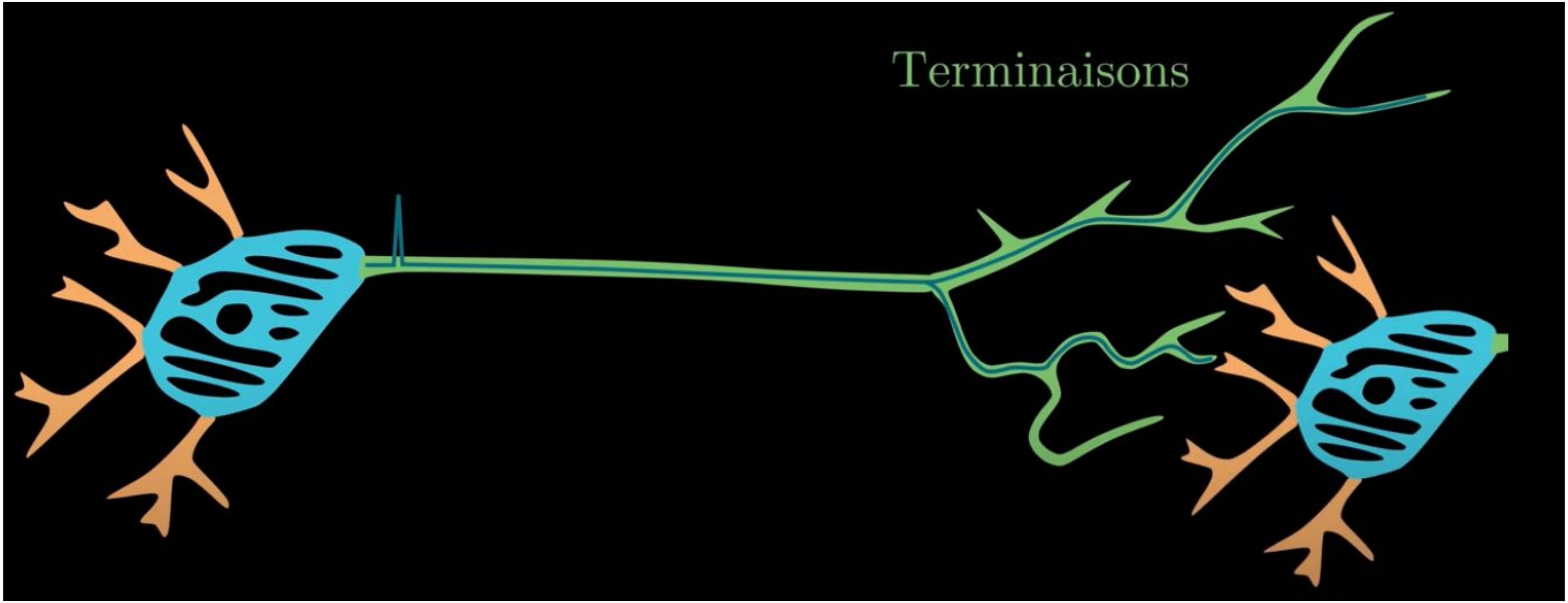
Les neurones




Excitateur : +1
Inhibiteur : -1
Synapse

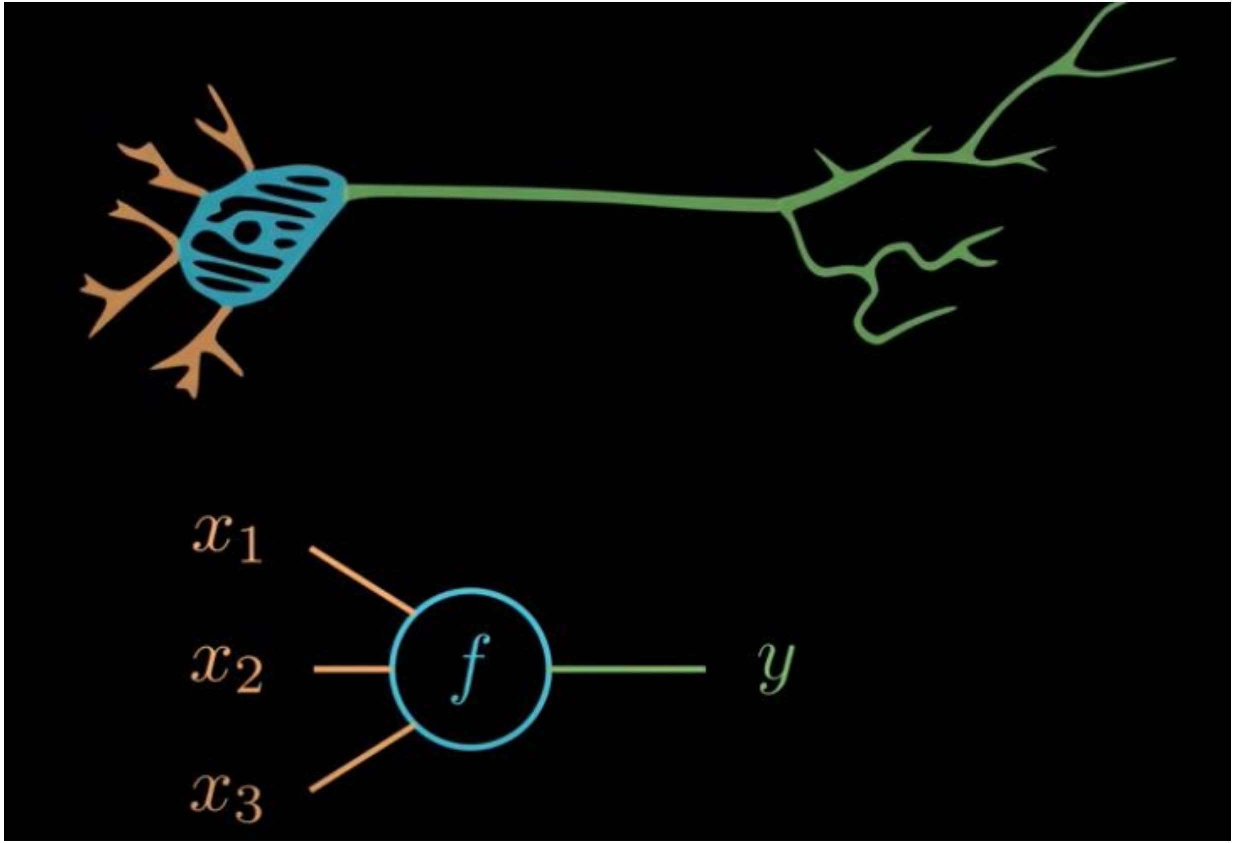


Dès que la somme de ces signaux dépasse un certain seuil le neurone s’active et produire un signal électrique qui est propager à d’autre neurones

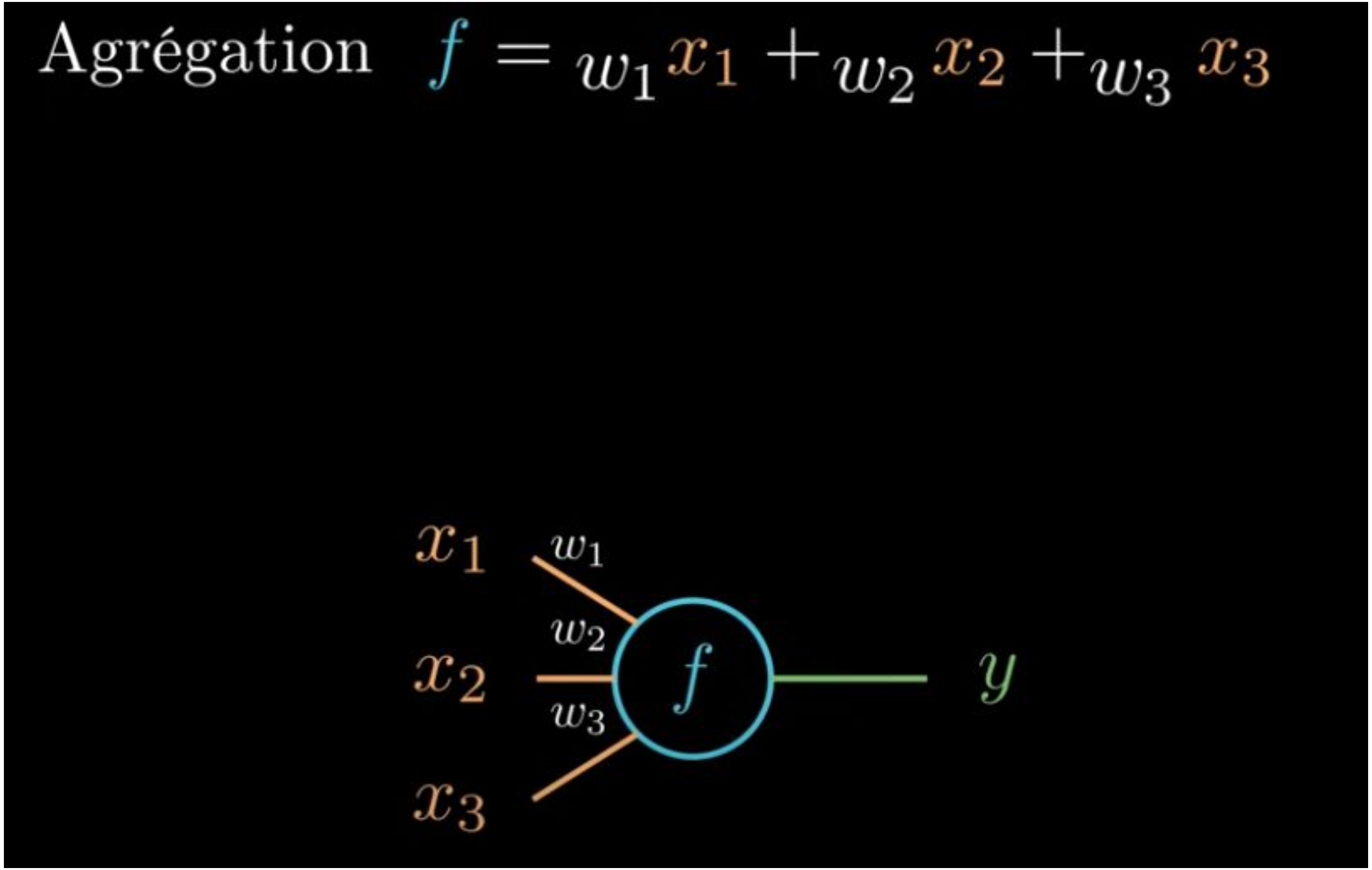


Dès que la somme de ces signaux dépasse un certain seuil le neurone s’active et produire un signal électrique qui est propager à d’autre neurones

Modélisation mathématique



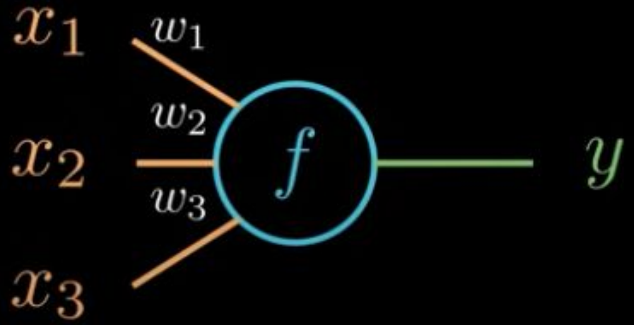
Modélisation mathématique



Modélisation mathématique

Agrégation $f = w_1 x_1 + w_2 x_2 + w_3 x_3$

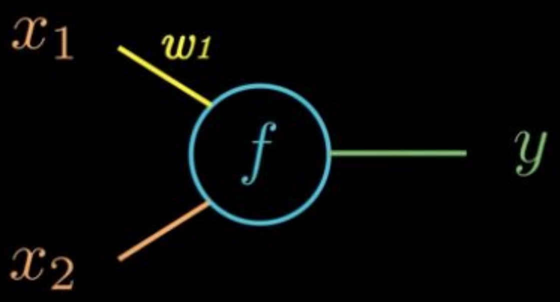
Activation $\begin{cases} y = 1 & \text{si } f \geq 0 \\ \boxed{y = 0} & \boxed{\text{sinon}} \end{cases}$



L'apprentissage

Le Perceptron (1957) :

Entraîner un neurone artificiel sur des données de références (X, y) pour que celui-ci renforce ses paramètres W à chaque fois qu'une entrée X est activée en même temps que la sortie y présente dans ces données.



$$\boxed{W = W + \alpha(y_{true} - y)X}$$

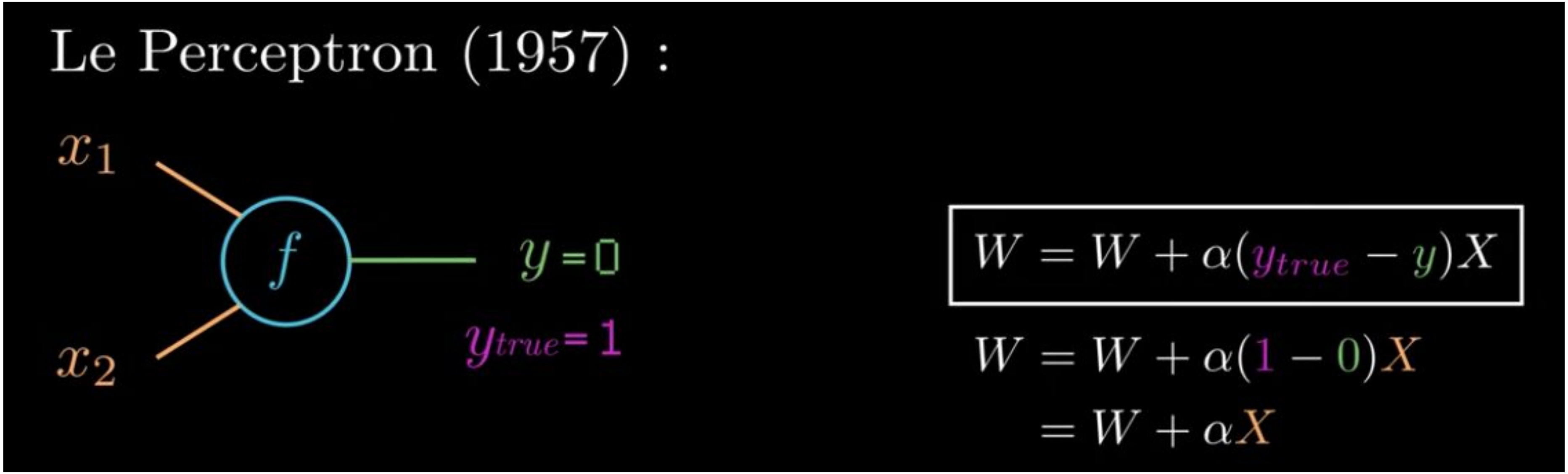
y_{true} : sortie de référence

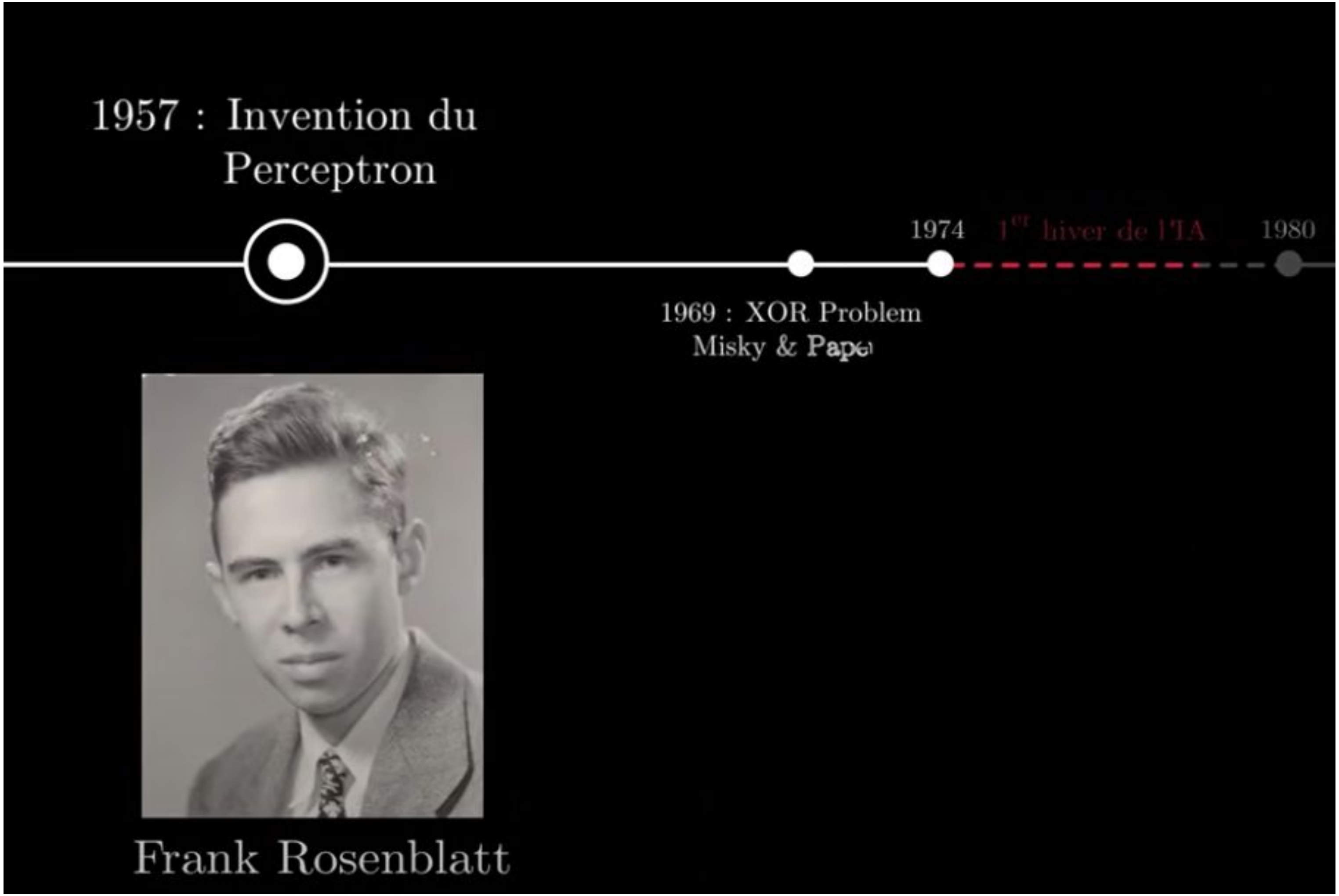
y : sortie produite par le neurone

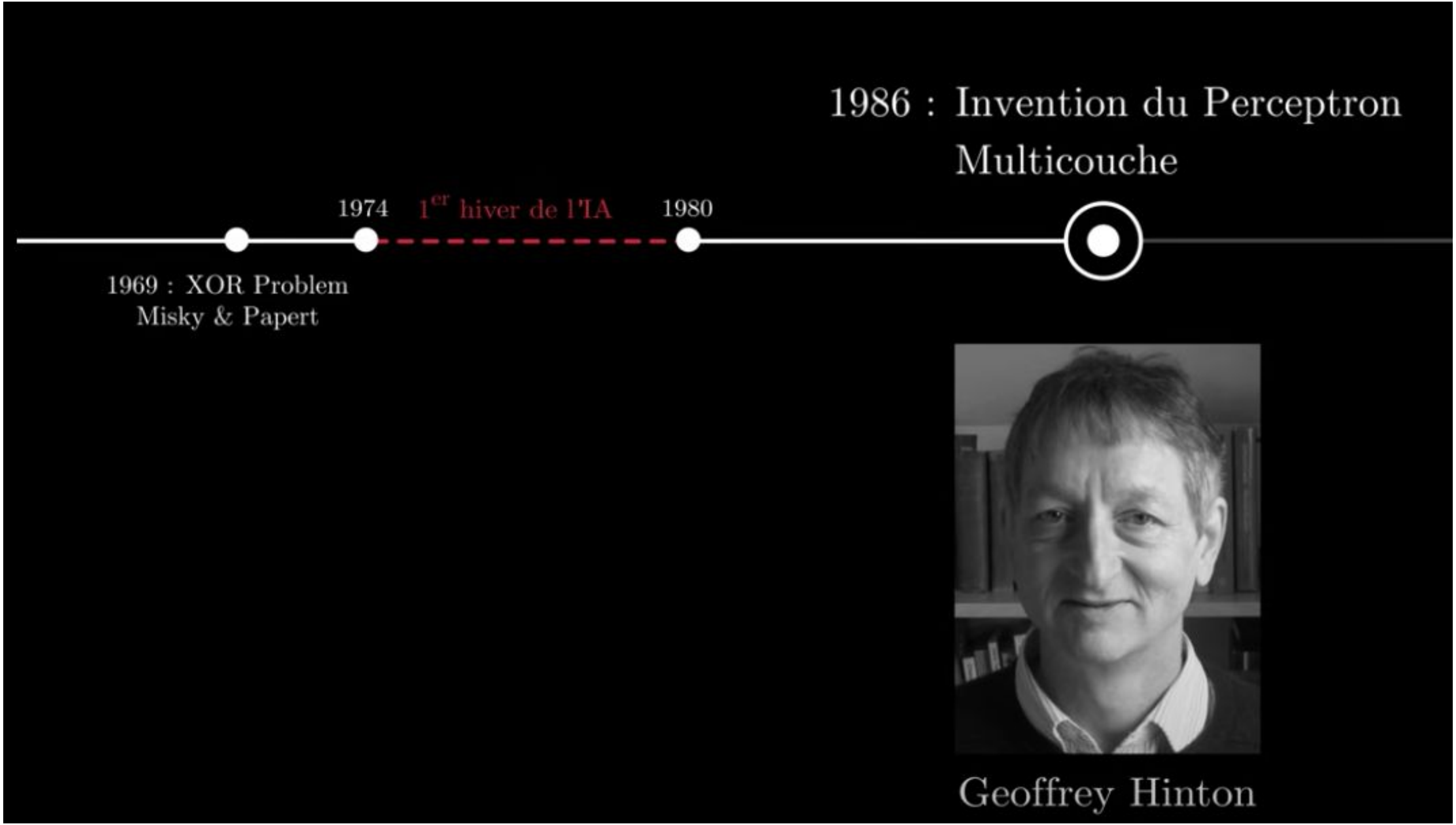
X : entrée du neurone

α : vitesse d'apprentissage

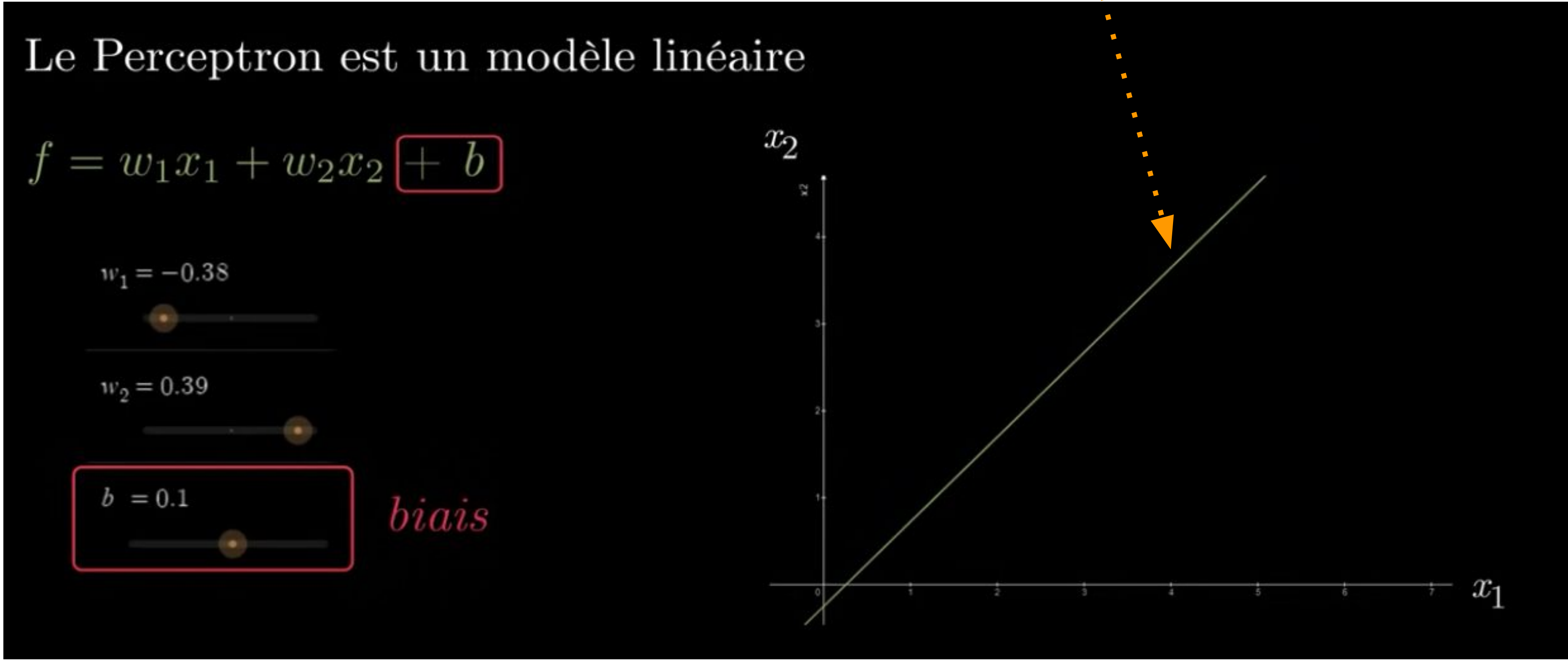
L'apprentissage



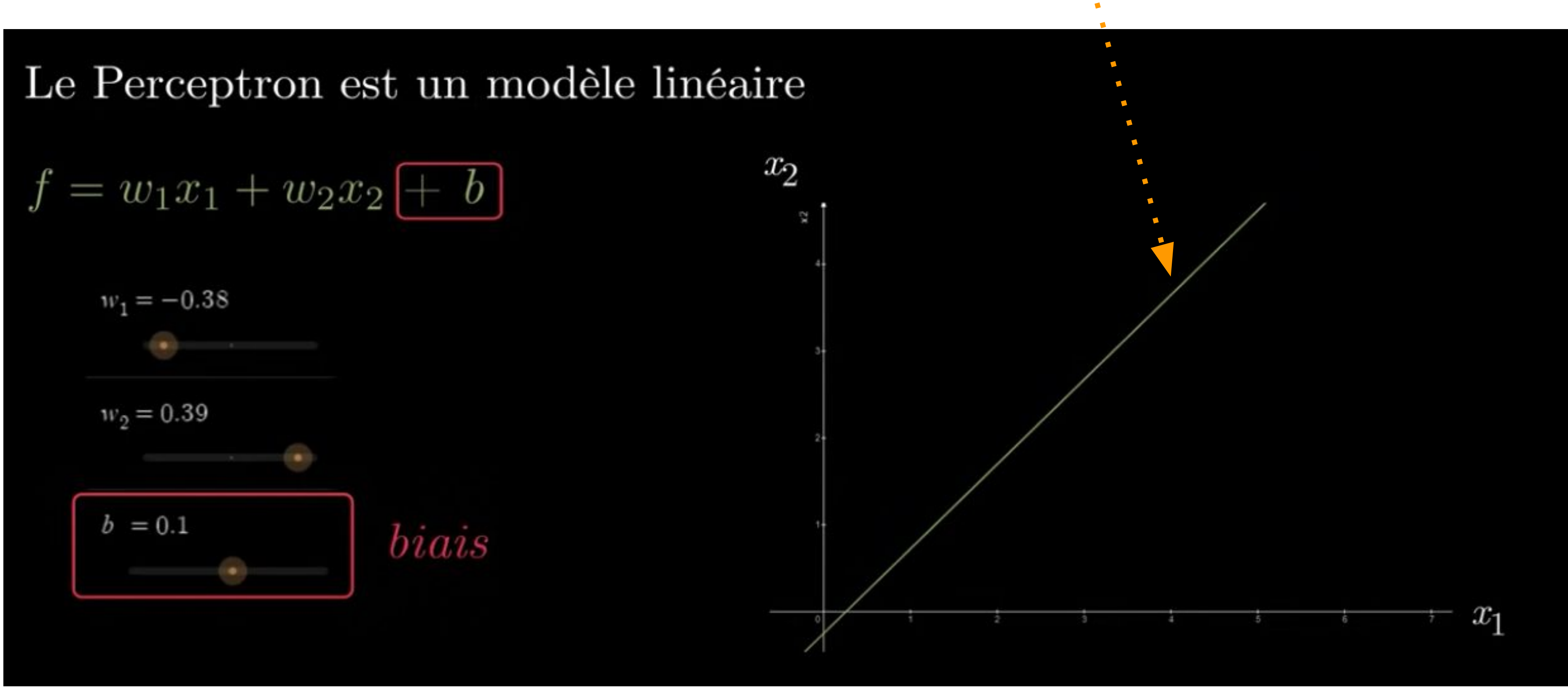

1957 : Invention du Perceptron
1974
1er hiver de l'IA
1980
1969 : XOR Problem
Misky & Pape
Frank Rosenblatt


1986 : Invention du Perceptron
Multicouche
1974
1^er hiver de l'IA
1980
1969 : XOR Problem
Misky & Papert
Geoffrey Hinton

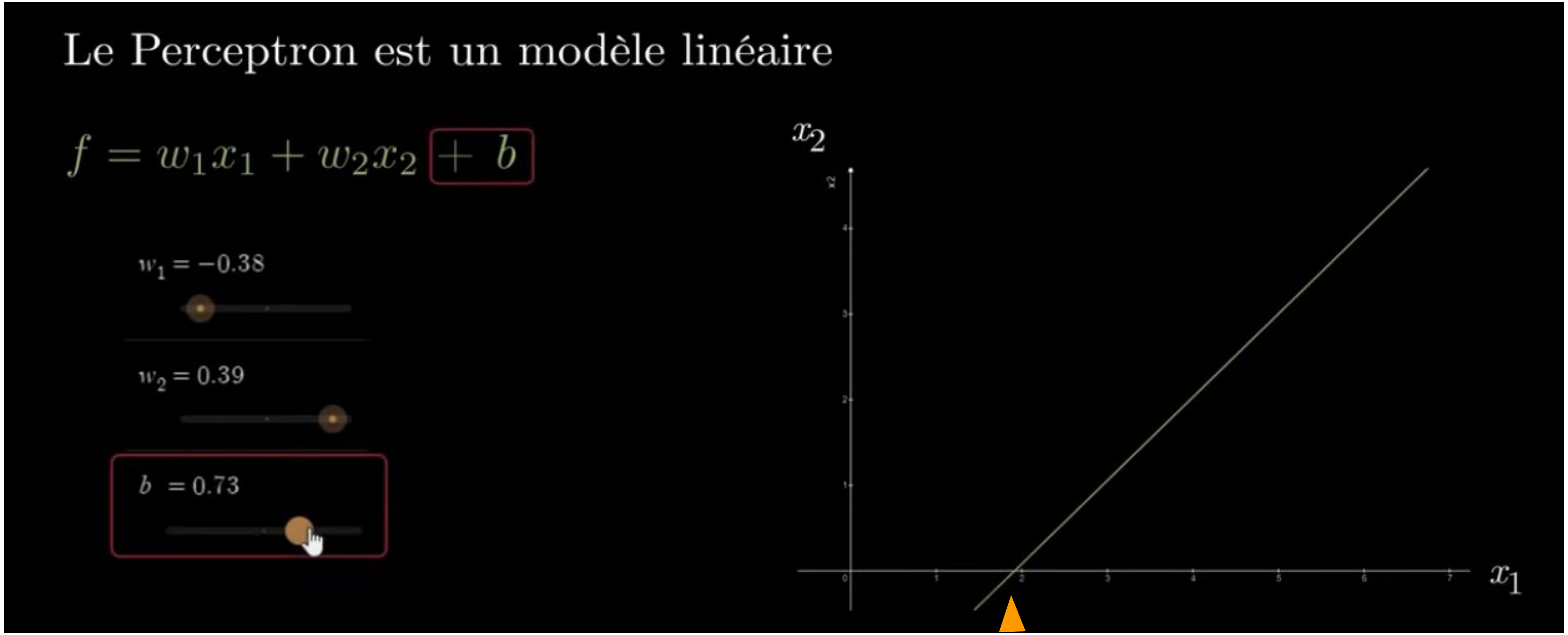
L'inclinaison de la droite est régulée par les paramètres w



L'inclinaison de la droite est régulée par les paramètres w

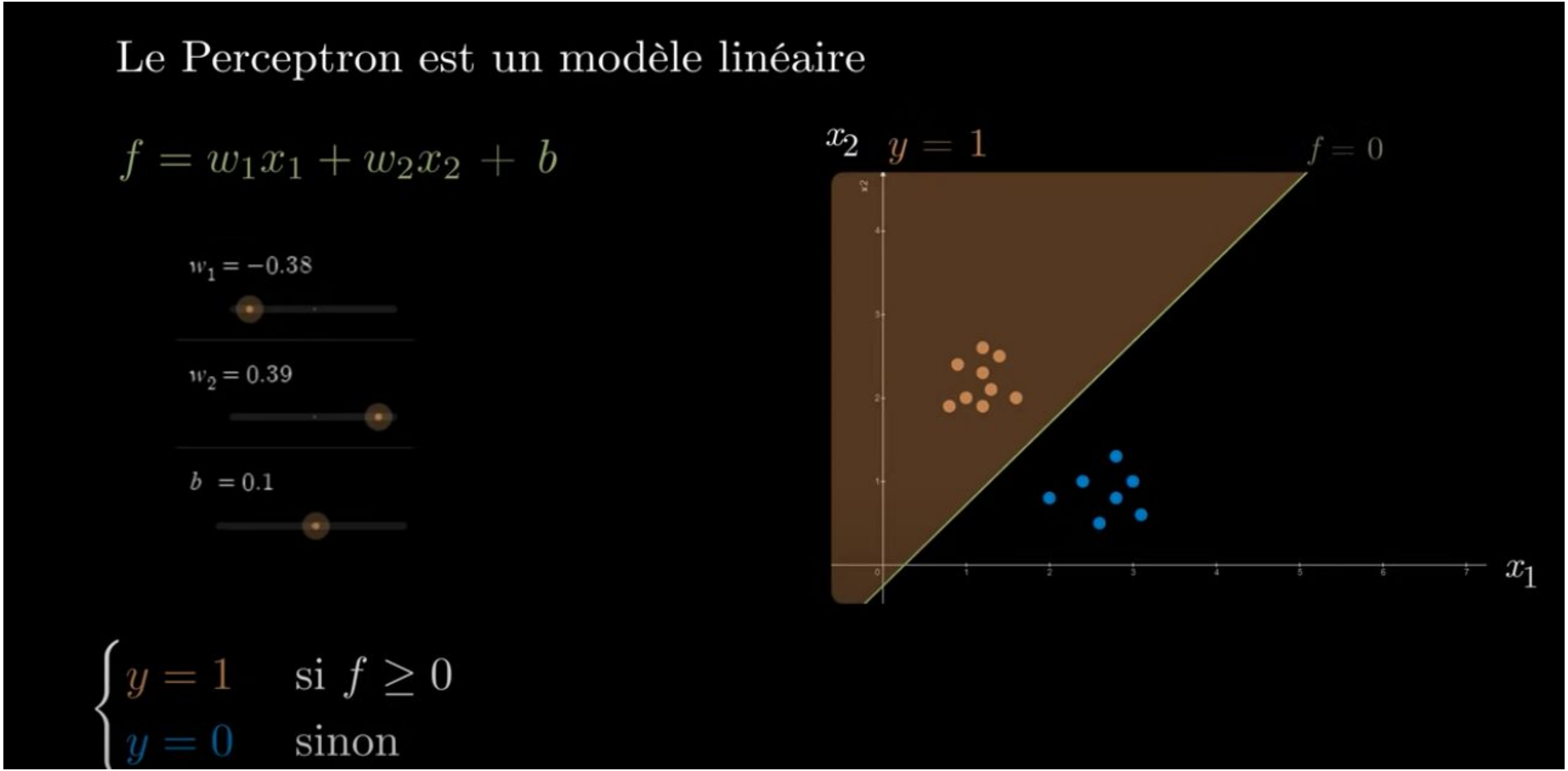


L'inclinaison de la droite est régulée par les paramètres w

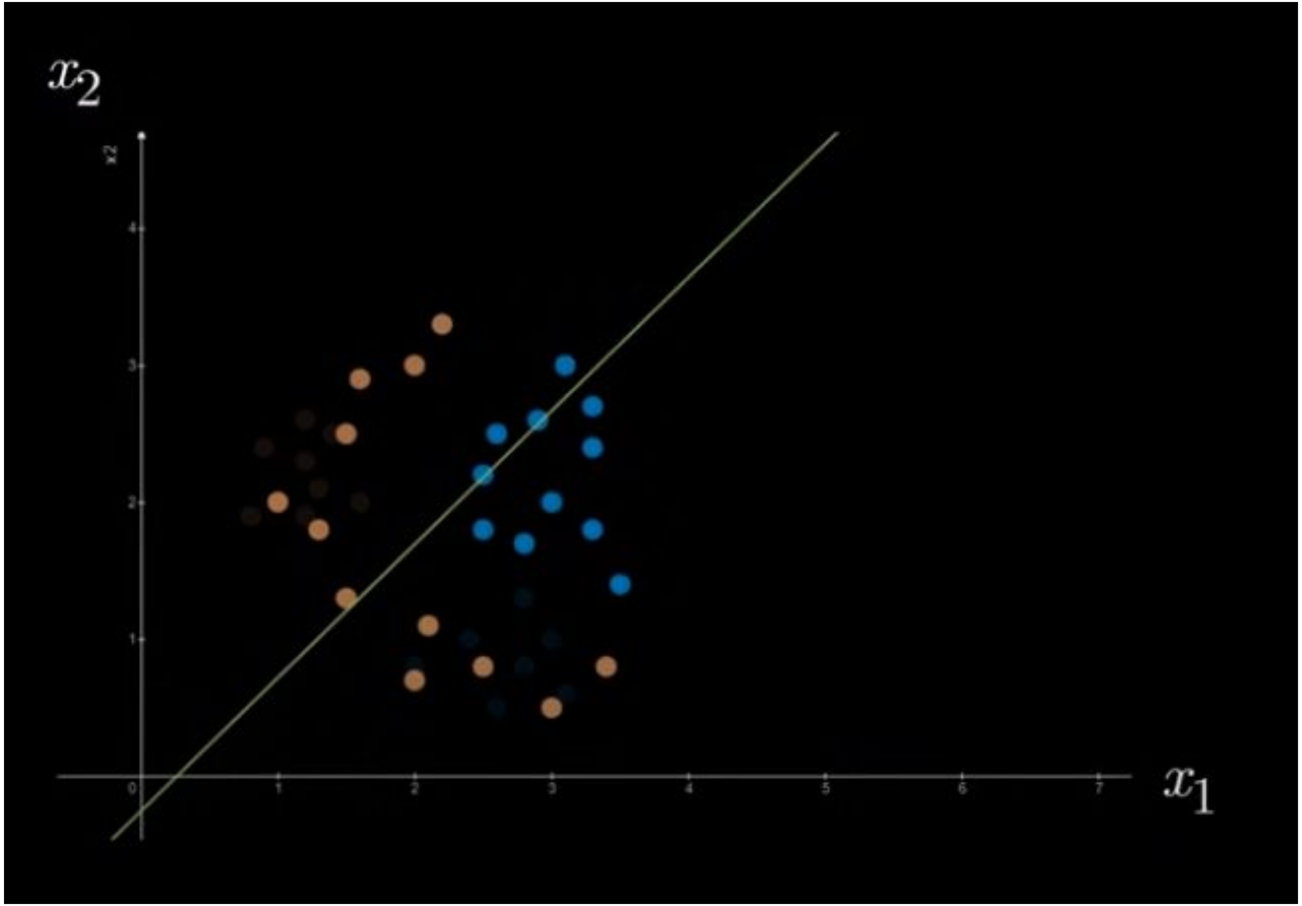


La position par le paramètre biais

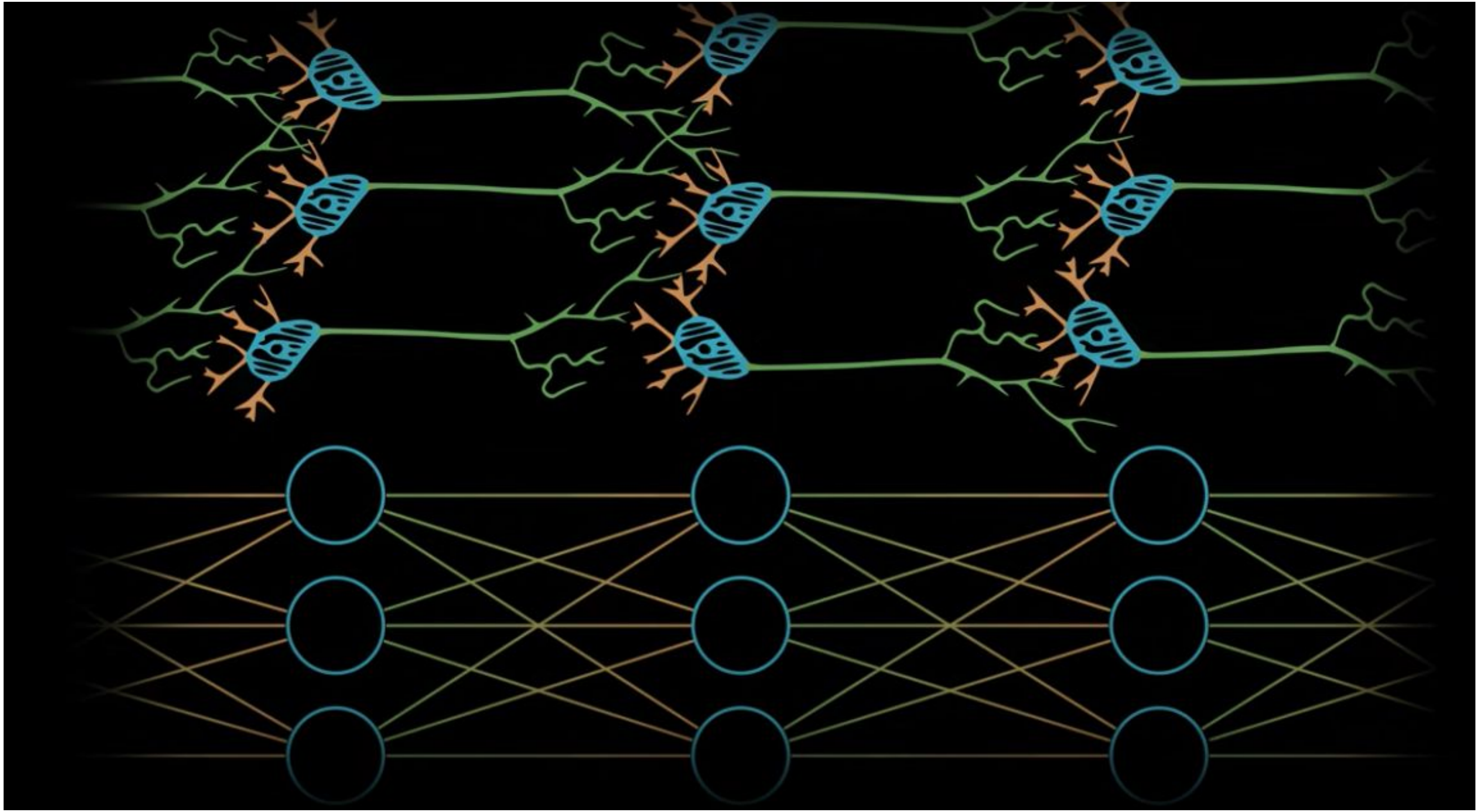
Nous pouvons séparer deux classe de points



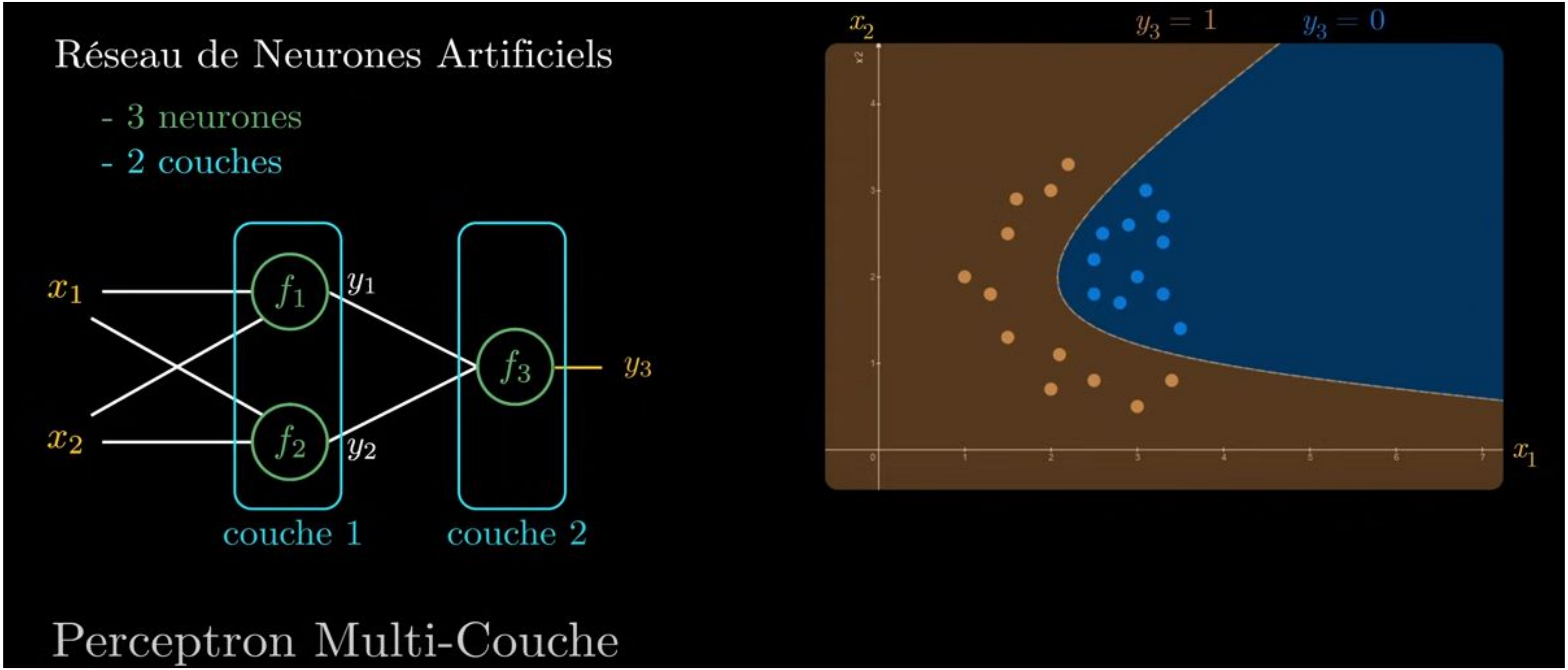
Le problème est qu'une grande partie de problèmes réels ne sont pas linéairement séparables.



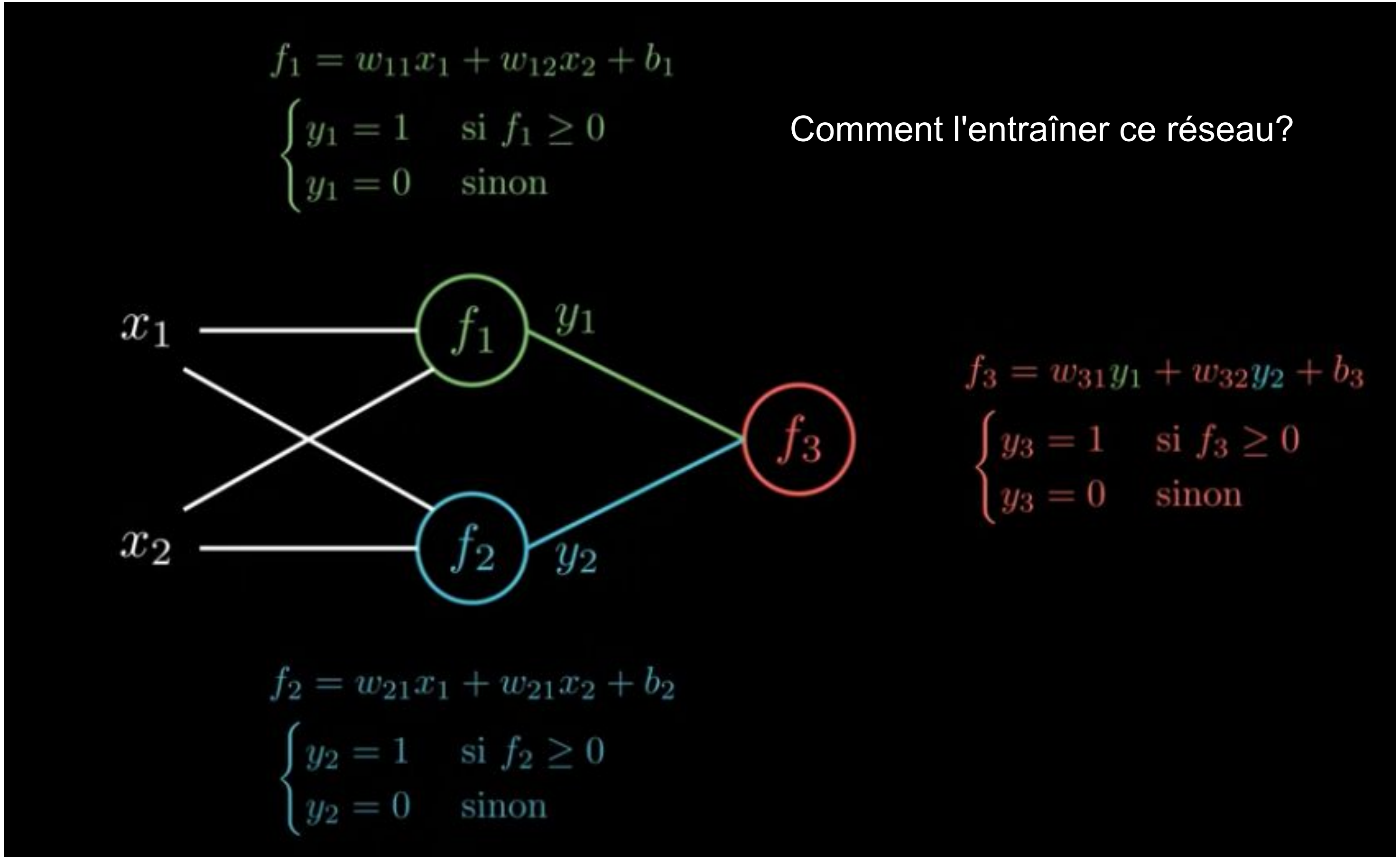
Nous pouvons accumuler des couches de neurones pour résoudre des tâches plus complexes.



Nous pouvons accumuler des couches de neurones pour résoudre des tâches plus complexes.

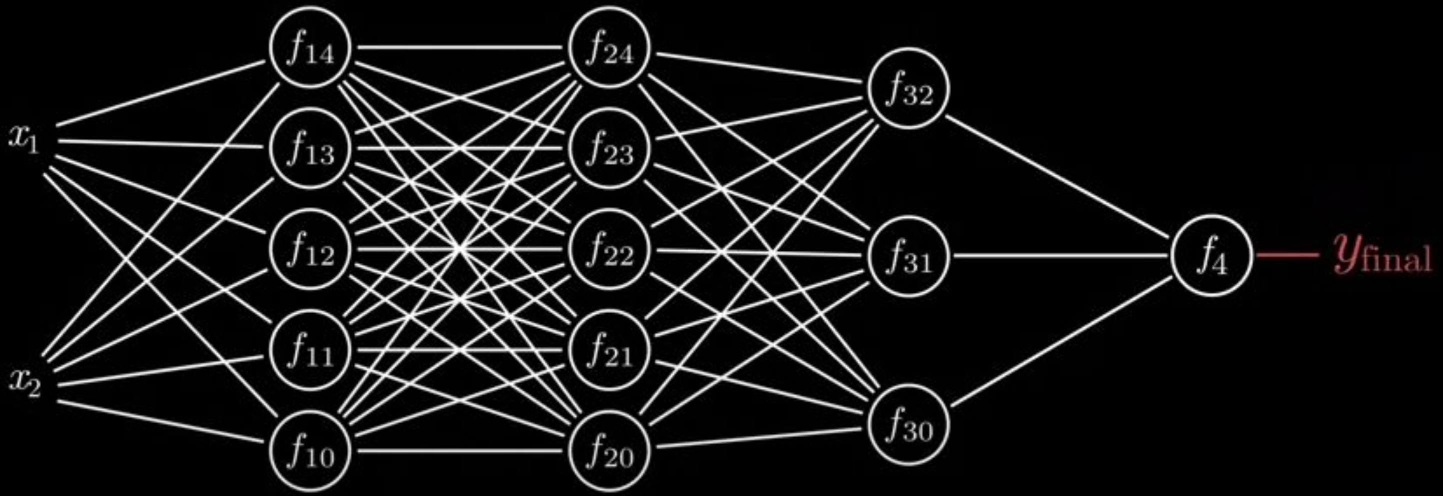


Nous pouvons accumuler des couches de neurones pour résoudre des tâches plus complexes.

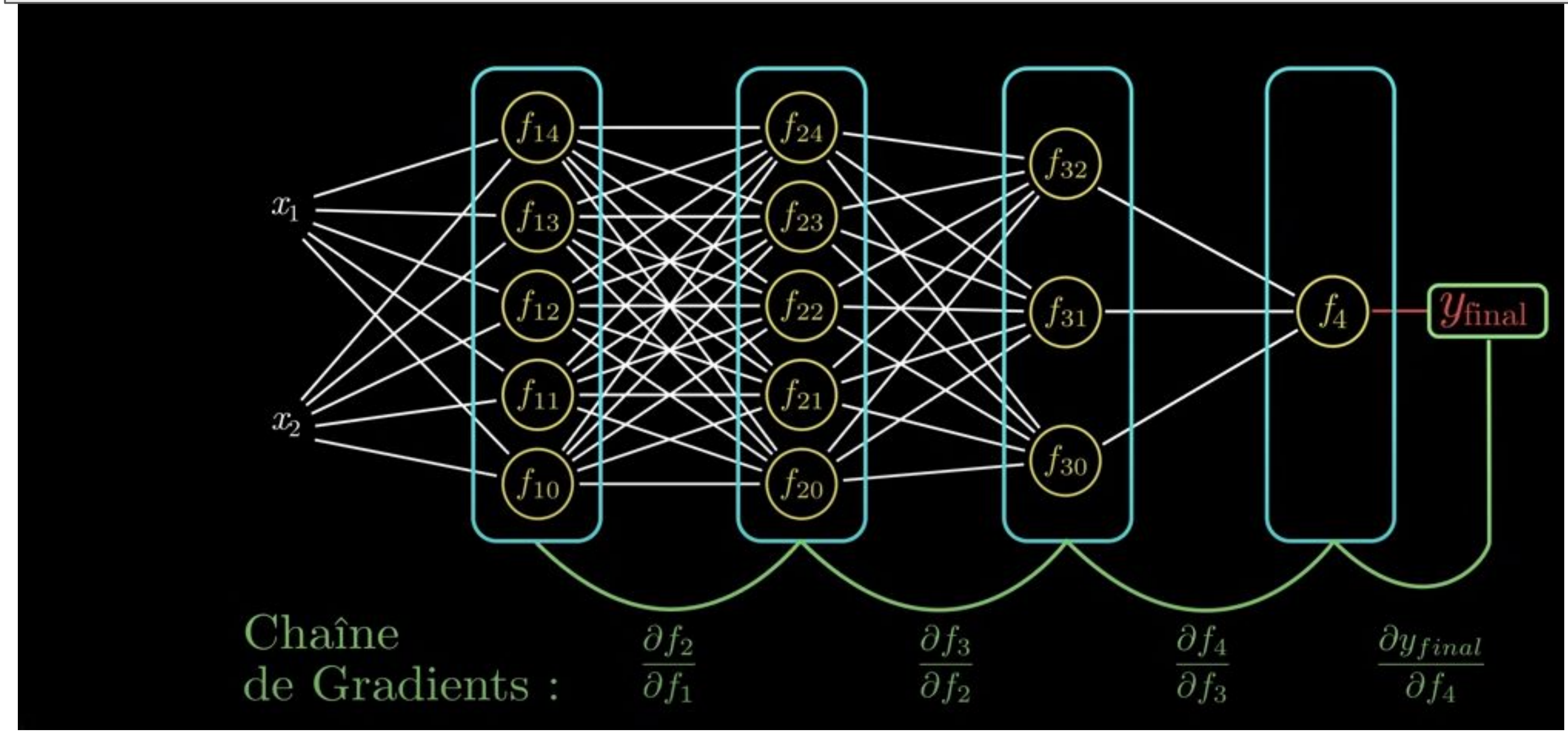


Back-Propagation

Consiste à déterminer comment la sortie du réseau varie en fonction des paramètres (W, b) présents dans chaque couche.

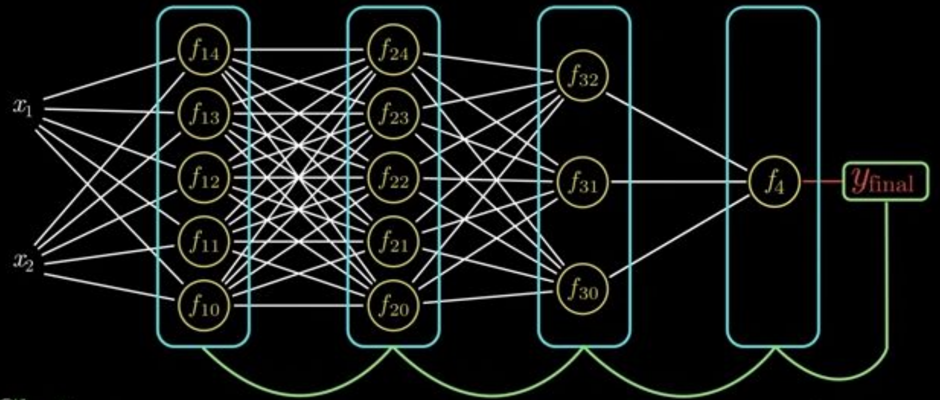


On calcule une chaîne de gradient, indiquant comment la sortie varie en fonction de la dernière couche, puis comment la dernière couche varie en fonction de l’avant dernier, etc.

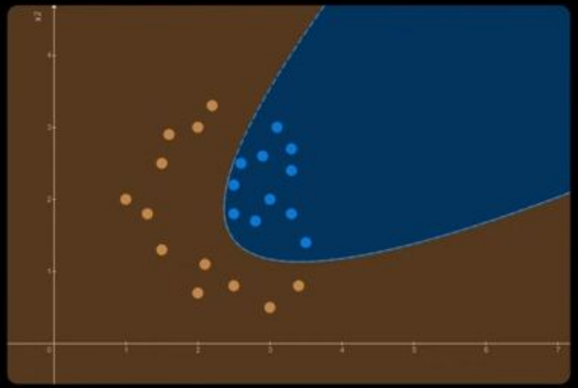


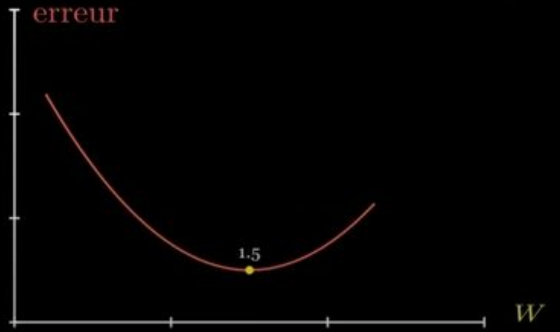
Back-Propagation

Grâce aux gradients, on peut alors mettre à jour les paramètres (W, b) de chaque couche de telle sorte à ce qu'ils minimisent l'erreur entre la sortie du modèle et la réponse attendue.



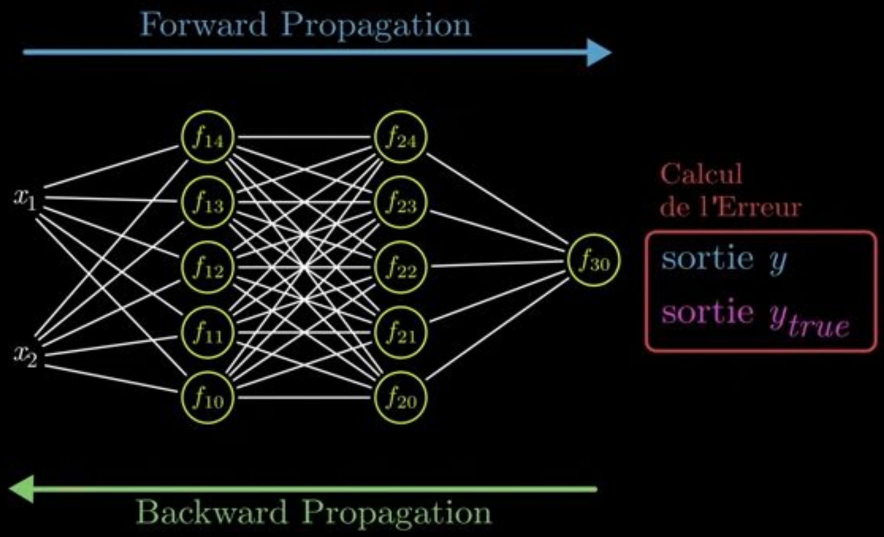
Chaîne de Gradients : $\frac{\partial f_2}{\partial f_1}$ $\frac{\partial f_3}{\partial f_2}$ $\frac{\partial f_4}{\partial f_3}$ $\frac{\partial y_{final}}{\partial f_4}$





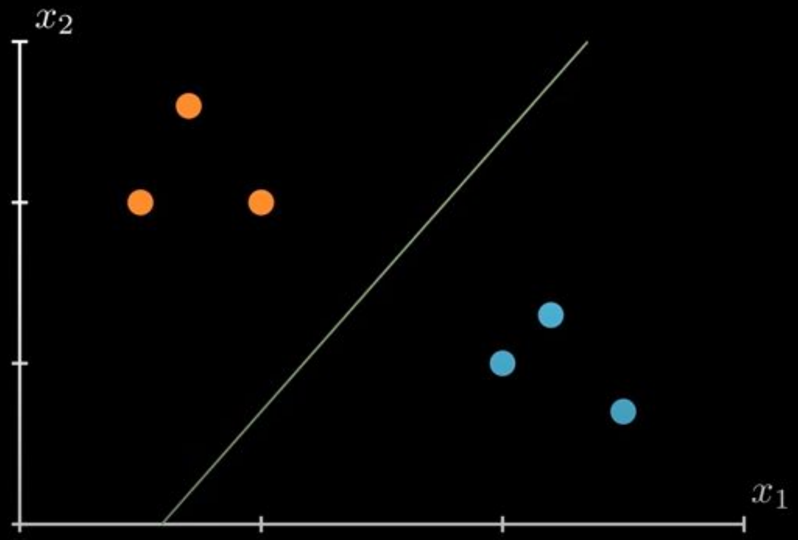
En Résumé :
Développer des Réseaux de Neurones Artificiels

1. Forward Propagation
2. Cost Function
3. Backward Propagation
4. Gradient Descent

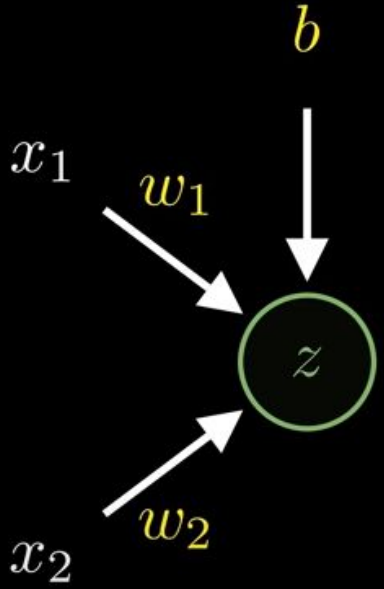


x_1 : largeur de la feuille
x_2 : longueur de la feuille

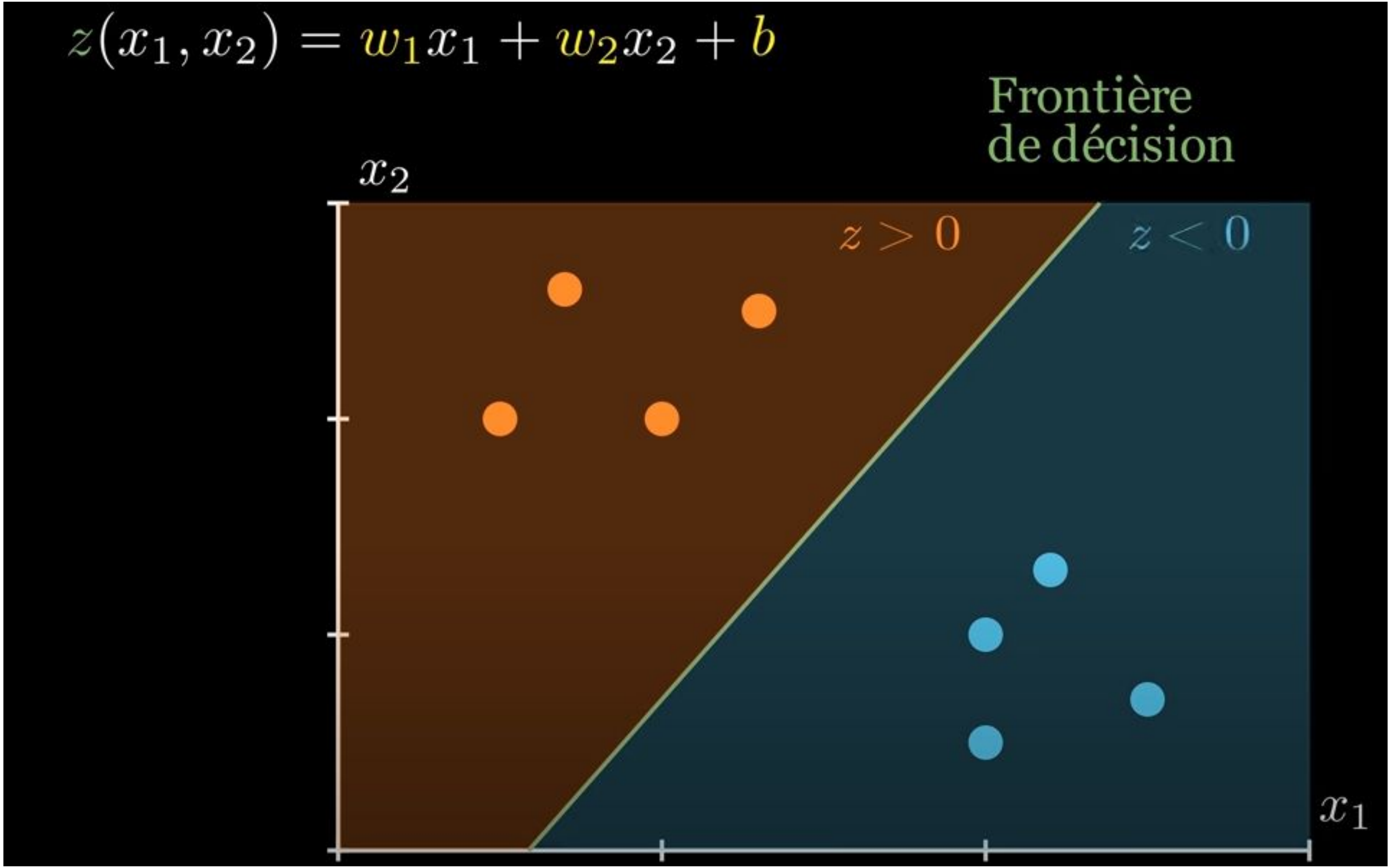
y	x_1	x_2
1	0.5	2.0
1	1.1	2.1
1	0.7	2.6
0	2.0	1.0
0	2.5	0.7
0	2.2	0.3



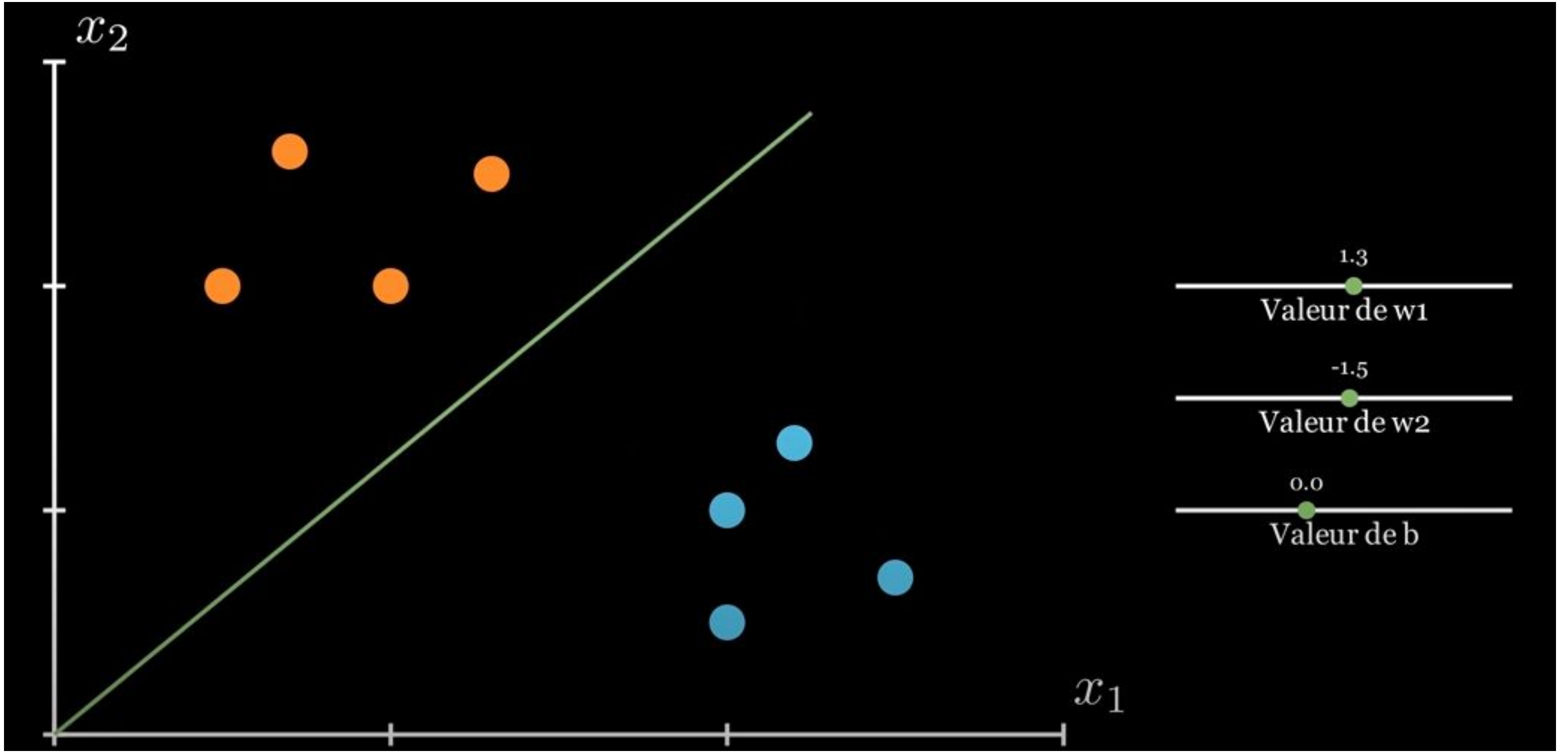
Modèle Linéaire

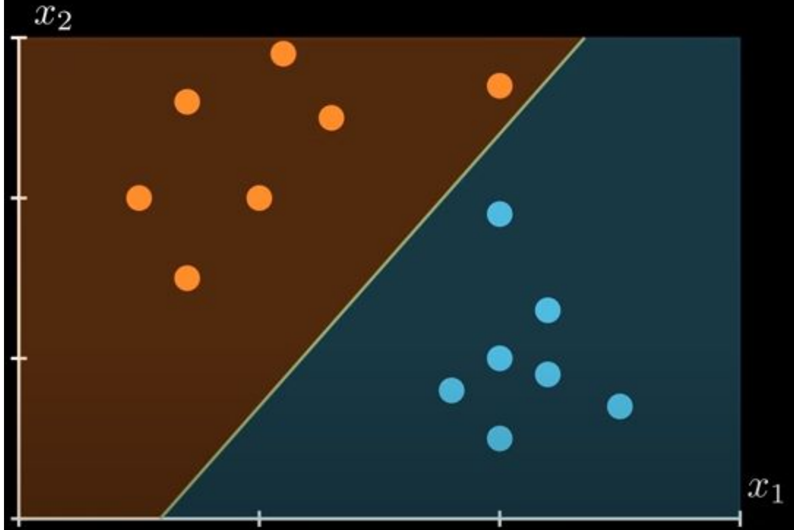


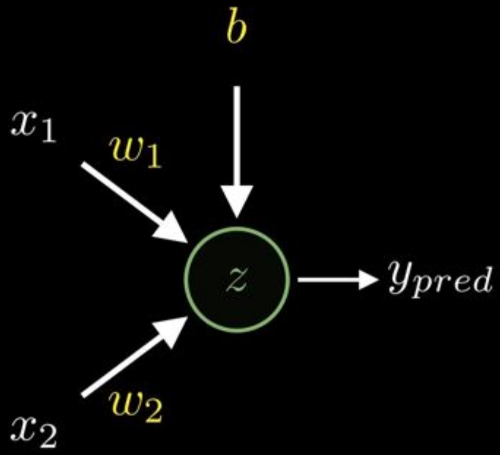
$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$


$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$
Frontière
de décision
x_2
$z > 0$
$z < 0$
x_1

Pour prédire si une nouvelle plante est toxique ou pas, il va falloir régler les paramètres W et b pour trouver la meilleure frontière de décision.



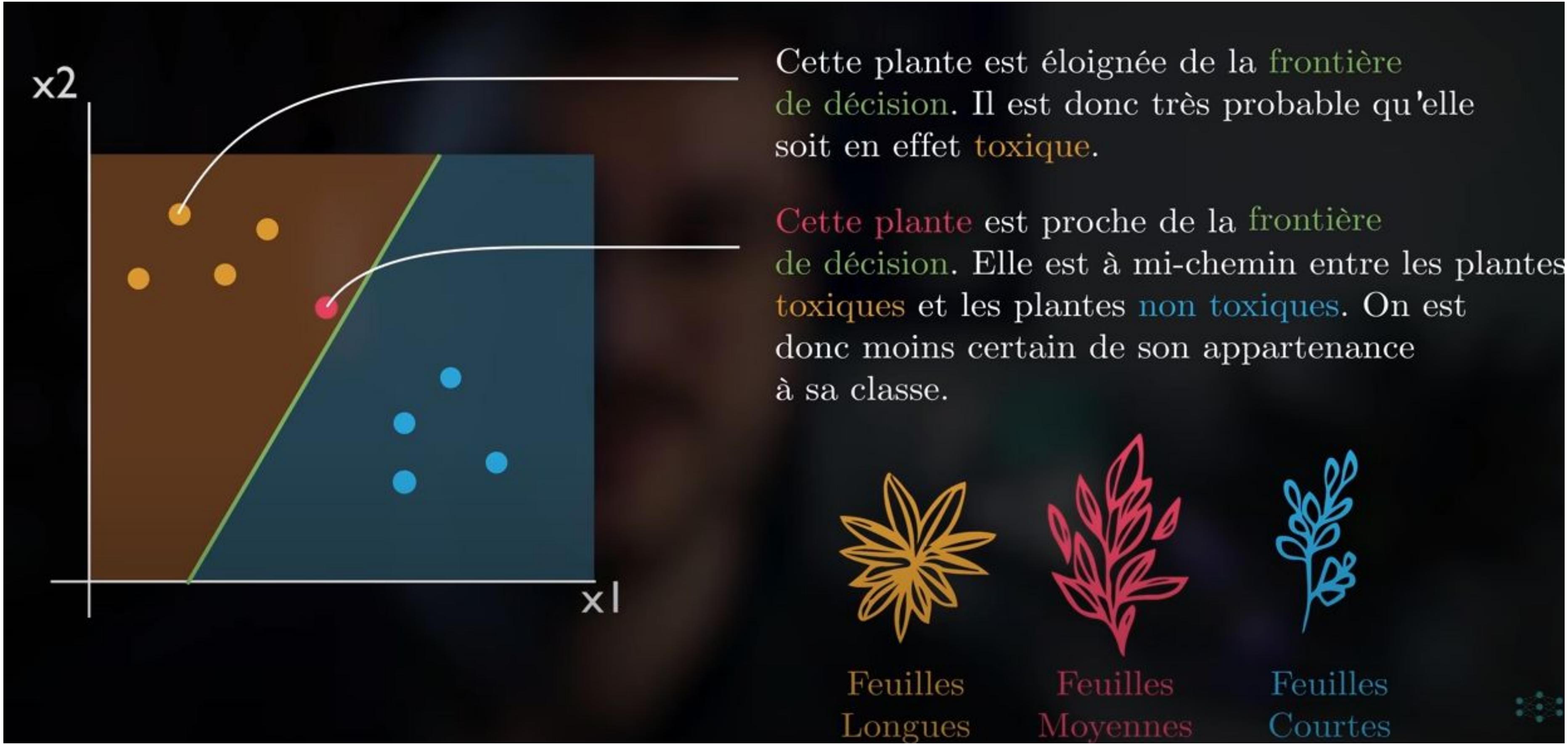




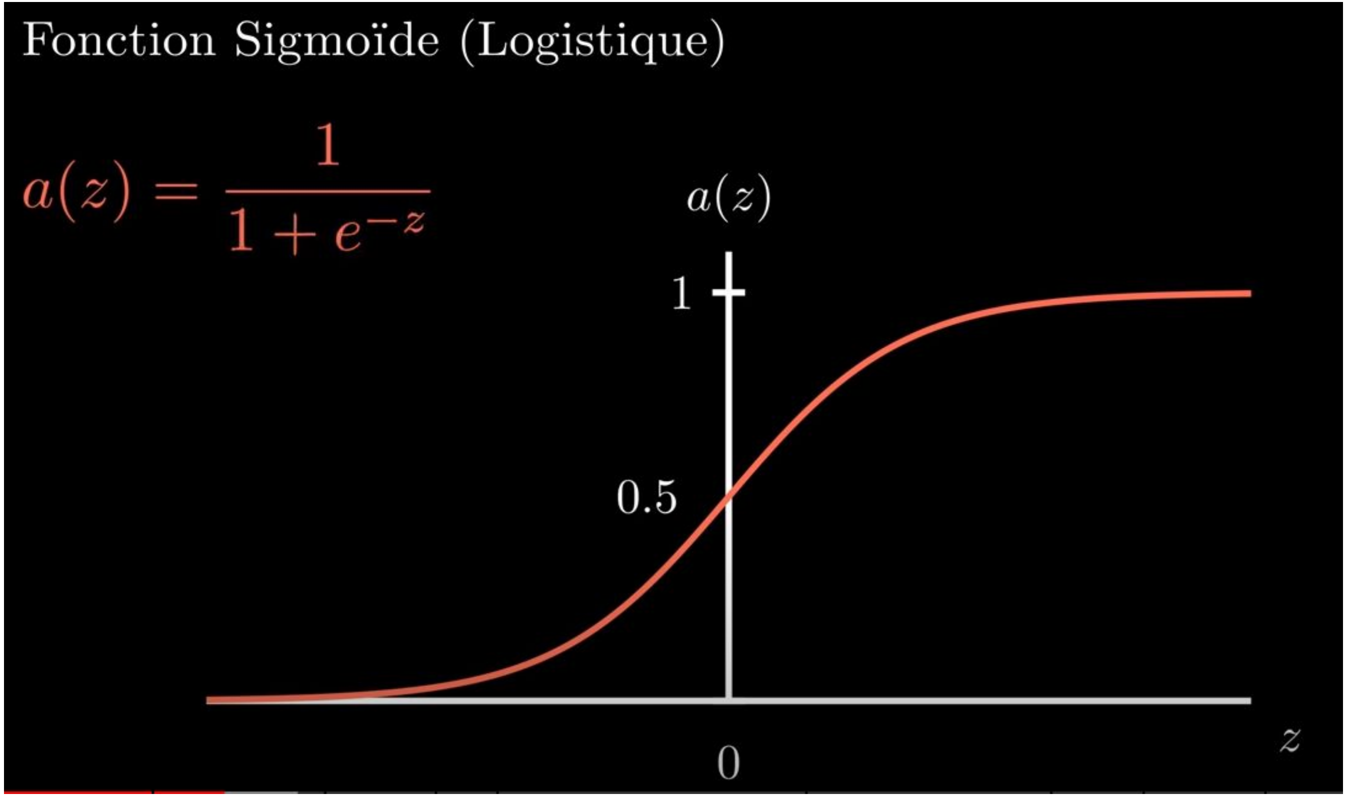
$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

$$\begin{cases} y_{pred} = 0 & si \quad z < 0 \\ y_{pred} = 1 & si \quad z \geqslant 0 \end{cases}$$

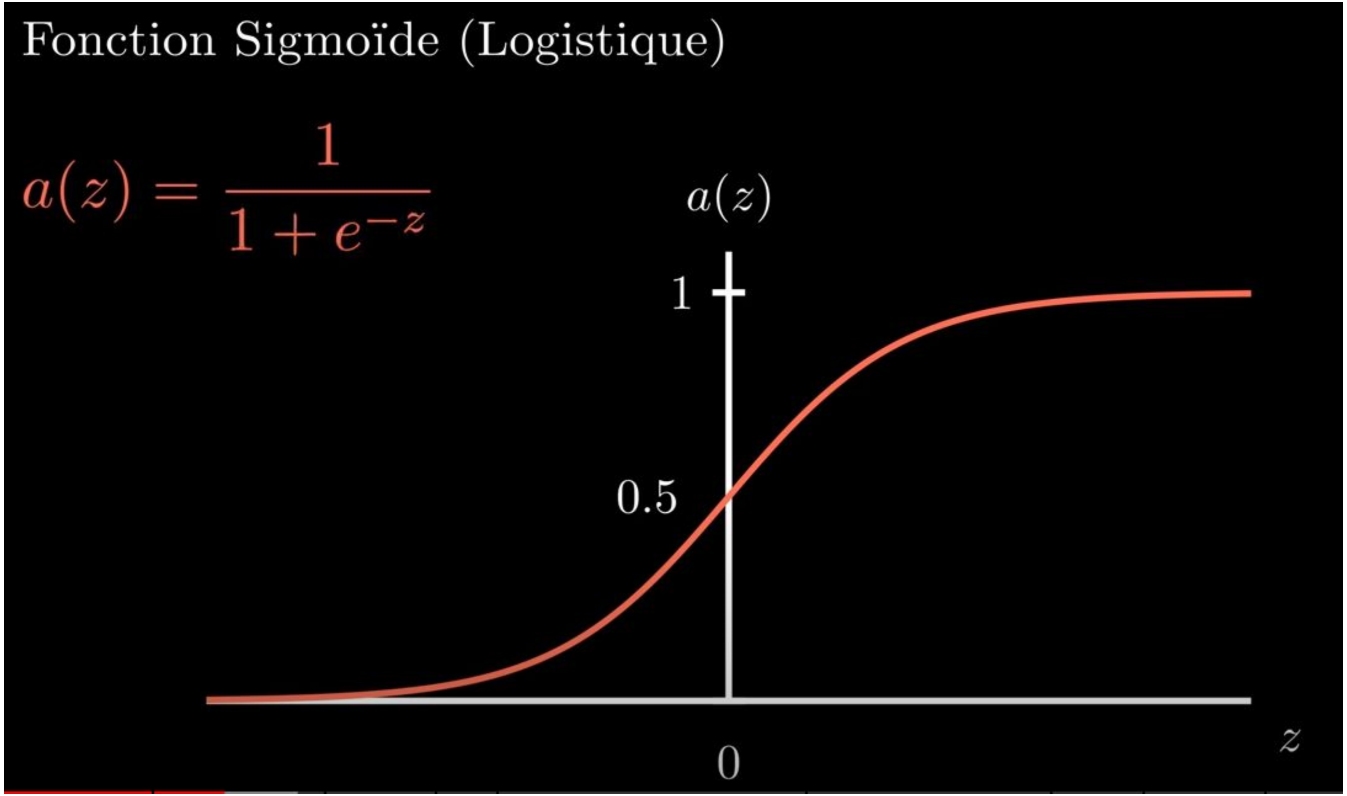
Probabilité d’une prédiction



Probabilité d’une prédiction

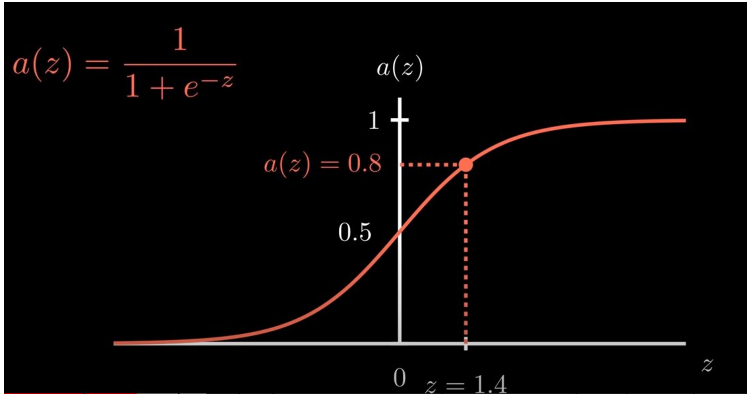


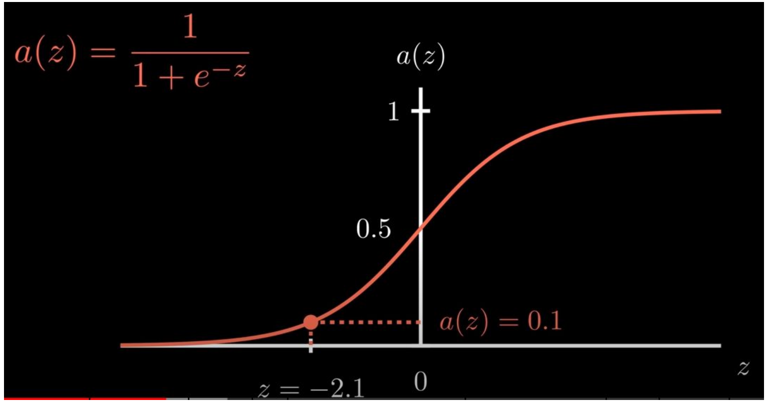
Probabilité d’une prédiction

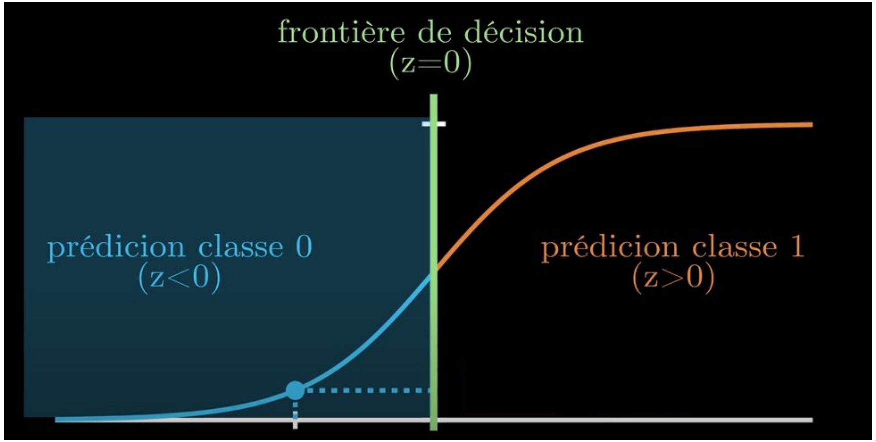


Cette fonction nous permets de convertir la sortie z à une probabilité a(z), qui correspond à la probabilité d'appartenir à la classe 1

Probabilité d'une prédiction







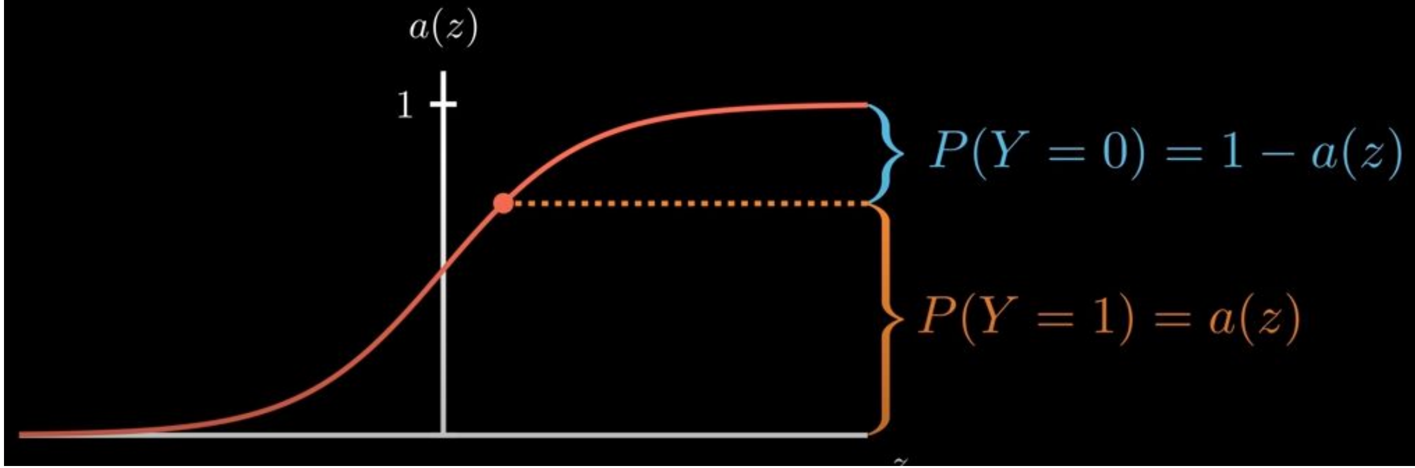
Probabilité d'une prédiction

Loi de Bernoulli :

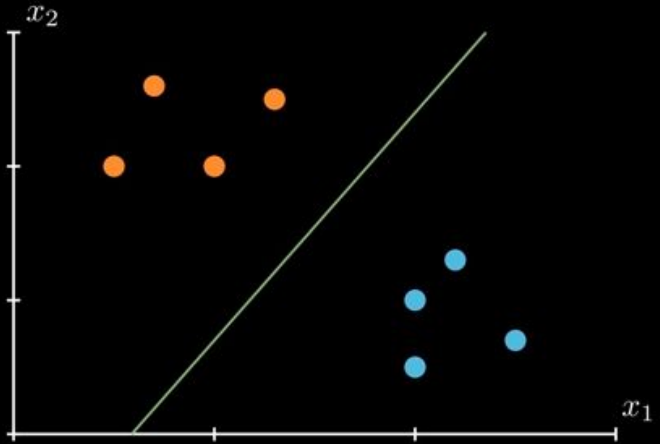
$$\boxed{P(Y = y) = a(z)^y \times (1 - a(z))^{1-y}}$$

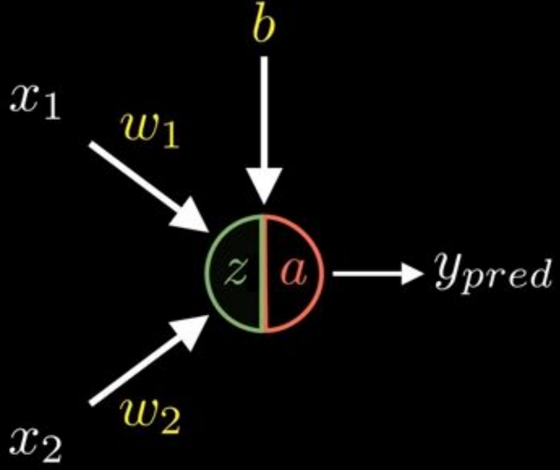
$$P(Y = 0) = a(z)^0 \times (1 - a(z))^{1-0}$$

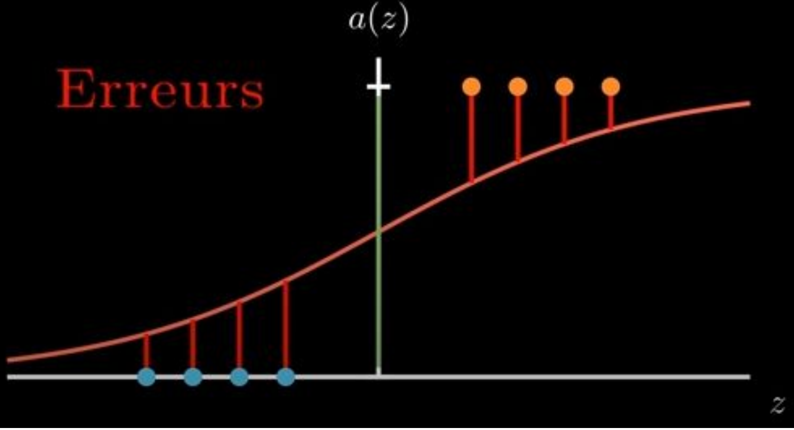
$$P(Y = 1) = a(z)^1 \times (1 - a(z))^{1-1}$$



En résumé...



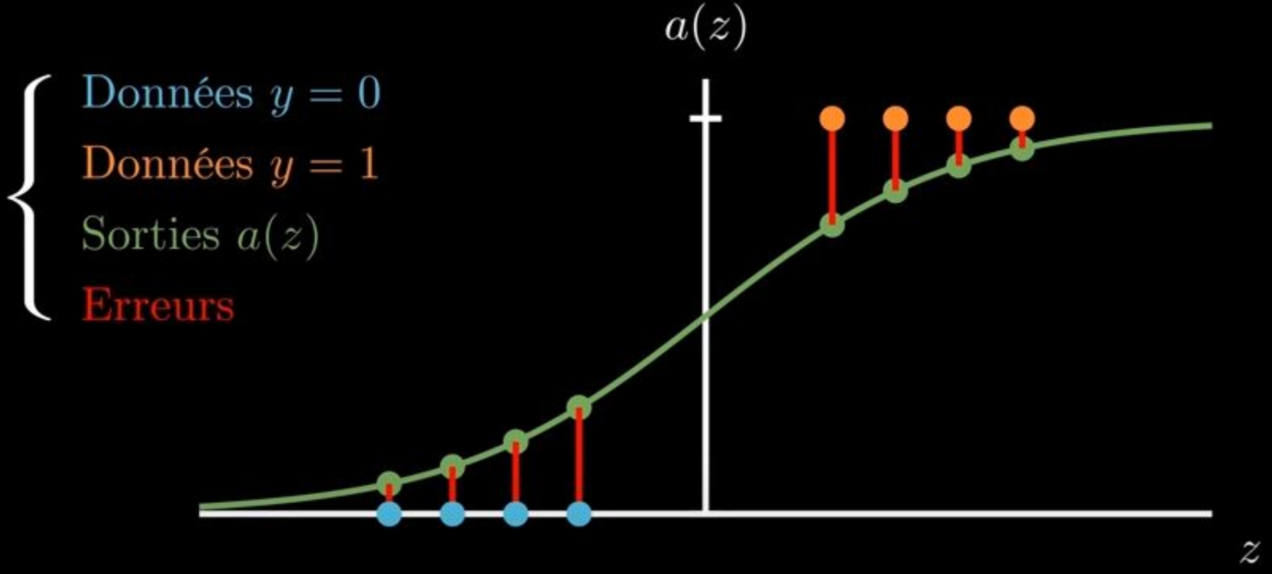




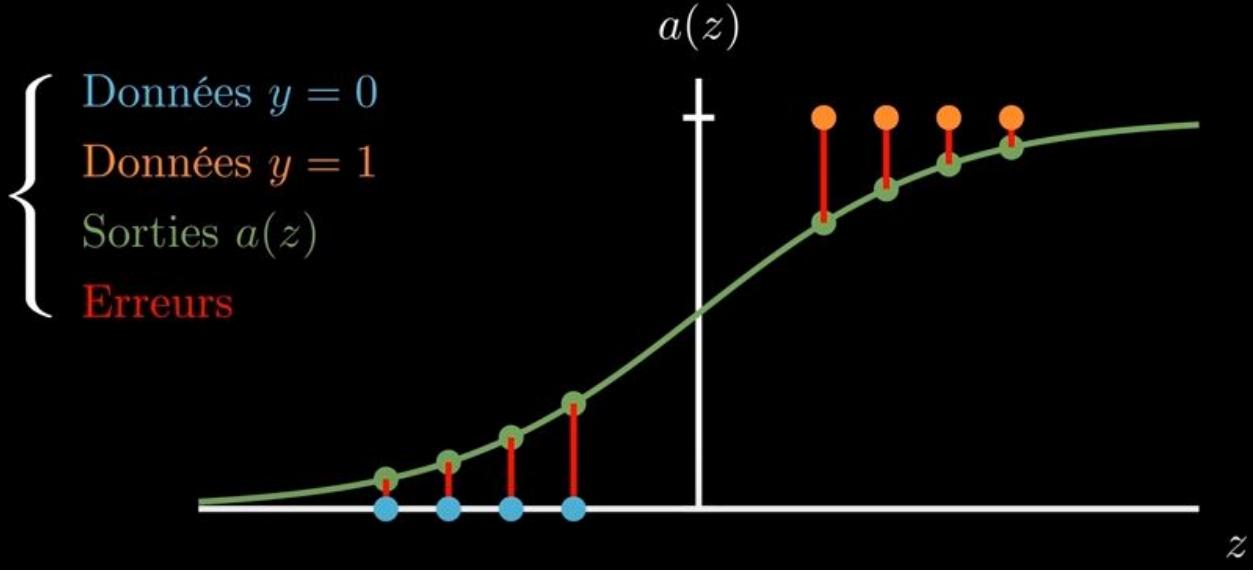
$$z = w_1 x_1 + w_2 x_2 + b$$

$$a = \frac{1}{1 + e^{-z}}$$

En Machine Learning, une Fonction Coût (Loss Function) c'est une fonction qui permet de quantifier les erreurs effectuées par un modèle.



En Machine Learning, une Fonction Coût (Loss Function) c'est une fonction qui permet de quantifier les erreurs effectuées par un modèle.



Log Loss

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

m : nombre de données
y_i : donnée n°i
a_i : sortie n°i

La Vraisemblance

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.

Invraisemblable

"J'étais à l'opéra"

Spectacle annulé

La Vraisemblance

Indique la plausibilité du modèle vis-à-vis de vraies données.

$y = 1$	$y = 1$	$y = 1$	$y = 0$	$y = 0$	$y = 0$
$P_1 = 0.9$	$P_1 = 0.7$	$P_1 = 0.8$	$P_0 = 0.9$	$P_0 = 0.8$	$P_0 = 0.7$

$$L = \prod_{i=1}^{m} P(Y = y_i)$$

La Vraisemblance

Indique la plausibilité du modèle vis-à-vis de vraies données.

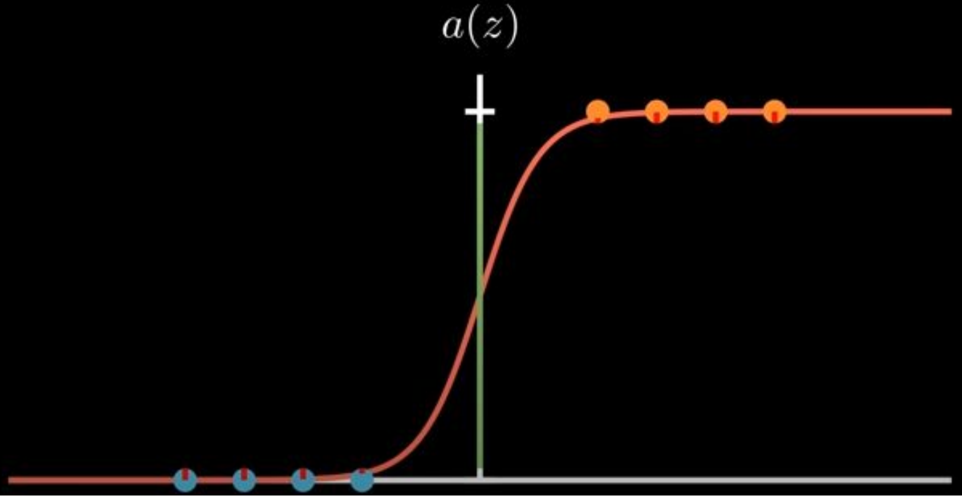
$y = 1$	$y = 1$	$y = 1$	$y = 0$	$y = 0$	$y = 0$
$P_1 = 0.9$	$P_1 = 0.7$	$P_1 = 0.8$	$P_0 = 0.9$	$P_0 = 0.8$	$P_0 = 0.7$

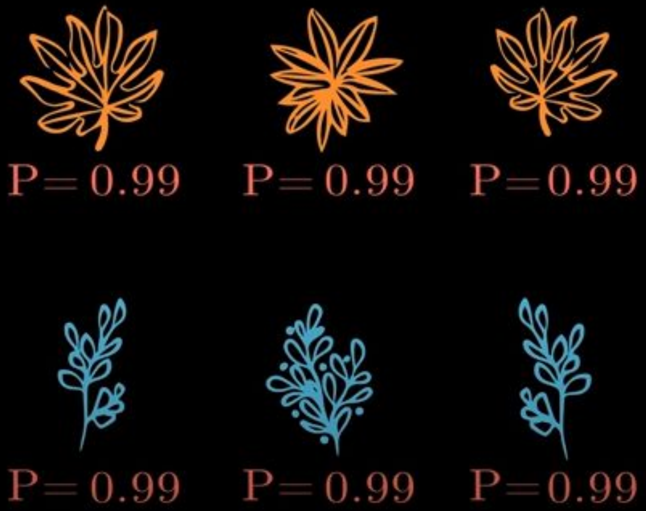
Loi de bernoulli

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

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

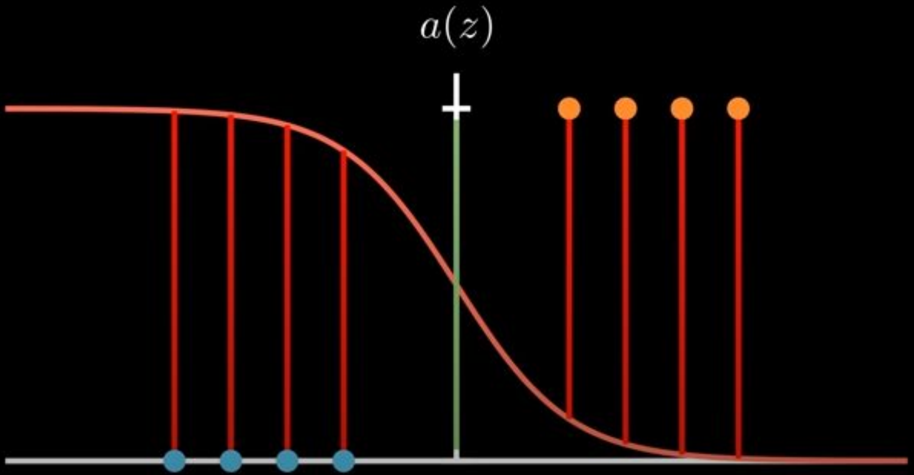
Notre modèle est vraisemblable

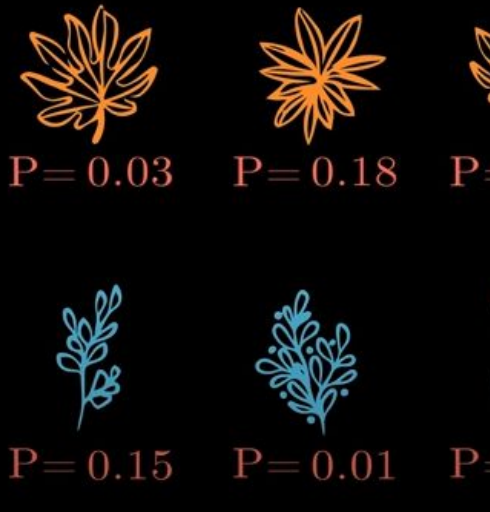




$$\boxed{L = \prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}} = 0 \ \%$$

Notre modèle n’ est pas
vraisemblable

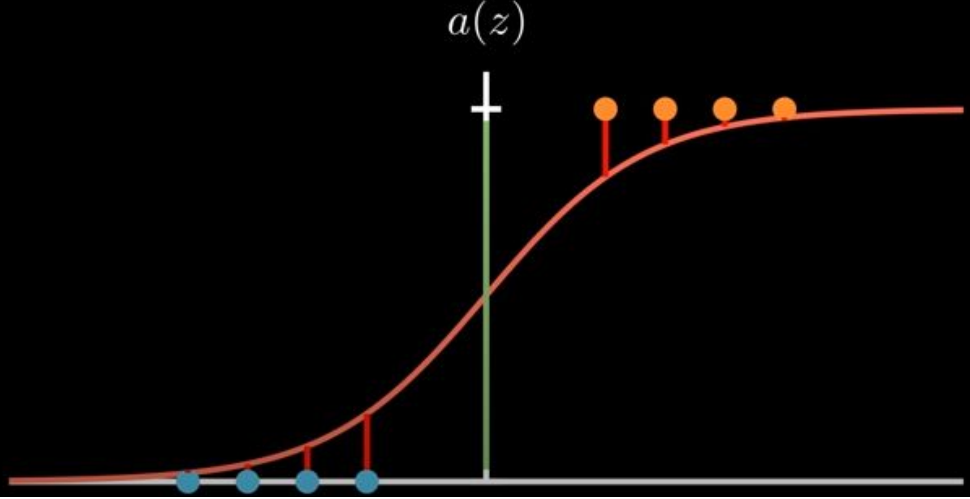


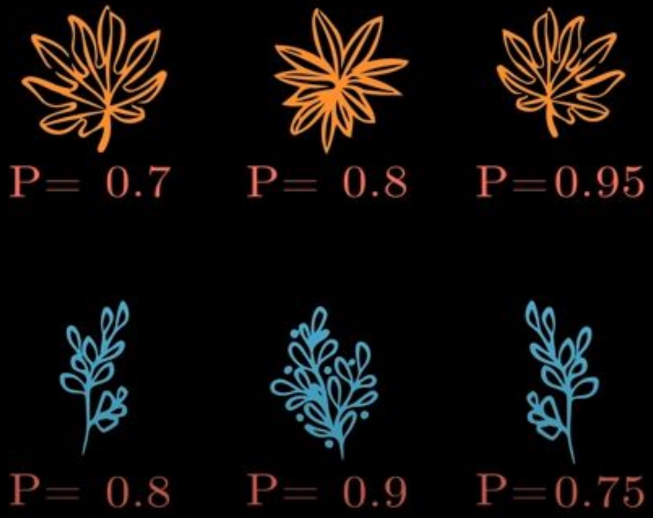


$$\boxed{L = \prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}}$$

plus il y a de nombres,
plus le résultat tend vers 0

$$L = 0.7 \times 0.8 \times 0.95 \times 0.8 \times 0.9 \times 0.75 = 0.3$$





$$\boxed{L = \prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}} \qquad log(ab) = log(a) + log(b)$$

$$L = 0.7 \times 0.8 \times 0.95 \times 0.8 \times 0.9 \times 0.75 \times \ldots\ldots \times 0.8 = 0.000\ldots.$$

underflow

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

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

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

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

$$= -1.24$$

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

$$= \sum_{i=1}^{m} log(a_i^{y_i} \times (1 - a_i)^{1-y_i})$$

Rappel :

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

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

$$\boxed{= \sum_{i=1}^{m} y_i \, log(a_i) + (1 - y_i) \, log(1 - a_i)}$$

Rappel :

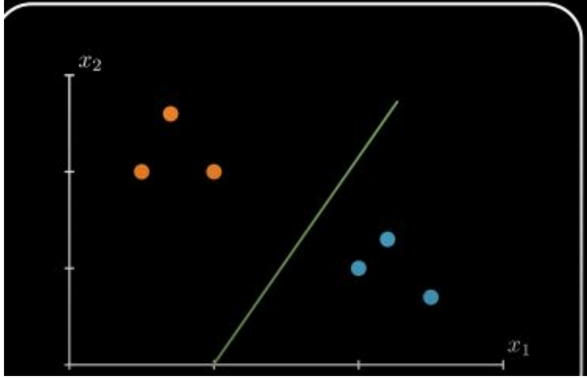
$$log(a^y) = ylog(a)$$

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

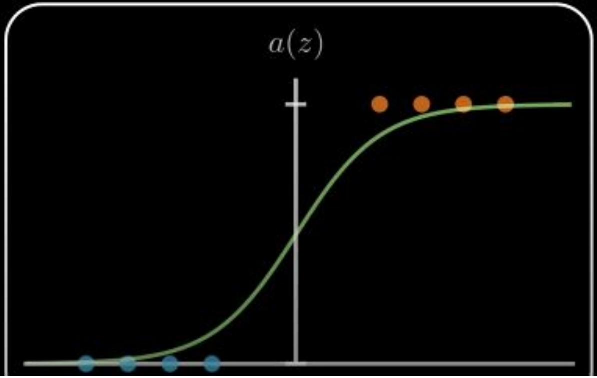
Descente de Gradient

Consiste à ajuster les paramètres W et b de façon à minimiser les erreurs du modèle, c'est-à-dire à minimiser la Fonction Coût (Log Loss)

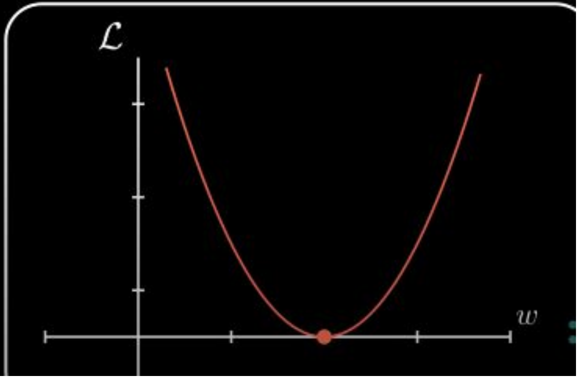
Dataset



Sigmoïde



Log Loss



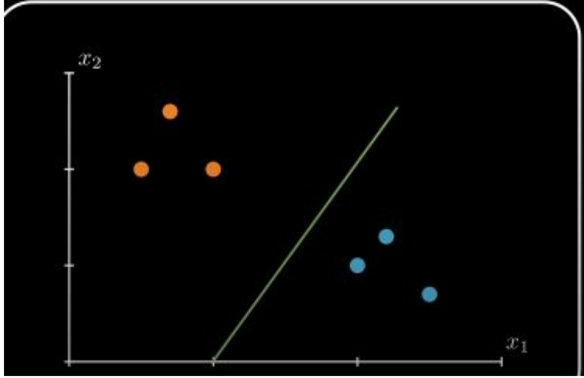
Descente de Gradient

Consiste à ajuster les paramètres W et b de façon à minimiser les erreurs du modèle, c'est-à-dire à minimiser la Fonction Coût (Log Loss)

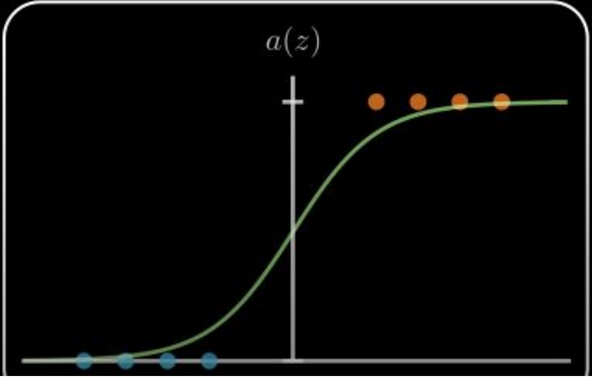
Pour ça, il faut déterminer comment est-ce-que cette fonction varie en fonction des différents paramètres.

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

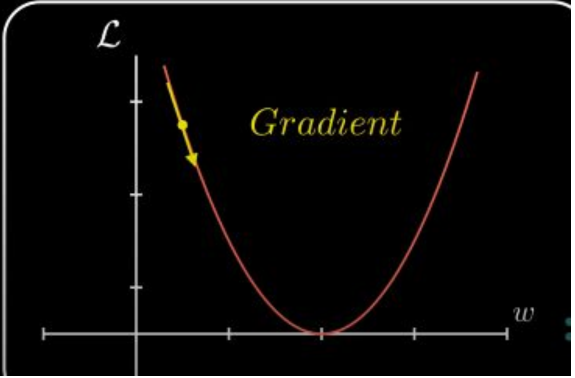
Dataset



Sigmoïde



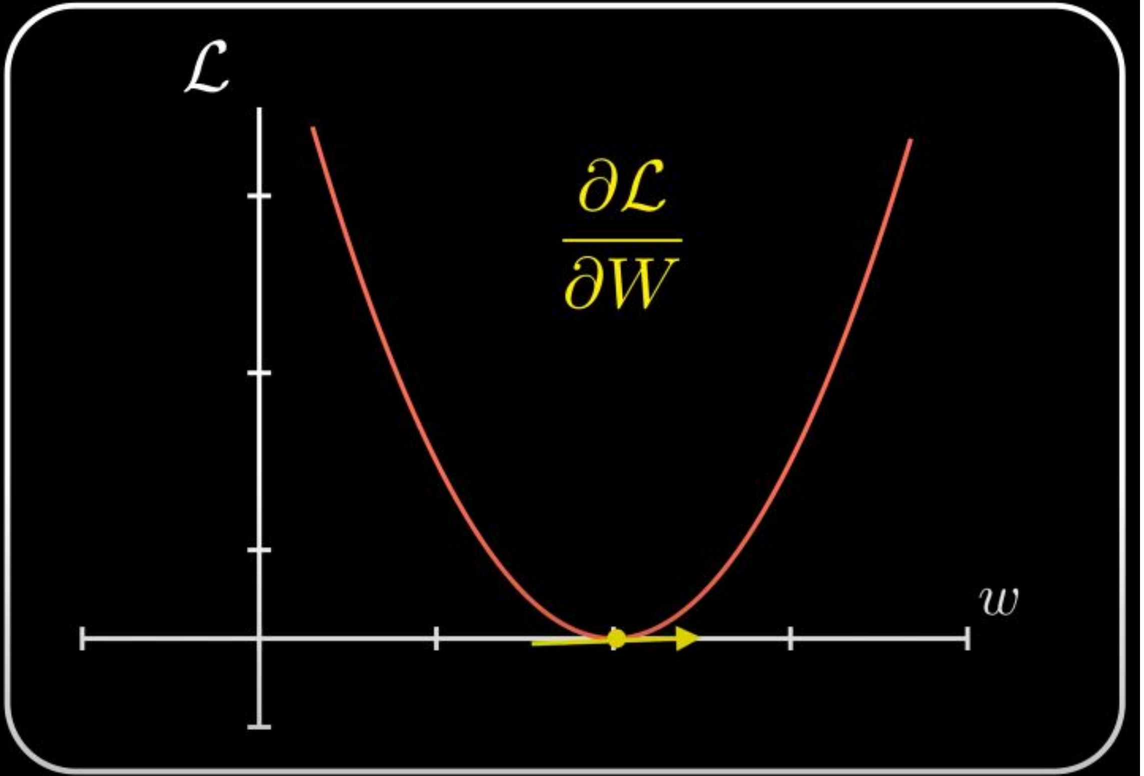
Log Loss

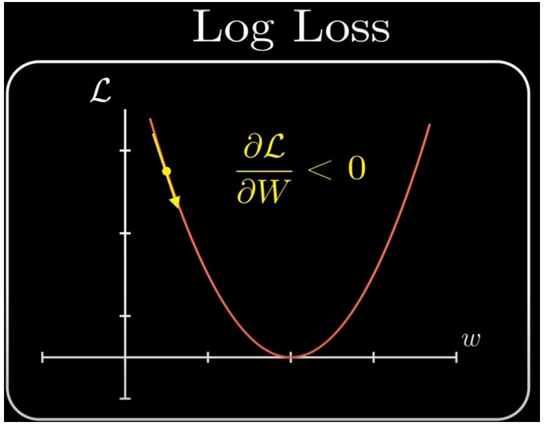


Descente de Gradient

En Mathématique,
la dérivée d'une fonction
indique comment
cette fonction varie.

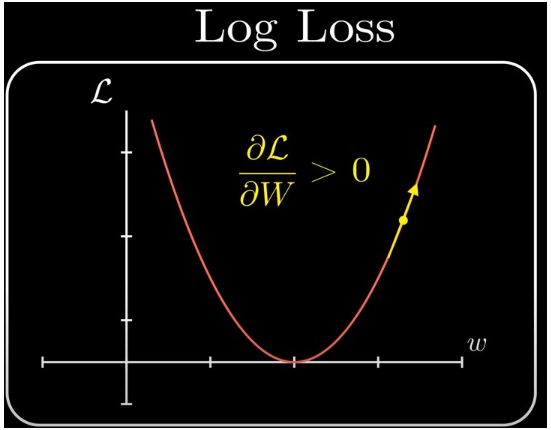
Log Loss





- Si la dérivée est négative, la fonction coût diminue quand w augmen

- Il Va donc faire augmenter w pour diminuer les erreurs



- Si la dérivée est positive, la fonction coût augmente quand w augmente.

- Il Va donc faire diminuer w pour diminuer les erreurs

Descente de Gradient

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

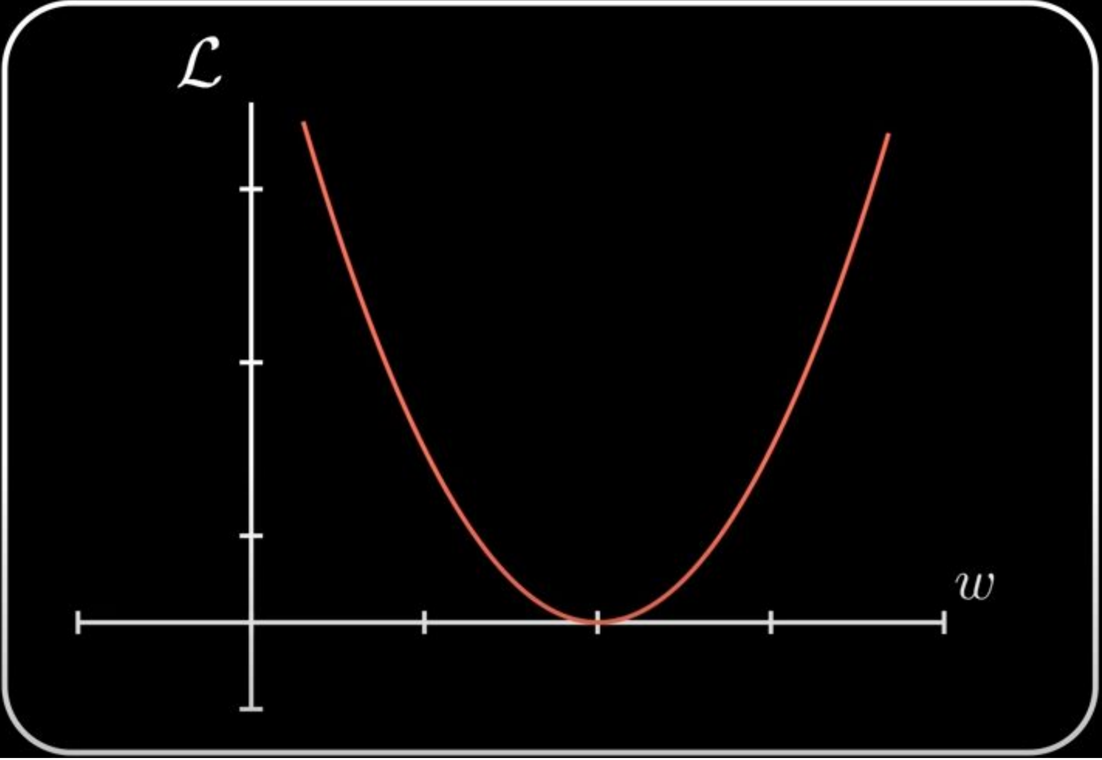
W_{t+1} : Paramètre W à l'instant t+1

W_t : Paramètre W à l'instant t

α : Pas d'apprentissage positif

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

Log Loss



Descente de Gradient

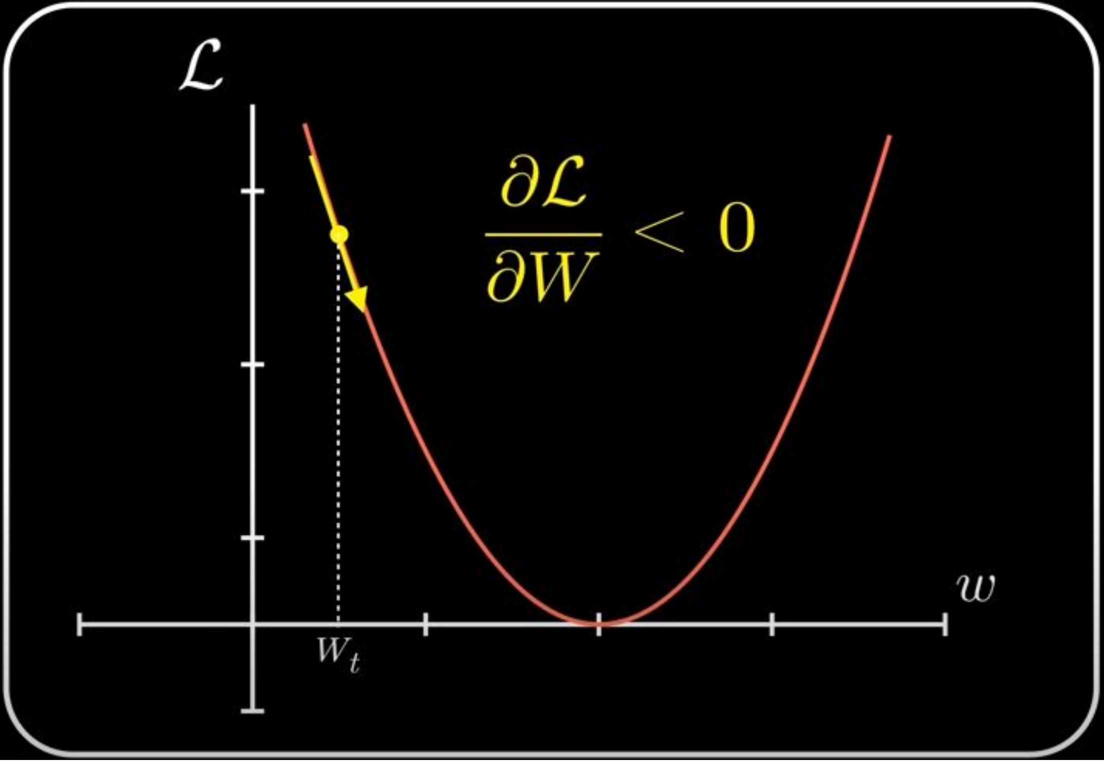
$$W_{t+1} = W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

$$= W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

$$= W_t + \text{qqch}_{\text{positif}}$$

w augmente

Log Loss

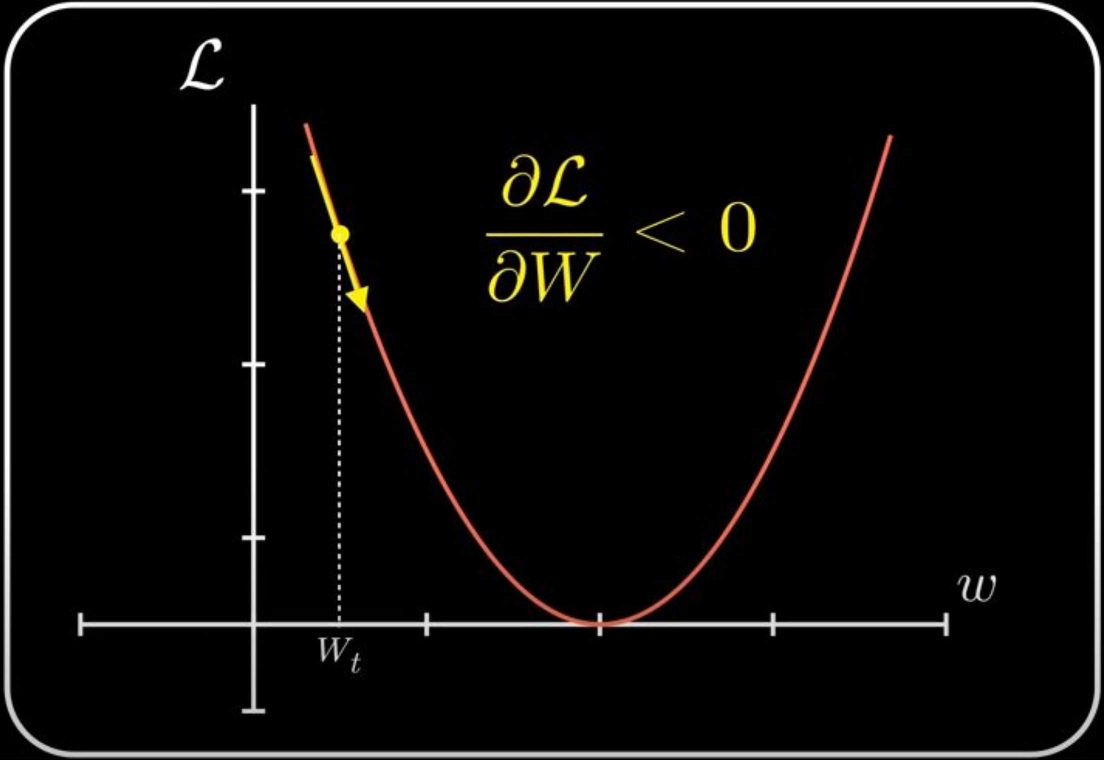


Descente de Gradient

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

w augmente

Log Loss



Descente de Gradient

Log Loss

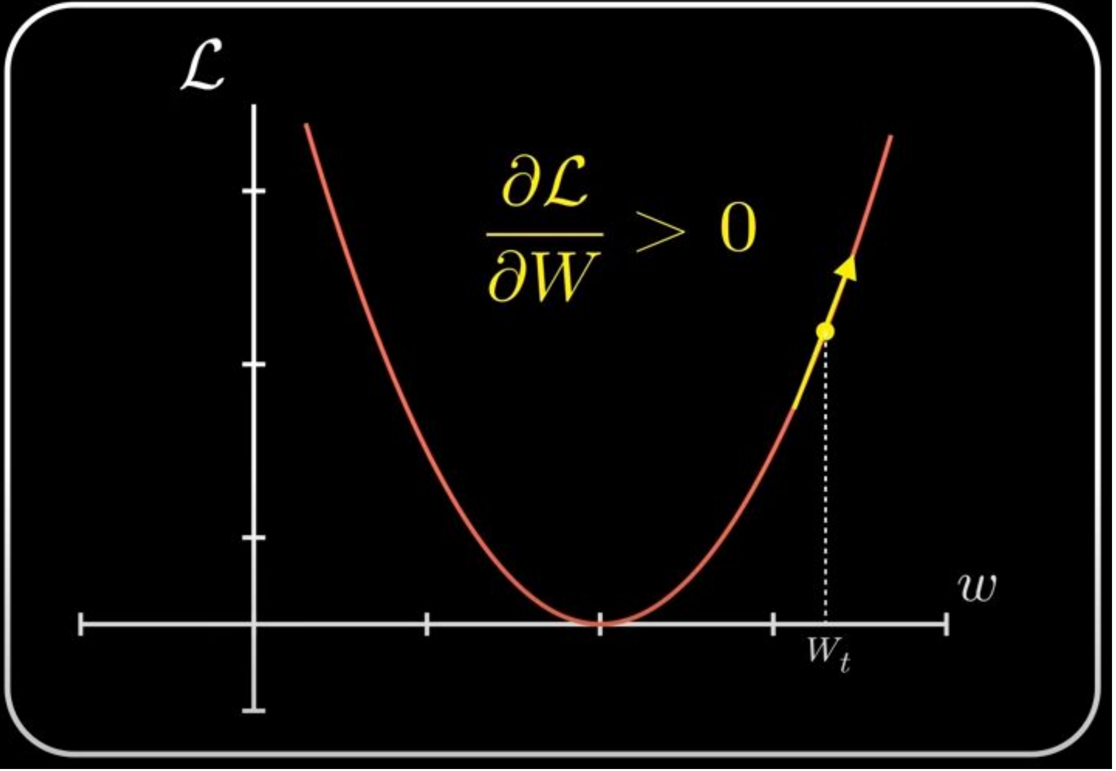
$$W_{t+1} = W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

$$= W_t - \boxed{\alpha \frac{\partial \mathcal{L}}{\partial W_t}}$$

$$> 0$$

$$= W_t + \text{qqch}_{\text{négatif}}$$

w diminue

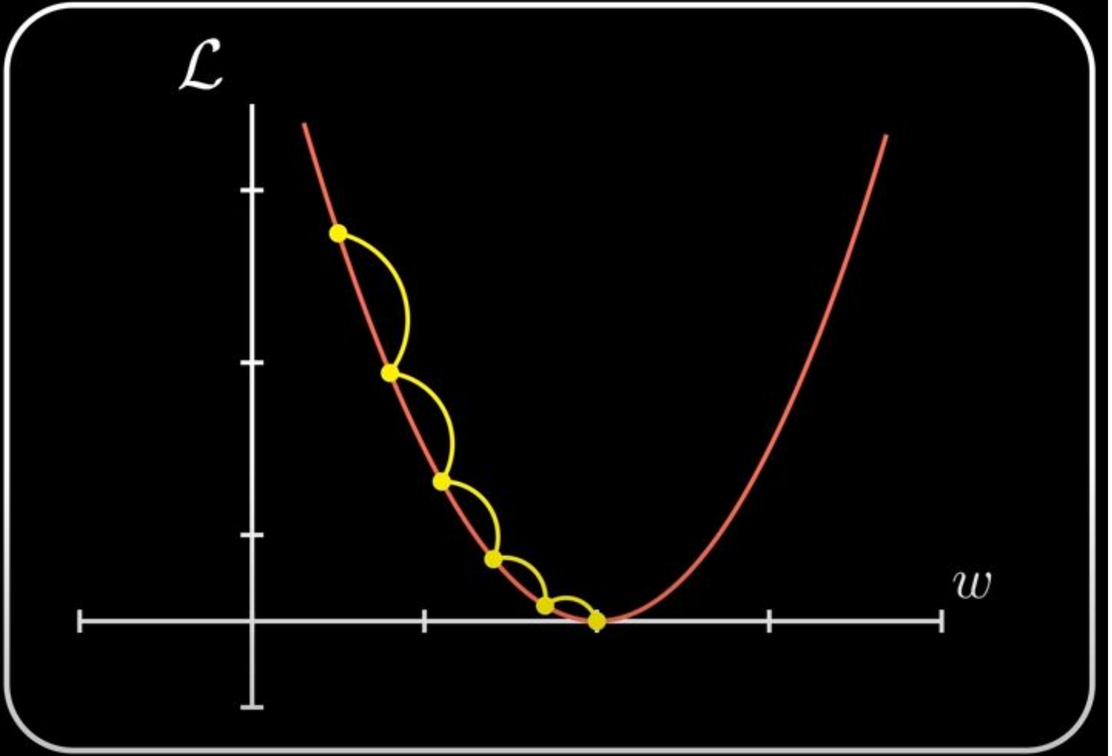


Descente de Gradient

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

En répétant cette formule en boucle, on est capable d'atteindre le minimum de la fonction coût en descendant progressivement sa courbe

Log Loss



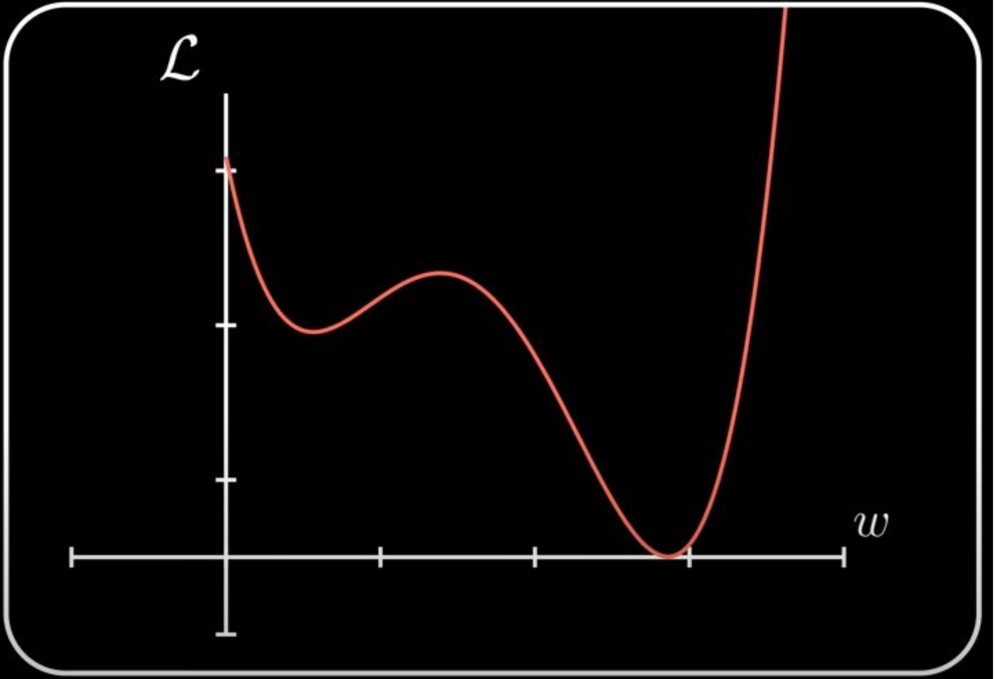
Descente de Gradient

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

Pour que ça marche la fonction coût doit être convexe et continue

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

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

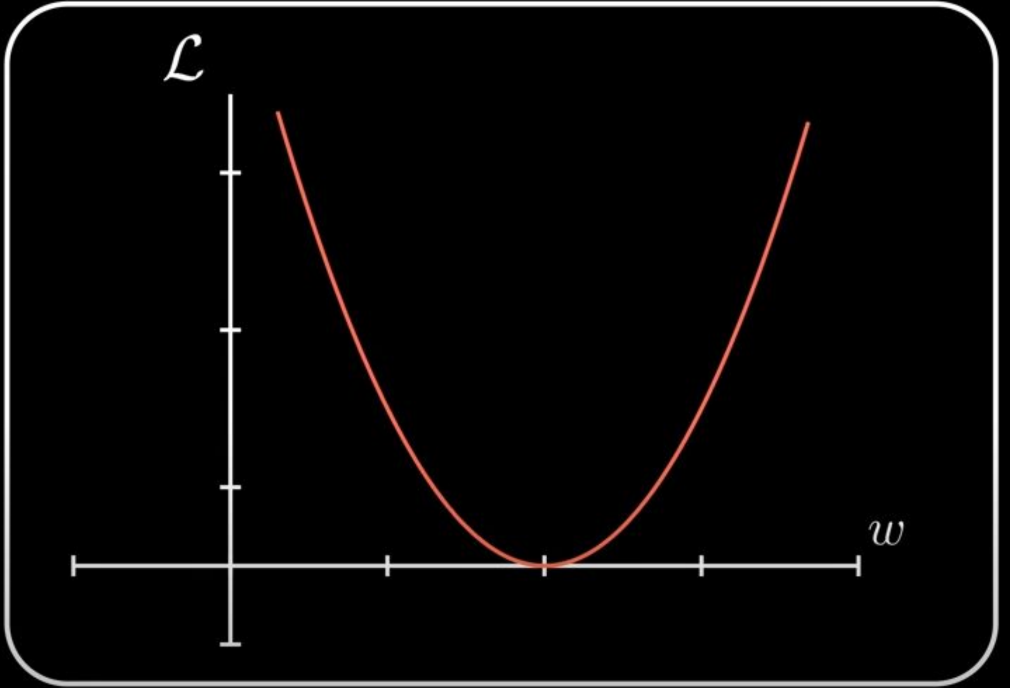


Descente de Gradient

Log Loss

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

Log Loss est une fonction convexe et continue



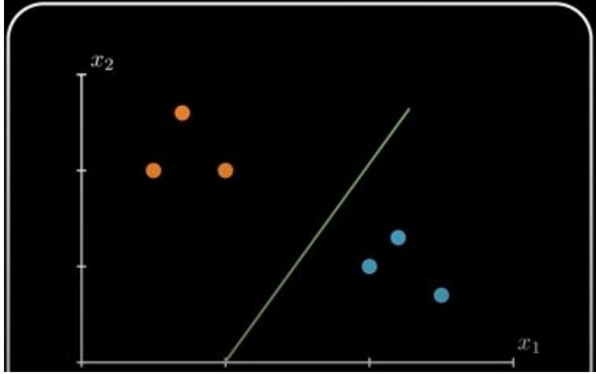
$$z = w_1x_1 + w_2x_2 + b$$

$$a = \frac{1}{1 + e^{-z}}$$

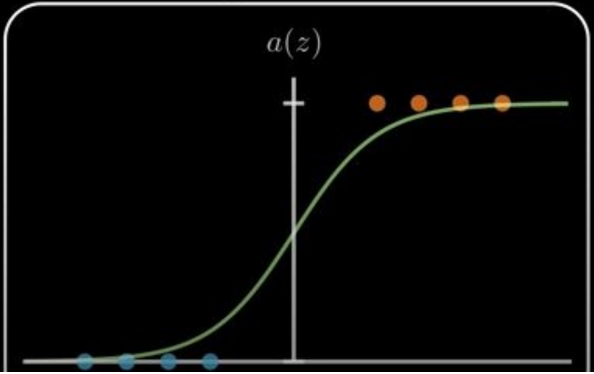
$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} y_i log(a_i) + (1 - y_i) log(1 - a_i)$$

$$W = W - \alpha\frac{\partial \mathcal{L}}{\partial W}$$

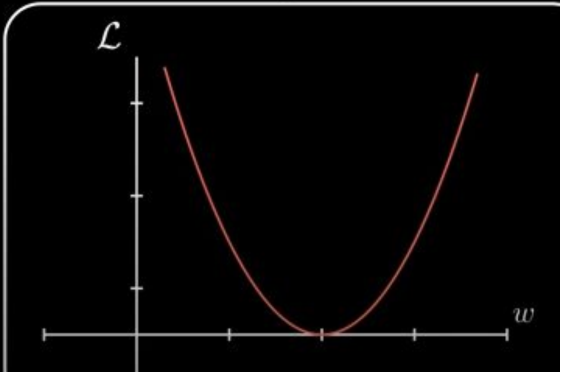
Dataset



Sigmoïde



Log Loss



$$z = w_1 x_1 + w_2 x_2 + b$$

$$a = \frac{1}{1 + e^{-z}}$$

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

$$W = W - \alpha \frac{\partial \mathcal{L}}{\partial W}$$ Comment calcule-t-on cela ?