Les neurones

NSI
Terminale

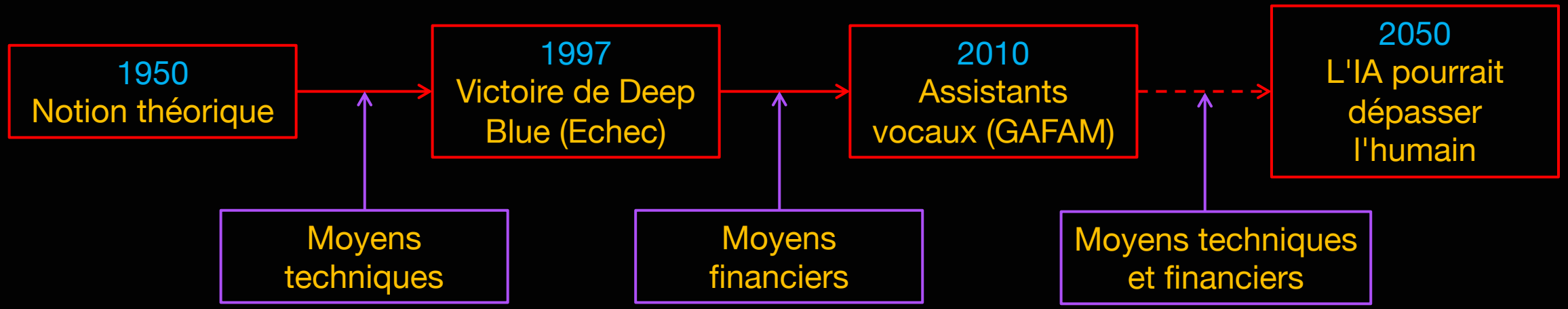
Définitions

DEFINITIONS

Généralités :

- L'Intelligence Artificielle (AI) sont les techniques qui cherchent à comprendre et reproduire le fonctionnement d'un cerveau humain.

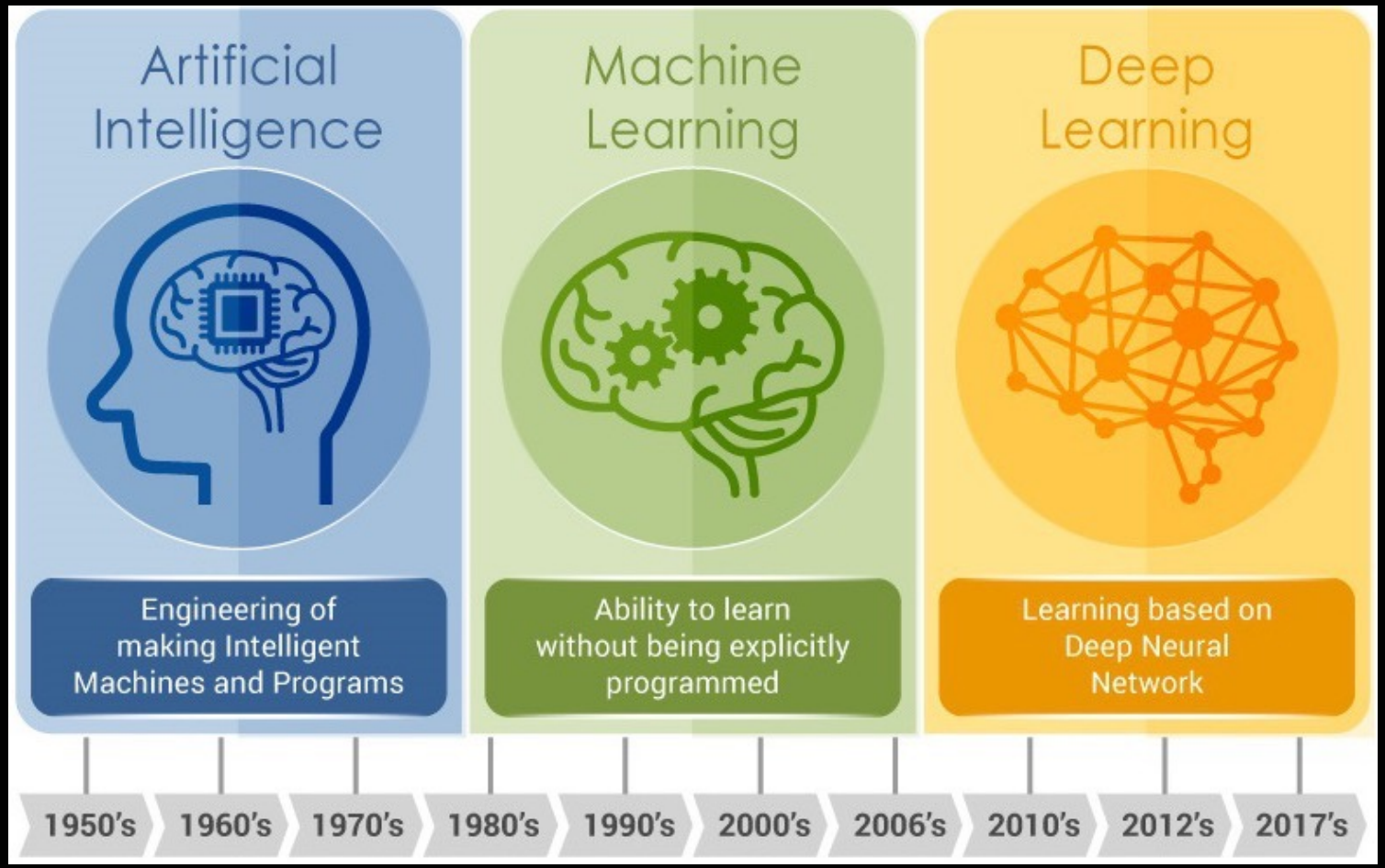
- Historique rapide :



DEFINITIONS

Machine Learning et Deep Learning :

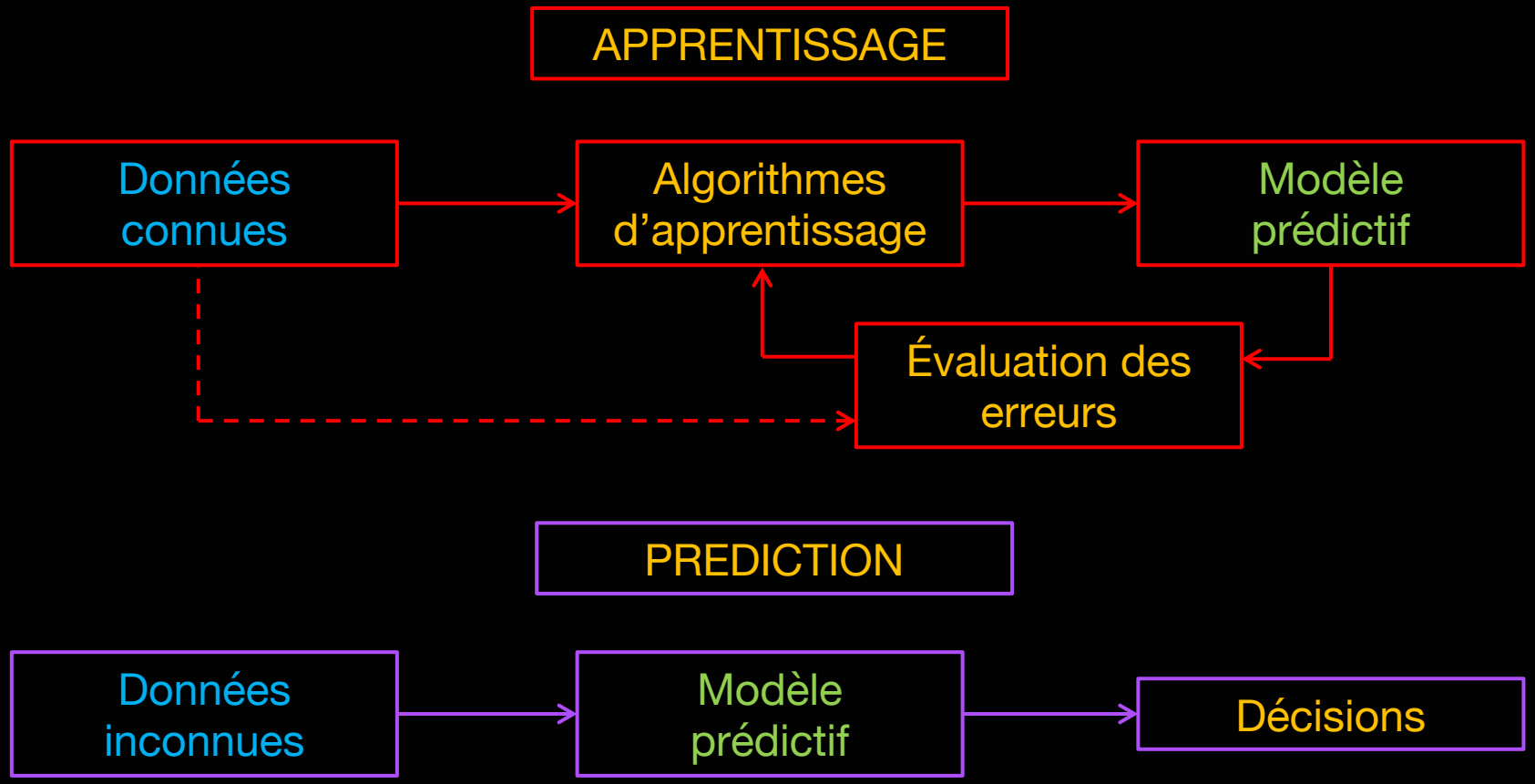
- Il existe des sous-ensembles de l'intelligence artificielle :



DEFINITIONS

Principe :

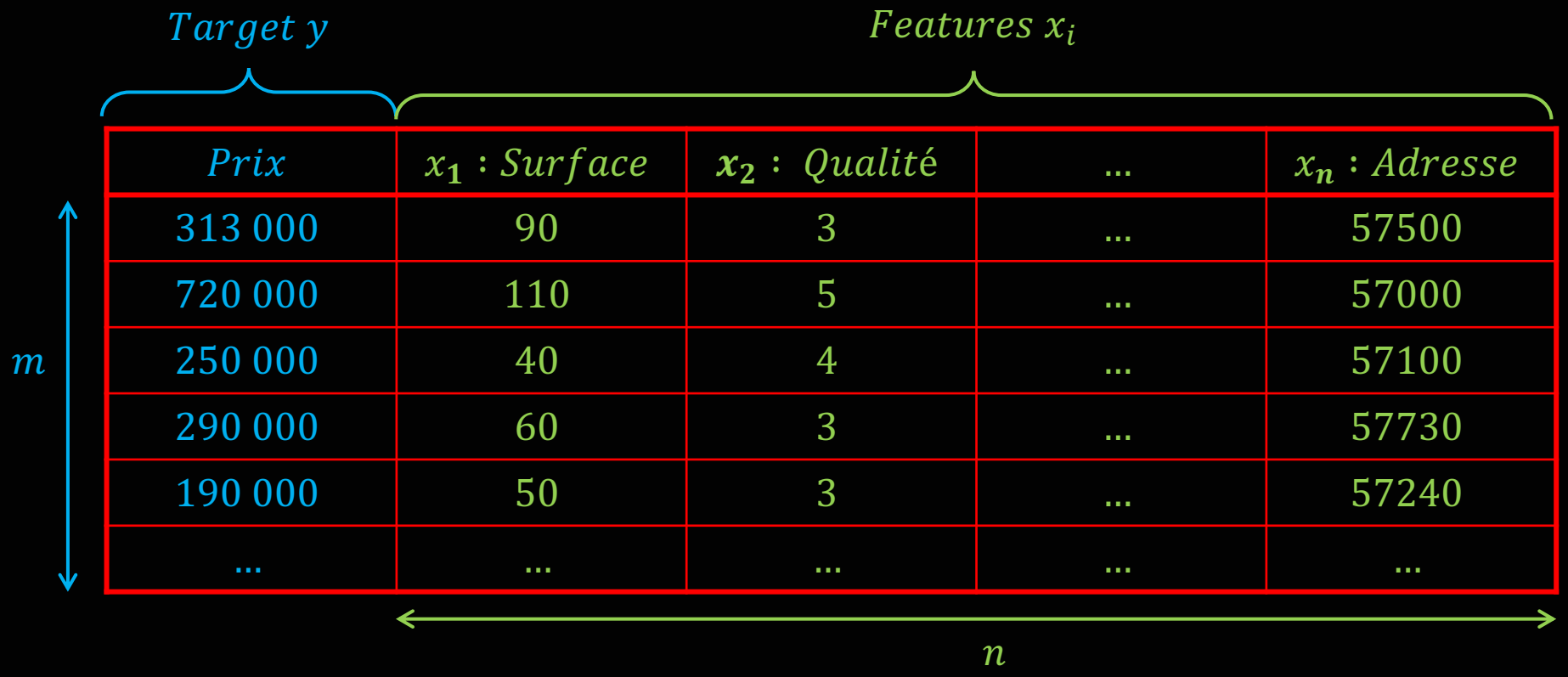
- Il existe deux grandes phases :



DEFINITIONS

Les données :

- Les m données (Dataset) se décomposent en :
  - La Target y (ou le Label) : cible que l'on cherche.
  - Les n Features x_i : entrées qui influent sur la Target.

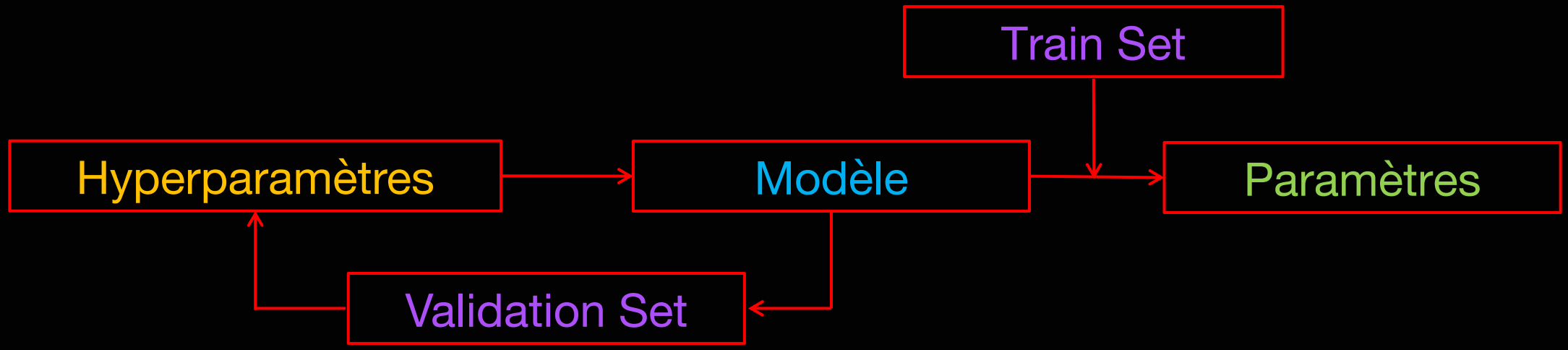


$Prix$	$x_1 : Surface$	$x_2 : Qualité$	…	$x_n : Adresse$
313 000	90	3	…	57500
720 000	110	5	…	57000
250 000	40	4	…	57100
290 000	60	3	…	57730
190 000	50	3	…	57240
…	…	…	…	…

DEFINITIONS

Les jeux de données (Dataset) :

- L'ensemble des données (Dataset) se séparent en trois parties :
  - Le Train Set (80%) : il permet d'entrainer le modèle pour ajuster les paramètres.
  - Le Validation Set $(x\%\ des\ 80\%)$: il permet d'ajuster les hyperparamètres.
  - Le Test Set (20%) : il permet de vérifier que le modèle est bien entrainé.



DEFINITIONS

Mesures de la performance d'un modèle :

- Les deux principales notions permettant de mesurer la performance d'un modèle sont :
    - Accuracy (exactitude) : calcul du rapport entre le nombre de prédictions correctes sur l'ensemble des individus.
      Plus la précision est grande, plus le modèle est performant.

    - Loss (perte) : calcul de la différence entre le résultat prédit et le résultat attendu.
      Plus la perte est minime, plus le modèle est performant.

DEFINITIONS

Exemple :

- Exemple d'une population de 200 individus pour prédire qui est malade et qui est sain :
  - Données réelles : 135 malades, 65 sains
  - Données prédites : 140 malades, 60 sains

Données réelles \ Données prédites	Malade	Sain	
Malade	$TP : 120$	$FP : 15$	135
Sain	$FN : 20$	$TN : 45$	65
	140	60	200

DEFINITIONS

Exemple :

- Calcul de l'*accuracy* :

		Données prédites : Malade	Données prédites : Sain	
Données réelles	Malade	TP : 120	FP : 15	135
Données réelles	Sain	FN : 20	TN : 45	65
		140	60	200

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$accuracy = \frac{120 + 45}{120 + 45 + 15 + 20} = \frac{165}{200} = 0{,}82 = 82\ \%$$

- Cela signifie que sur 100 prévisions, 82 seront justes.

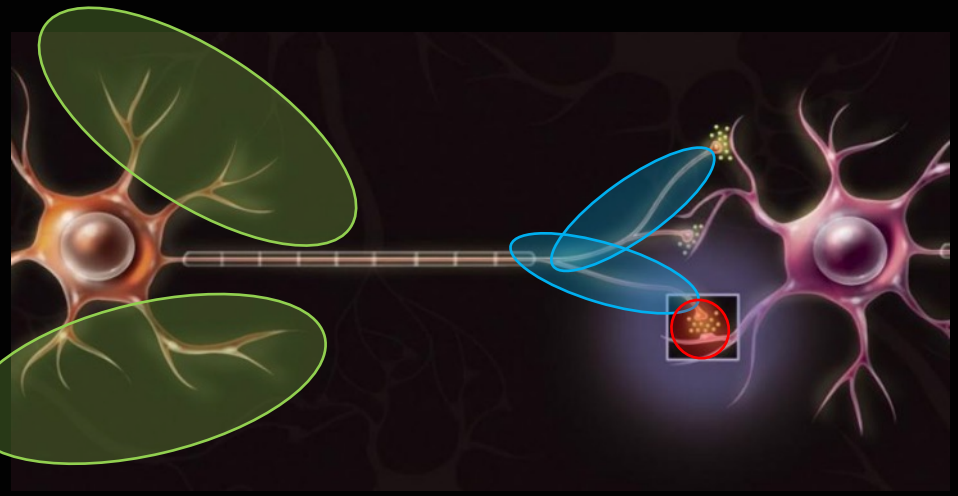
Le perceptron

LE PERCEPTRON

Le neurone :

- Le neurone est l'unité de travail de base du cerveau (entre $100\ millions$ et $100\ milliards$ chez les mammifères).

- Le neurone est composé :
  - d'un corps cellulaire
  - de dendrites
  - d'un axone.



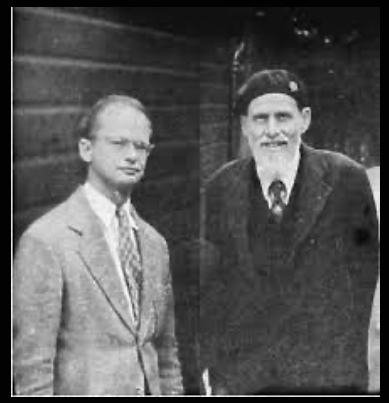
- Les dendrites sont les capteurs : lorsque les signaux reçus dépassent un seuil, le neurone est activé et envoyé un signal dans l'axone (sortie) à un autre neurone à travers la synapse.

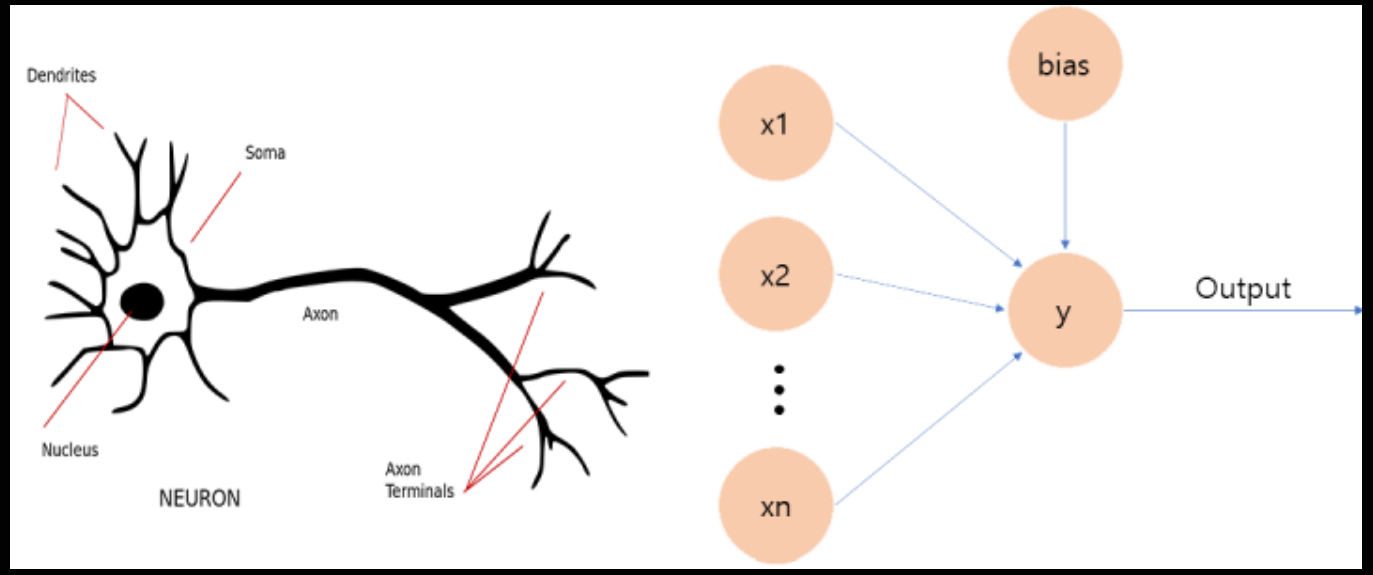
LE PERCEPTRON

Définition :

- Le perceptron cherche à reproduire le fonctionnement du neurone des mammifères.

- Le perceptron est inventé en 1957 par Rosenblatt sur les bases du neurone artificiel de Mac Culloch et Pitts, créé en 1943, mais qui n'avait pas d'algorithme d'apprentissage.

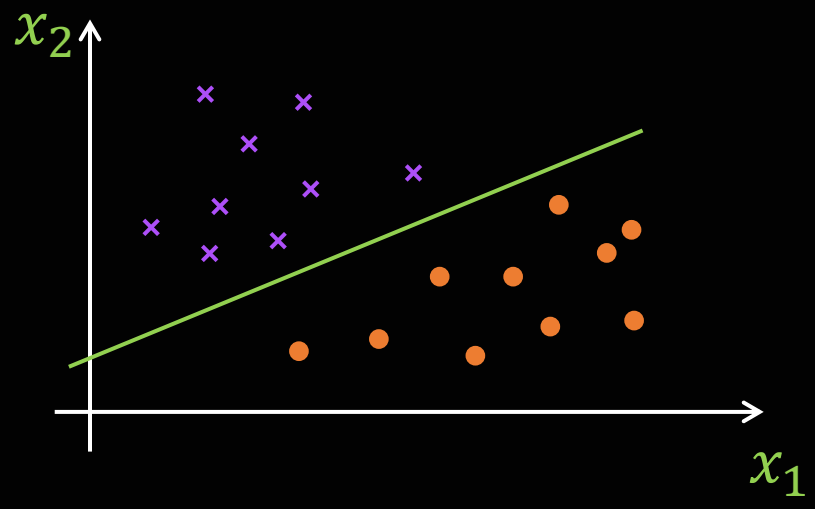




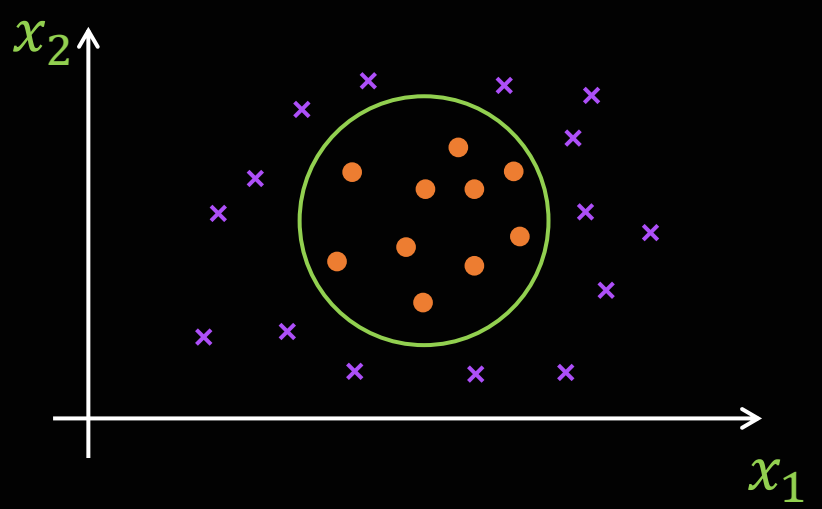
LE PERCEPTRON

Définition :

- Un perceptron est capable de séparer linéairement deux classes :



$Classe\ 1$
$Classe\ 0$



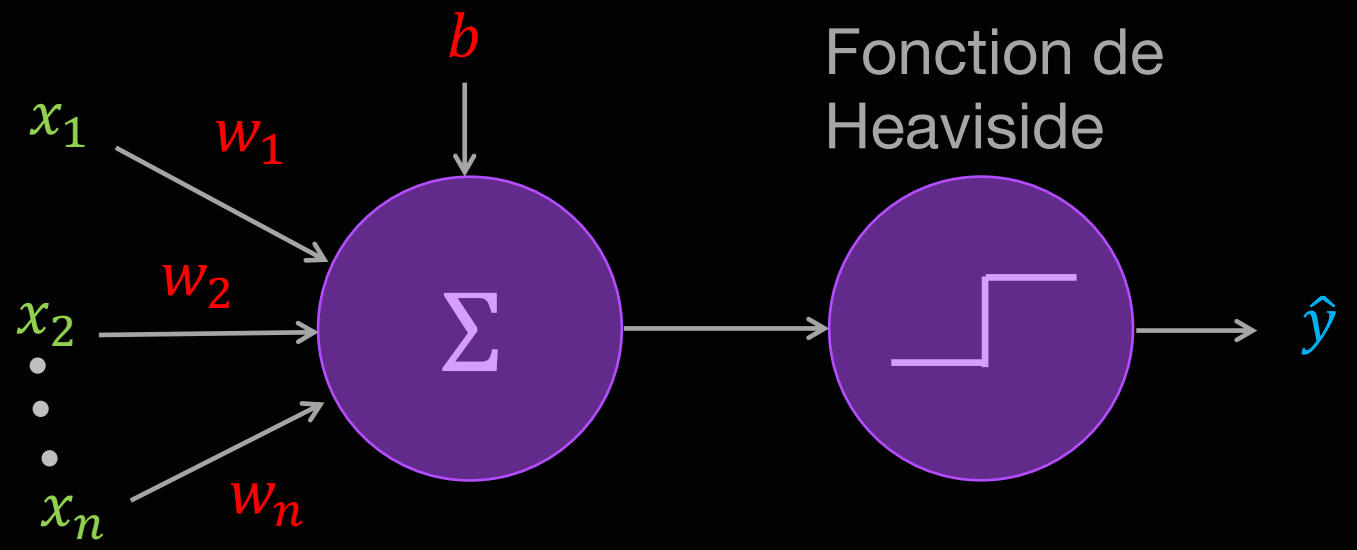
Classification binaire linéaire

Classification binaire non linéaire

LE PERCEPTRON

Mise en œuvre :

- Schéma :



- Le perceptron possède :
  - Des entrées $X(x_1, x_2, \ldots, x_n)$ (dendrites)
  - Des poids $W(w_1, w_2 \ldots, w_n)$ et un biais b (fonctionnement du neurone)
  - Une sortie $\hat{y}$ (axone)
- La sortie est donnée par la fonction de Heaviside : $\hat{y} = \begin{cases} 1 & si \ \sum_i w_i x_i + b \geq 0 \\ 0 & si \ \sum_i w_i x_i + b < 0 \end{cases}$

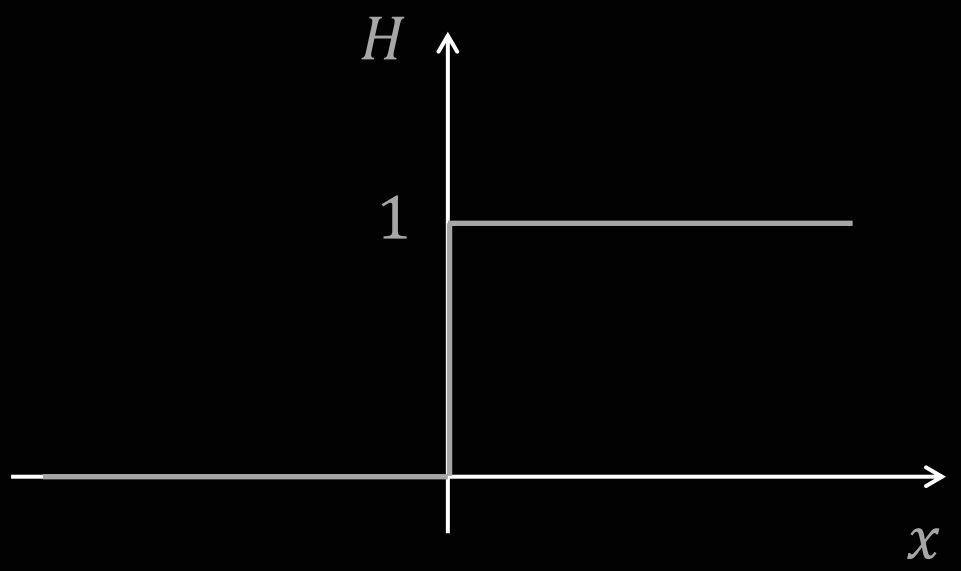
LE PERCEPTRON

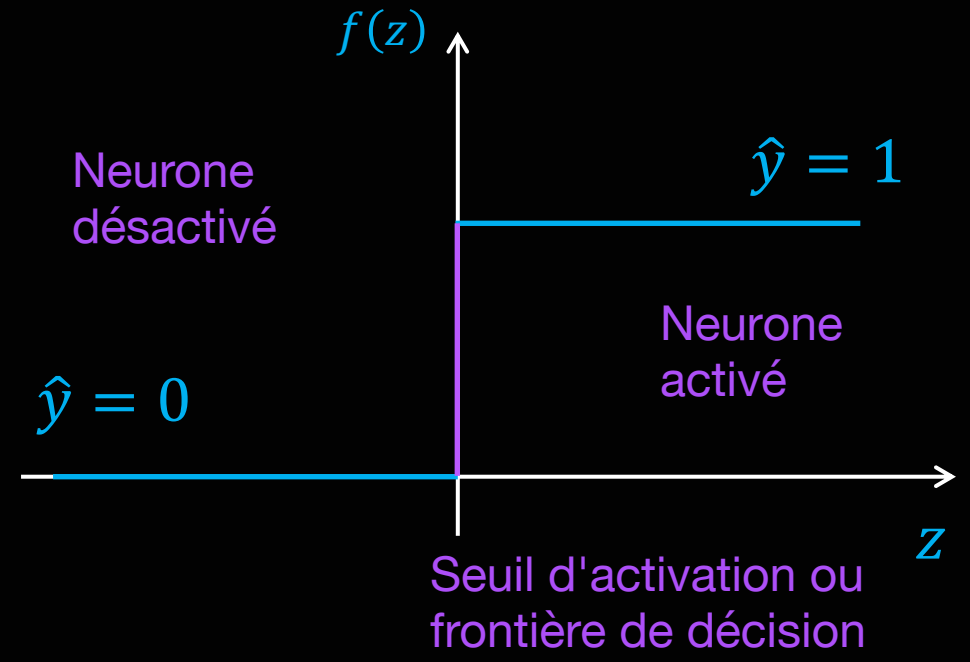
Activation du neurone :

- La fonction de Heaviside :

$$H(x) = \begin{cases} Si\ x \geq 0 \Longrightarrow f(x) = 1 \\ Si\ x < 0 \Longrightarrow f(x) = 0 \end{cases}$$

$$\hat{y} = \begin{cases} 1 & si\ z = \sum_i w_i x_i + b \geq 0 \\ 0 & si\ z = \sum_i w_i x_i + b < 0 \end{cases}$$

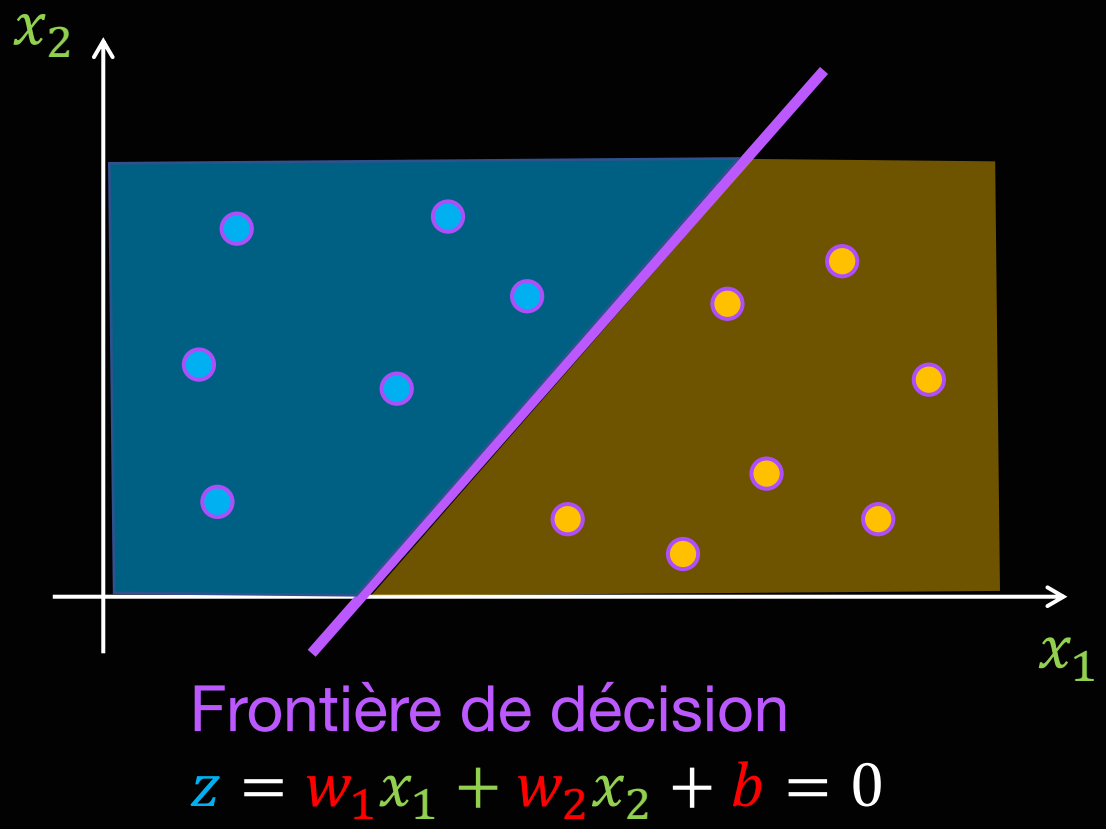




LE PERCEPTRON

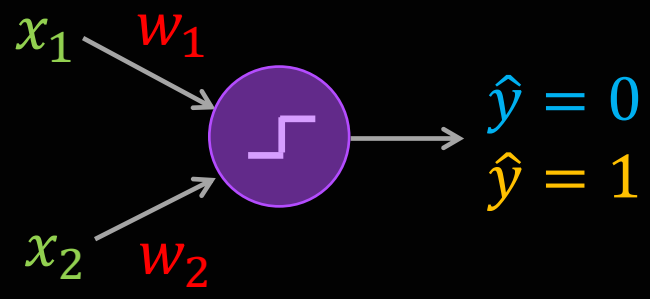
Séparation de 2 classes :

- Exemple de 2 classes points bleus et points oranges :



Frontière de décision

$z = w_1 x_1 + w_2 x_2 + b = 0$



$z = w_1 x_1 + w_2 x_2 + b$

$$\begin{cases} \hat{y} = 0 \; si \; z < 0 \\ \hat{y} = 1 \; si \; z \geq 0 \end{cases}$$

- Il « suffit » de faire apprendre le perceptron, donc trouver w_1 et w_2.

LE PERCEPTRON

Algorithme d'apprentissage :

- L'algorithme d'apprentissage est basé sur la théorie de Hebb sur le renforcement : on entraine le perceptron (calcul des poids $W(w_1, w_2 \dots, w_n)$) sur des entrées (features) $X(x_1, x_2, \dots, x_n)$ et une sortie (target) connue y.

- Le calcul du nouveau poids $W'(w'_1, w'_2 \dots, w'_n)$ est :

$$\forall\, i \in [1, n] \;\; w'_i = w_i + \alpha(y - \hat{y})x_i$$

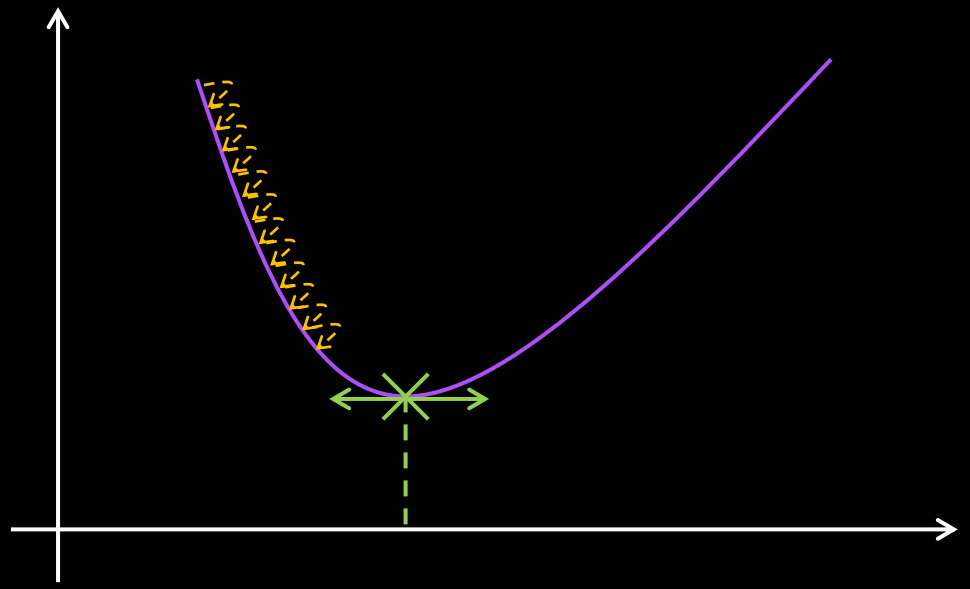
avec l'hyperparamètre α (taux d'apprentissage ou learning rate), positif et l'entrée x_i la force du signal.

- Si α est trop petit => lenteur de convergence
  Si α est trop grand => oscillations
  En général : $0{,}001 < \alpha < 0{,}1$

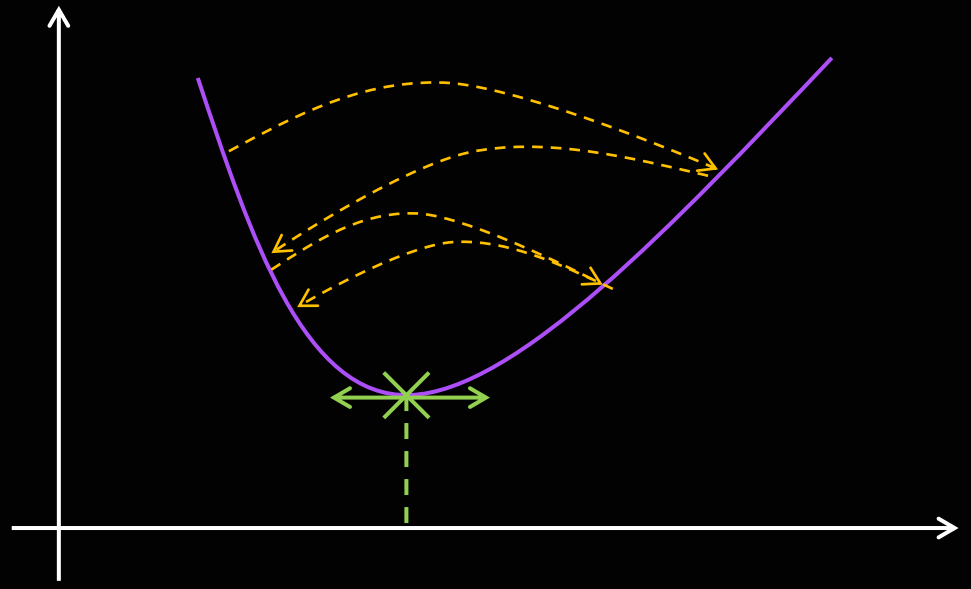
Importance du learning rate :

- L'hyperparamètre α (learning rate) est déterminant pour la convergence de l'algorithme.

α trop petit $\Longrightarrow$ convergence trop lente



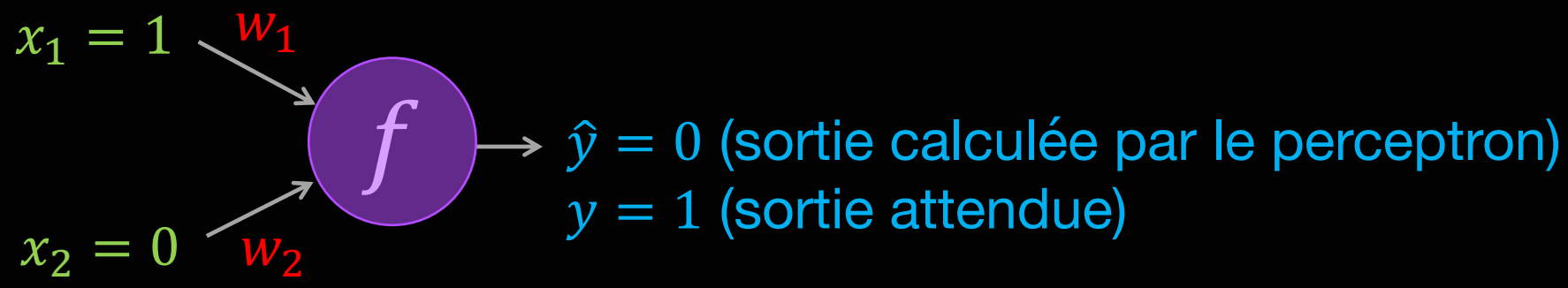
α trop grand $\Longrightarrow$ oscillations



LE PERCEPTRON

Fonctionnement du perceptron :

- Exemple d'un perceptron à 2 entrées : $z = w_1 x_1 + w_2 x_2$



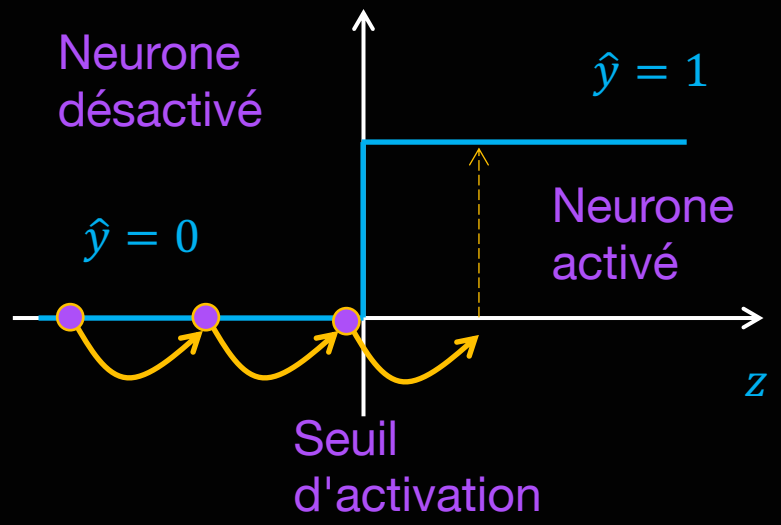
- On obtient :

$$\begin{cases} w_1' = w_1 + \alpha(y - \hat{y})x_1 = w_1 + \alpha \\ w_2' = w_2 + \alpha(y - \hat{y})x_2 = w_2 \end{cases}$$

- On voit que : $w_1' = w_1 + \alpha > w_1$

Fonctionnement du perceptron :

- Si $w_1' > w_1 \Longrightarrow z' = w_1' x_1 + w_2 x_2 > z \Longrightarrow z'$ s'approche du seuil d'activation.



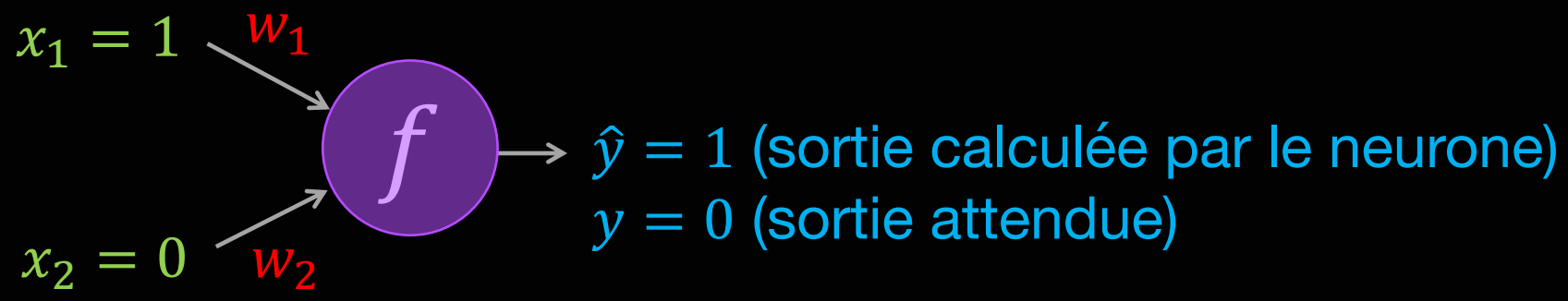
- Tant que $\hat{y} = 0$, w_1 continue d'augmenter.

- Lorsque $\hat{y} = 1$, $w_1' = w_1 + \alpha(y - \hat{y})x_1 = w_1$, w_1 cesse d'augmenter, le perceptron est activé.

LE PERCEPTRON

Fonctionnement du perceptron :

- Exemple d'un perceptron à 2 entrées : $z = w_1 x_1 + w_2 x_2$



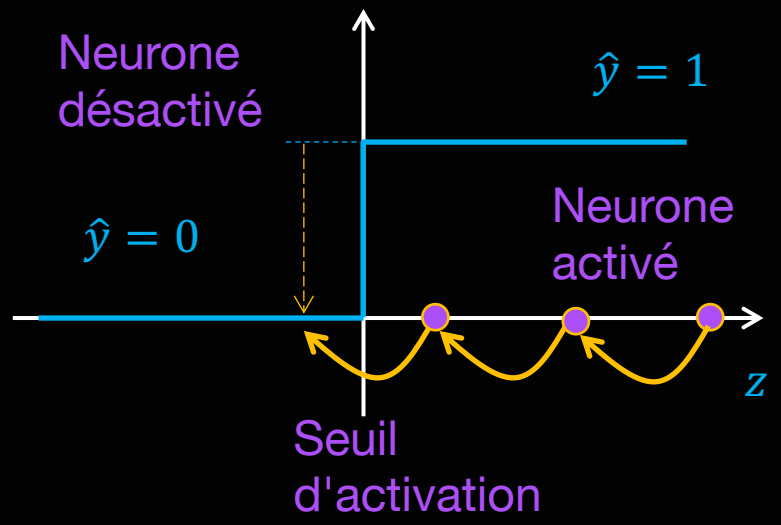
- On obtient :

$$\begin{cases} w_1' = w_1 + \alpha(y - \hat{y})x_1 = w_1 - \alpha \\ w_2' = w_2 + \alpha(y - \hat{y})x_2 = w_2 \end{cases}$$

- On voit que : $w_1' = w_1 - \alpha < w_1$

Fonctionnement du perceptron :

- Si $w_1' < w_1 \Longrightarrow z' = w_1' x_1 + w_2 x_2 < z \Longrightarrow z'$ s'approche du seuil d'activation.



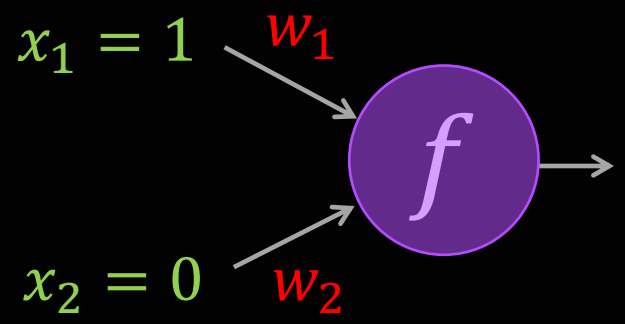
- Tant que $\hat{y} = 1$, w_1 continue de diminuer.

- Lorsque $\hat{y} = 0$, $w_1' = w_1 + \alpha(y - \hat{y})x_1 = w_1$, w_1 cesse de diminuer, le perceptron est désactivé.

LE PERCEPTRON

Fonctionnement du perceptron :

- Exemple d'un perceptron à 2 entrées : $z = w_1 x_1 + w_2 x_2$



$\hat{y} = 0\ ou\ \hat{y} = 1$ (sortie calculée par le neurone)
$y = 0\ ou\ y = 1$ (sortie attendue)

- On obtient :

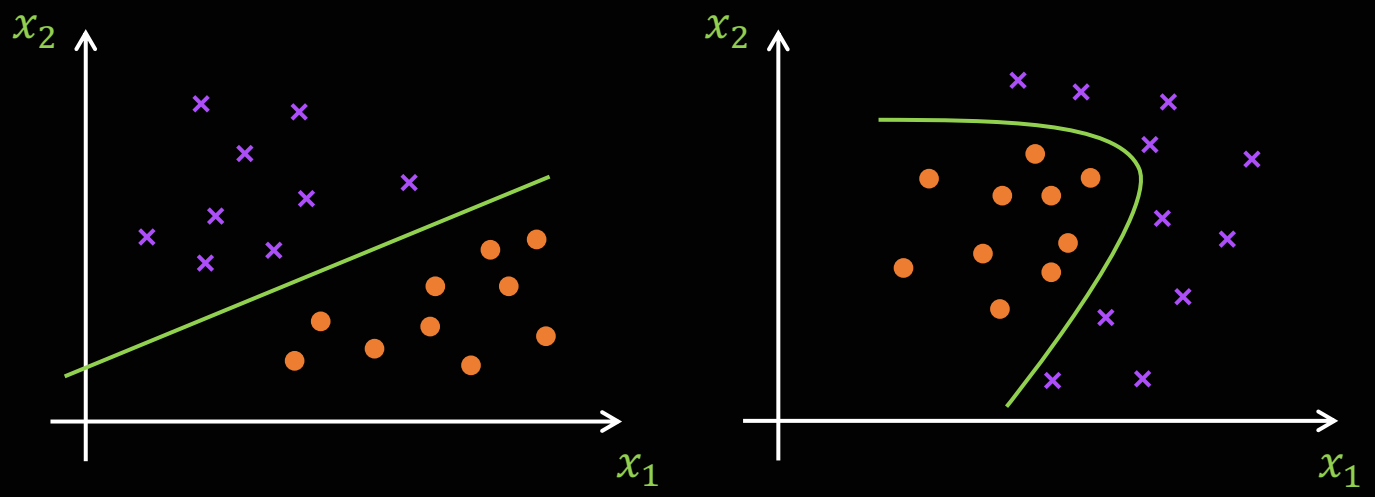
$$\begin{cases} w_1' = w_1 + \alpha(y - \hat{y})x_1 = w_1 \\ w_2' = w_2 + \alpha(y - \hat{y})x_2 = w_2 \end{cases}$$

- On voit que : $w_1' = w_1$, il ne se passe rien.

Le neurone

Le perceptron multicouche :

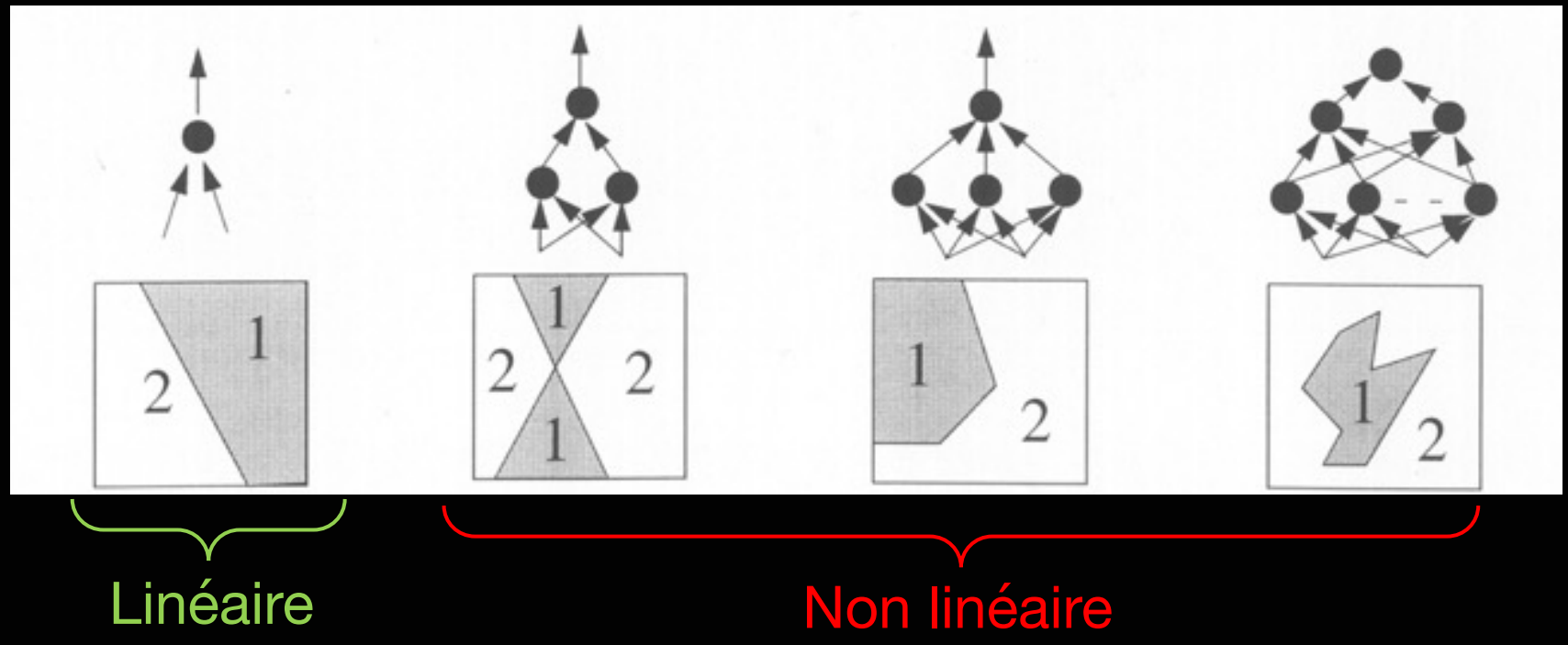
- Le perceptron simple a un gros défaut : il ne permet de résoudre qu'une tâche de classification linéaire.



- Pour pouvoir répondre à des problèmes de classification non linéaires, on fait appel aux neurones multicouches (MultiLayerPerceptron ou MLP), introduit par dans les années 80 par Hinton et Yann LeCun.

Le perceptron multicouche :

- Le perceptron multicouches (PMC ou MLP) permet de trouver une combinaison optimale de séparateurs linéaires pour produire un séparateur non linéaire.



- Plus le nombre de perceptrons augmente, plus le pouvoir séparateur augmente.

Le perceptron multicouche :

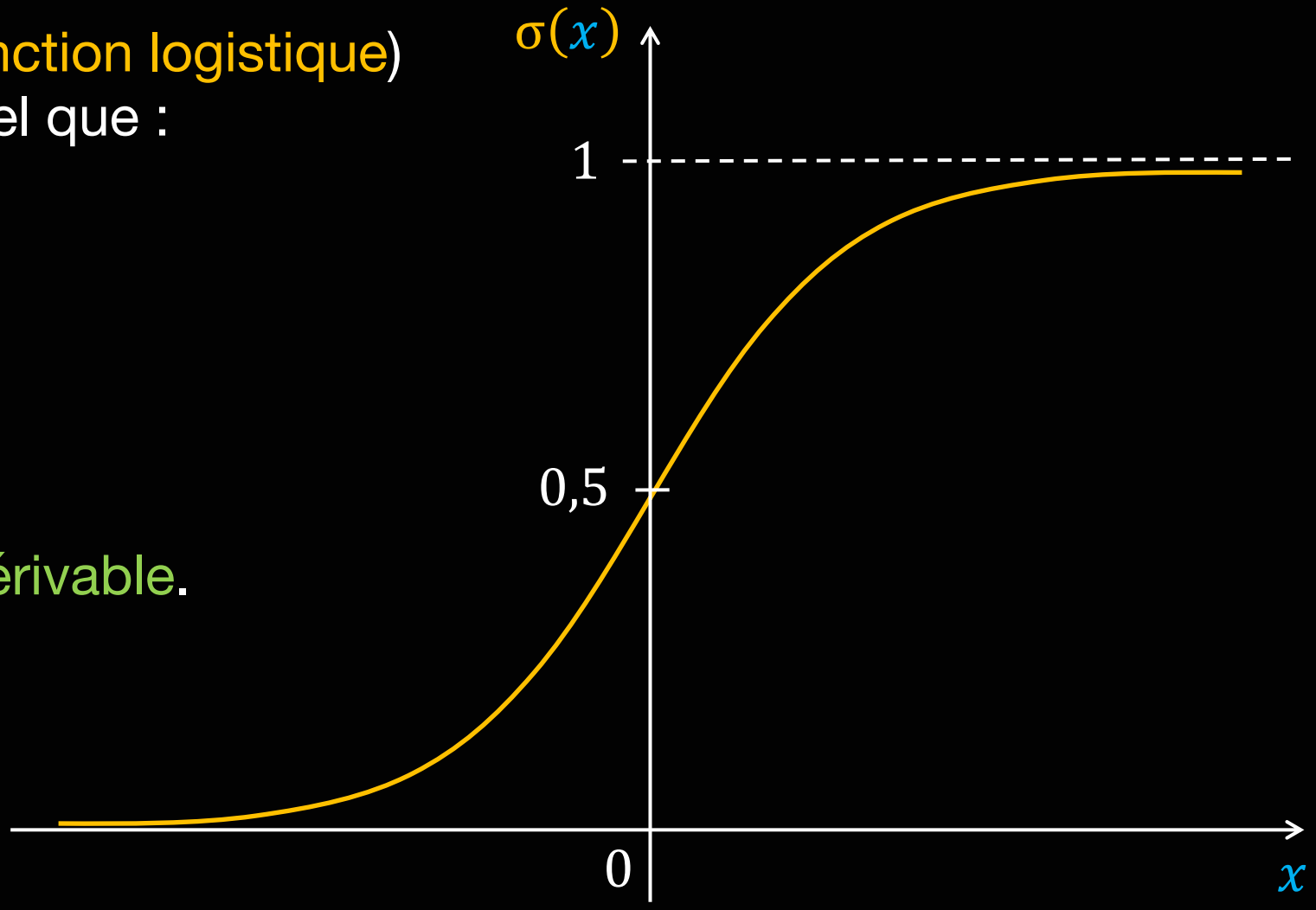
- Mais les perceptrons multicouches ne sont PAS ENTRAINABLES.
- Il est impossible de mettre à jour les poids des couches cachées en connaissant l'erreur en sortie $\mathrm{E} = y_3 - \hat{y}$ à cause de la fonction d'activation de Heaviside.
- Pour entrainer des perceptrons multicouches, on cherche à utiliser la notion de dérivée, or la fonction de Heaviside n'est pas dérivable en 0.
- On a donc trouvé une nouvelle fonction d'activation : la sigmoïde.

La sigmoïde :

- Une sigmoïde (ou fonction logistique) est la fonction $\sigma(x)$ tel que :

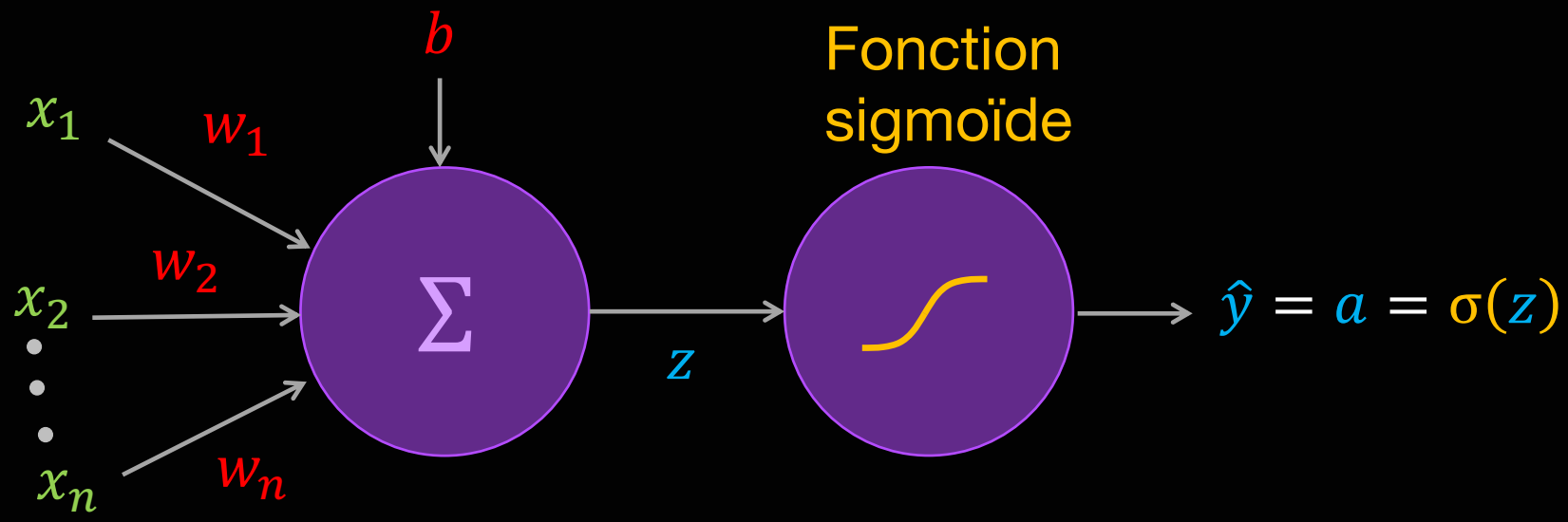
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Cette fonction est dérivable.



La sigmoïde :

- Le neurone est donc :



- La sortie $\hat{y} = a$ est donnée par :

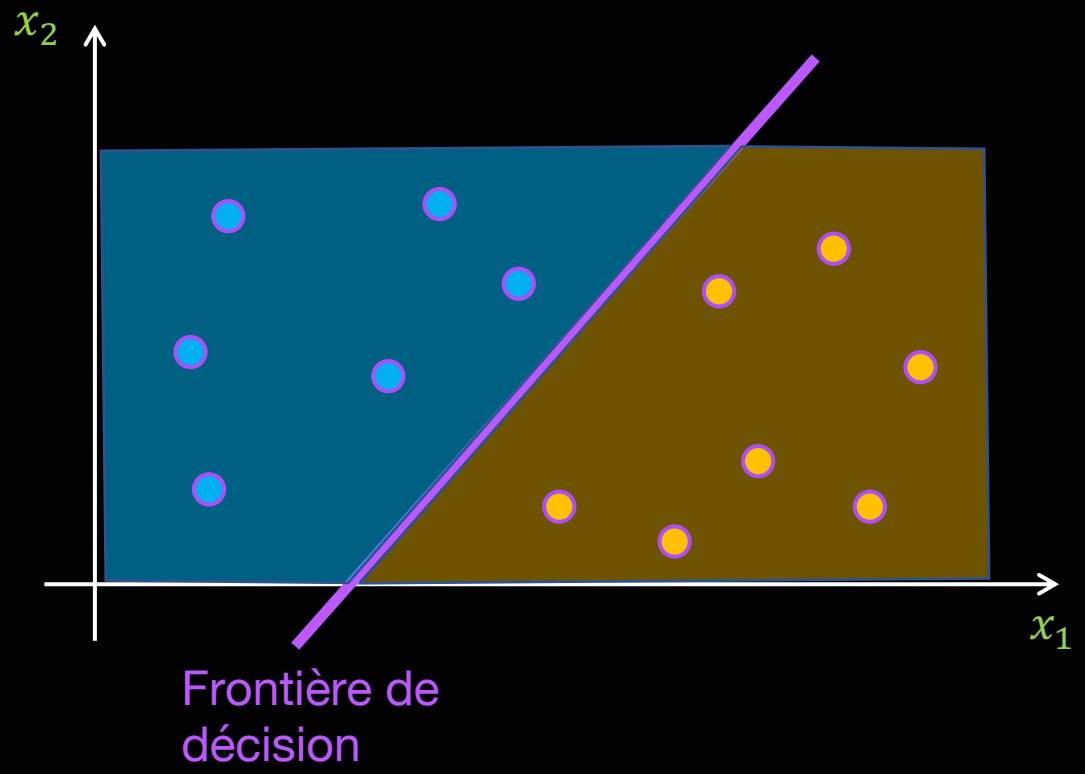
$$\hat{y} = a = \sigma(z) = \frac{1}{1 + e^{-z}} \quad avec\ z = \sum_{i=1}^{n} w_i x_i + b$$

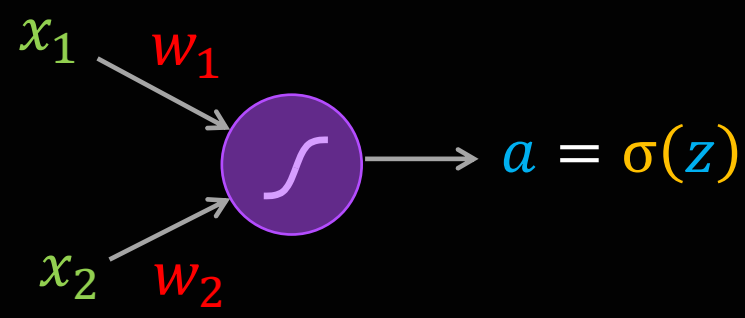
LE NEURONE

La sigmoïde :

- L'avantage de la fonction Sigmoïde est qu'elle fournit une probabilité, donc une valeur entre 0 et 1 : elle est donc très utilisée pour les classifications binaires.

- Exemple de 2 classes points bleu et points orange :





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

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

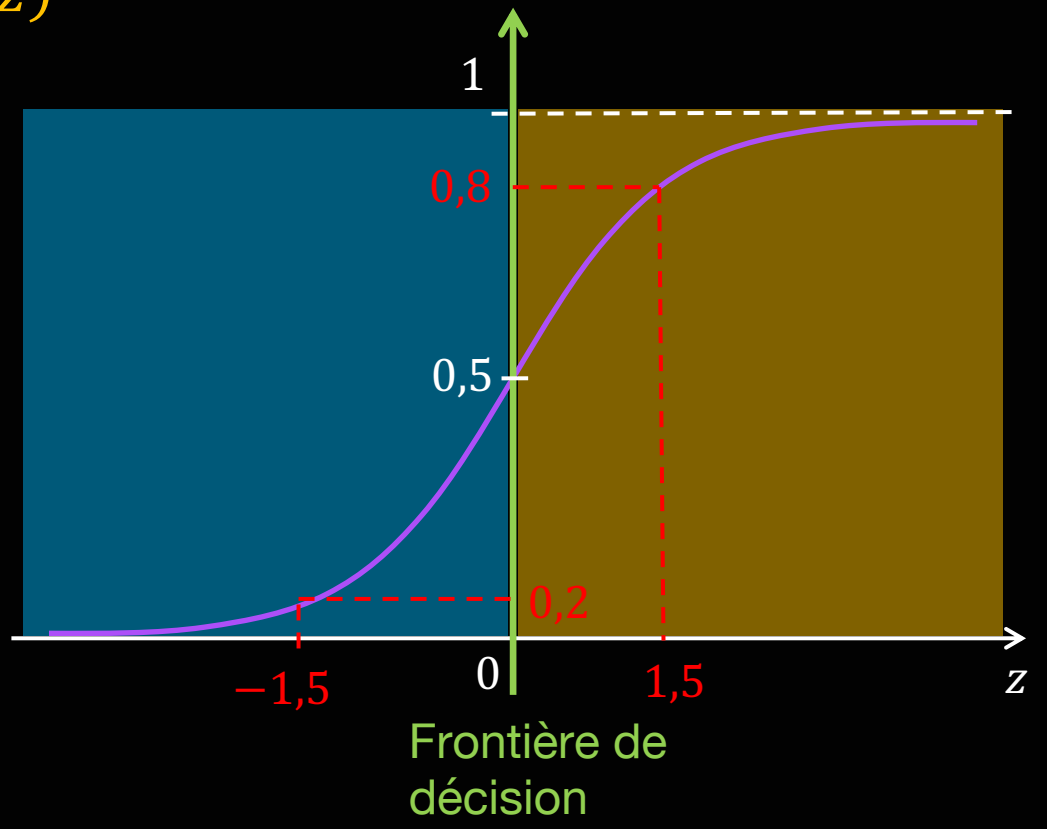
Notion de probabilité :

- Plus un point orange est éloigné de la frontière de décision, plus il est probable qu'il fait bien parti de cette classe.
  La probabilité pour la classe est : $P(z) = a(z)$

$$z = 1{,}5 \Longrightarrow P = 0{,}8 = 80\%$$
$$z = -1{,}5 \Longrightarrow P = 0{,}2 = 20\%$$

- Plus z est grand, plus il est probable que le point appartienne à la classe.

  Plus z est petit, moins il est probable que le point appartienne à la classe.



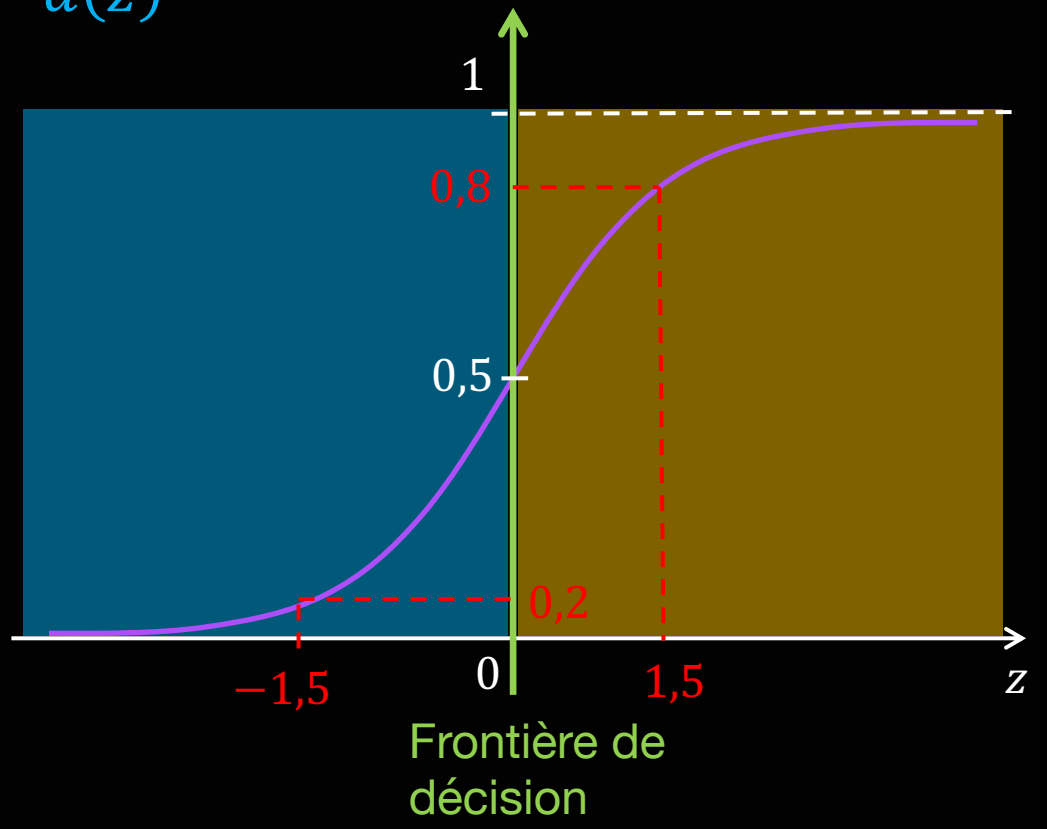
Notion de probabilité :

- Plus un point bleu est éloigné de la frontière de décision, plus il est probable qu'il fait bien parti de cette classe.
  La probabilité pour la classe est : $P(z) = 1 - a(z)$

$$z = 1{,}5 \Longrightarrow P = 1 - 0{,}8 = 20\%$$
$$z = -1{,}5 \Longrightarrow P = 1 - 0{,}2 = 80\%$$

- Plus z est grand, moins il est probable que le point appartienne à la classe.

  Plus z est petit, plus il est probable que le point appartienne à la classe.



Notion de probabilité (cas de la classification) :

- Si on pose :
  - Pour la classe bleu $Y = 0$
  - Pour la classe orange $Y = 1$

- On peut résumer les deux propriétés précédentes sur les probabilités par la loi de Bernoulli :

$$P(Y = y) = a(z)^{y}.\left(1 - a(z)\right)^{1-y}$$

- Pour $Y = 0$, on a : $P(Y = 0) = a(z)^{0}.\left(1 - a(z)\right)^{1-0} = 1 - a(z)$

- Pour $Y = 1$, on a : $P(Y = 1) = a(z)^{1}.\left(1 - a(z)\right)^{1-1} = a(z)$

Vraisemblance :

- Les neurones sont entrainables.
- On peut mettre à jour les poids en connaissant et en cherchant à maximiser la vraisemblance L (Likelyhood) du modèle :

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

$$avec \begin{cases} m : nombre\ de\ données \\ y_i : sortie\ n°i\ attendue \\ a_i : sortie\ n°i\ du\ neurone \end{cases}$$

- Plus la vraisemblance L tend vers 100%, meilleur est le modèle.

Fonction coût :

- Mais la vraisemblance L est difficile à calculer et tends vers 0 (produit de nombre compris entre 0 et 1).

- On utilise alors, pour régler les poids w_i et le biais b, une fonction coût $\mathcal{L}$ (Loss function ou Log Loss) que l'on cherchera à minimiser :

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

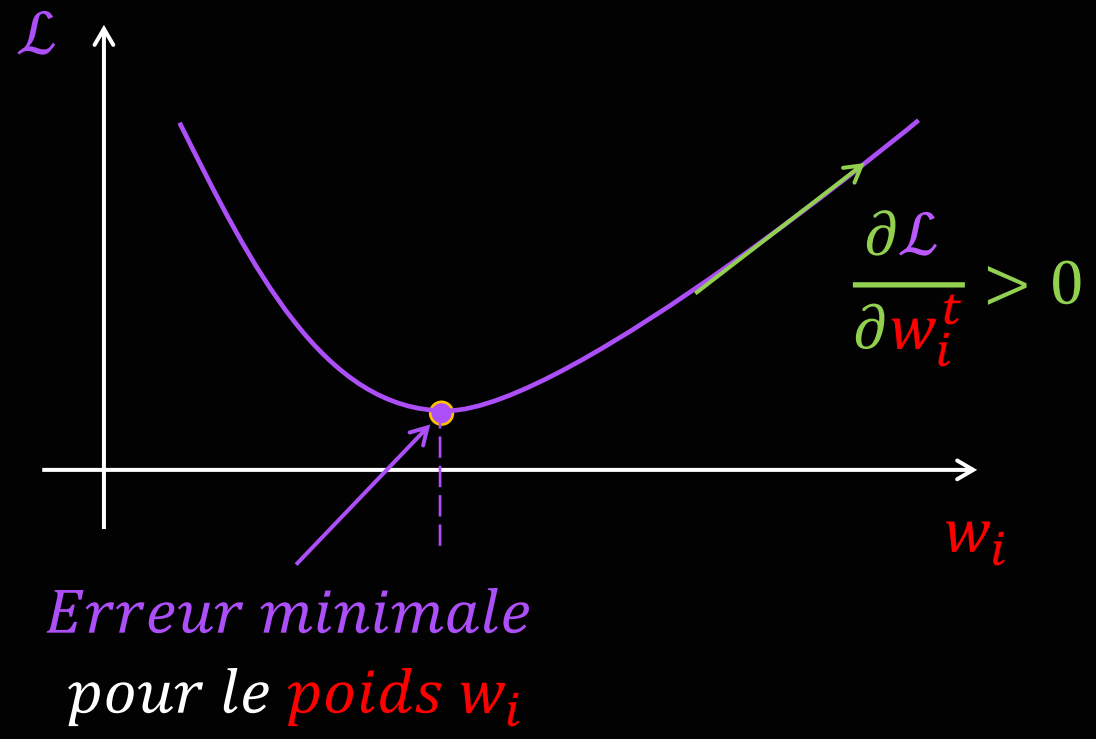
$$avec \begin{cases} m : nombre\ de\ données \\ y_i : sortie\ n°i\ attendue \\ a_i : sortie\ n°i\ du\ neurone \end{cases}$$

- Le logarithme $log(x)$ est le logarithme naturel, c'est-à-dire $ln(x)$ en France.

Descente de gradient :

- On trouve alors l'algorithme de la descente de gradient.

- Le but est de calculer les poids w_i et le biais b pour minimiser la fonction coût $\mathcal{L}$: on utilise pour cela la notion de dérivée (ou de gradient).

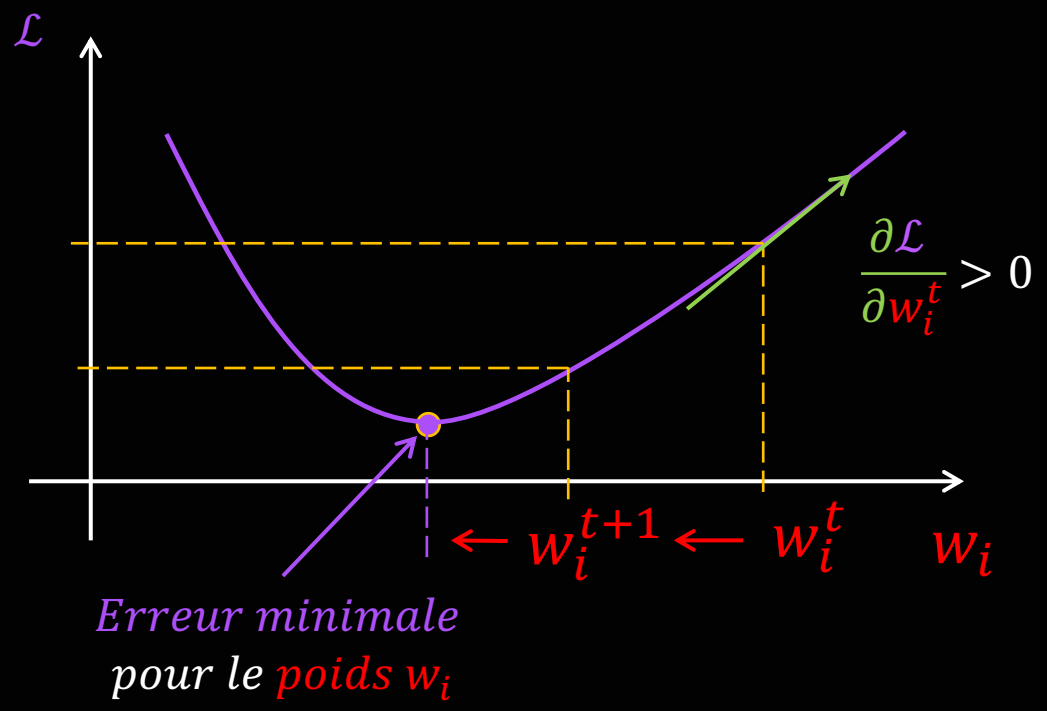


Descente de gradient :

- Algorithme de la descente de gradient :

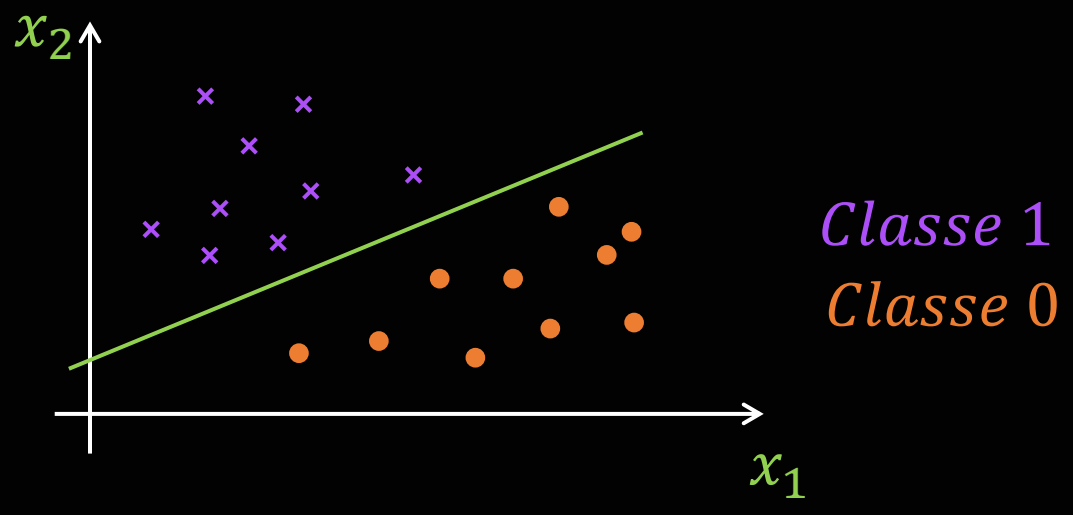
$$w_i^{t+1} = w_i^t - \alpha \frac{\partial \mathcal{L}}{\partial w_i^t}$$

avec
- w_i^{t+1} : poids final (instant $t+1$)
- w_i^t : poids initial (instant t)
- $\frac{\partial \mathcal{L}}{\partial w_i^t}$: dérivée partielle $\mathcal{L}$ par rapport à w_i^t
- α : pas d'apprentissage (> 0)

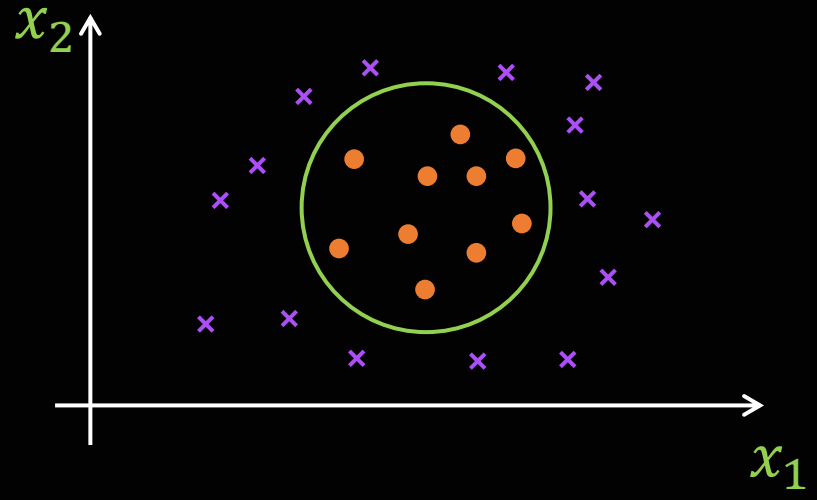


Définitions :

- Un neurone artificiel n'est capable que de séparer linéairement deux classes de points.



Classification binaire linéaire



Classification binaire non linéaire

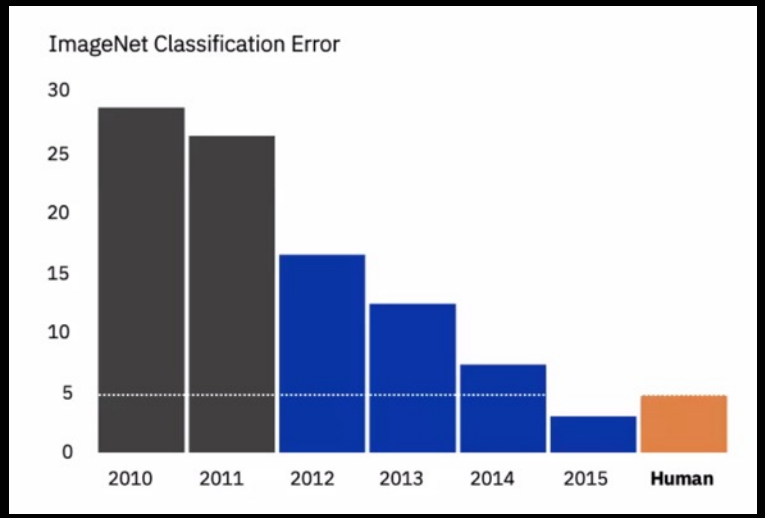
- Pour résoudre des problèmes non linéaires, on utilise un réseau de neurones.

Le Deep Learning

Apogée du Deep Learning :

- L'apogée du Deep Learning arrive dans les années 2010 grâce à la conjonction :
    - Du nombre de données (big data)
    - Des algorithmes
    - De la puissance de calcul

- Les taux d'erreurs de reconnaissance du concours ImageNet, grâce à cette trilogie à diminuer jusqu'à battre les humains.



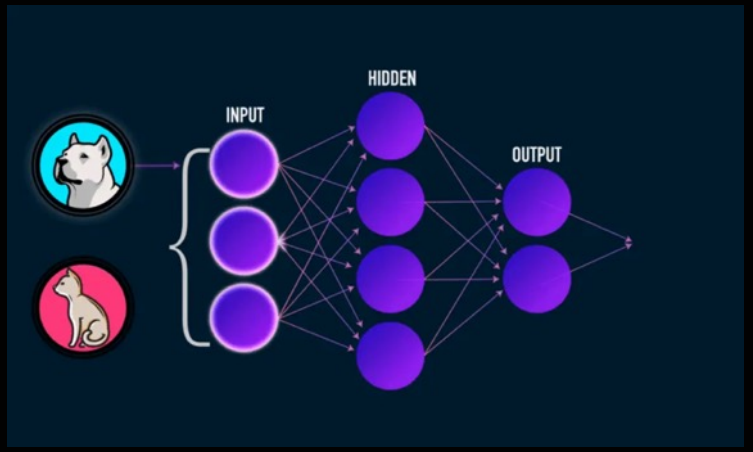
LE DEEP LEARNING

Définition :

- Le Deep Learning (apprentissage profond) est une technique de Machine Learning supervisé reposant sur le modèle des réseaux neurones artificiels qui s’inspire du cerveau humain (100 milliards de neurones).

- Un réseau de neurone a besoin d'un grand nombre de données pour apprendre : c'est le big data qui a permis son développement.

- L'ancêtre du Deep Leaning est le perceptron.



Applications :

- Pour pouvoir entrainer les algorithmes, on a besoin d'un grand nombre de données : utilisation du big data, détenu par les Américains et les Chinois, mais les Européens cherchent à se mettre à niveau (projet VoiceLab, par exemple).

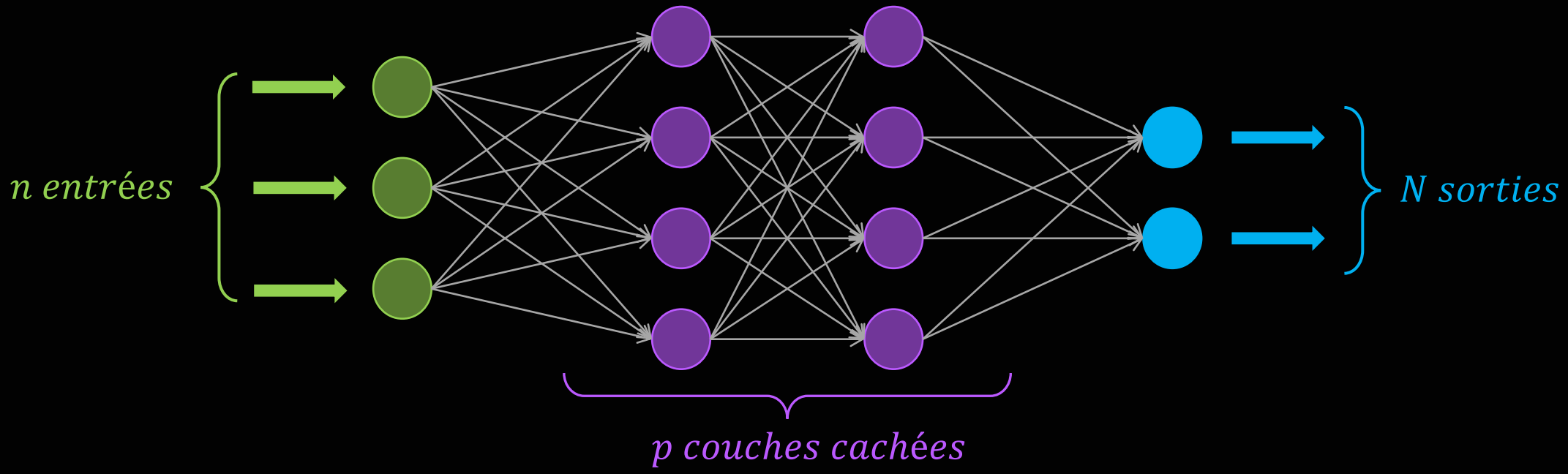
- Les applications prennent place dans des domaines diverses :
    - Médecine
    - Justice
    - Véhicules autonomes
    - Réseaux sociaux
    - Assistants vocaux
    - Détection des fraudes
    - Détection de visages
    - …

Les différents types de réseaux de neurones :

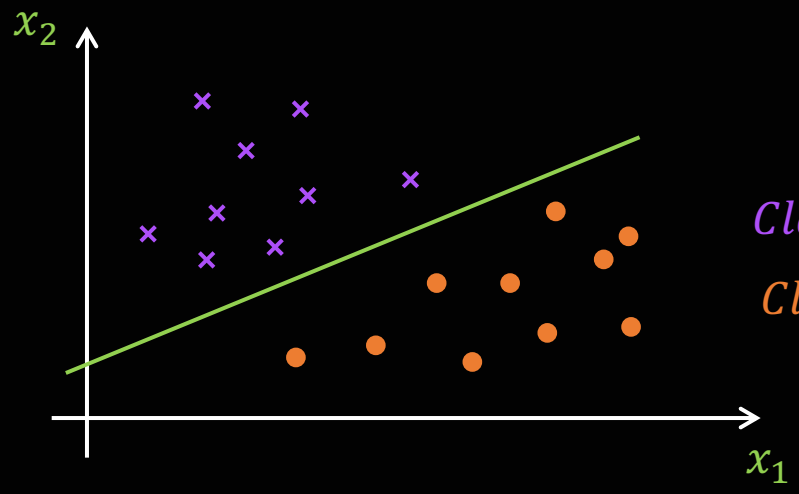
- Il existe différents types de réseaux de neurones suivant l'utilisation que l'on veut en faire :
    - Réseaux à Convolution (CNN) : réseau de neurones permettant l'analyse d'images (détection des visages, analyse médicale, …).
    - Réseaux Récurrents (RNN) : réseau de neurones permettant la reconnaissance des textes et vocale qui prennent en compte le temps (analyse des sentiments, analyse de la parole, …).
    - Réseaux Antagonistes Génératifs (GANs) : réseau de neurones permettant la génération d'images réalistes.
    - …

Architecture d'un réseau de neurones :

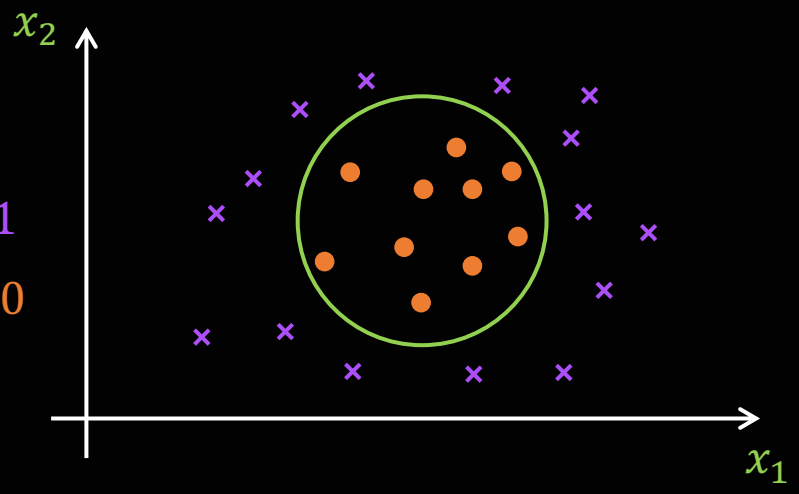
- Un réseau de neurones est défini par des hyperparamètres.
  Il est composé :
  - D'une couche d'entrée (input layer)
  - De couches cachées (hidden layers)
  - D'une couche de sortie (output layer)



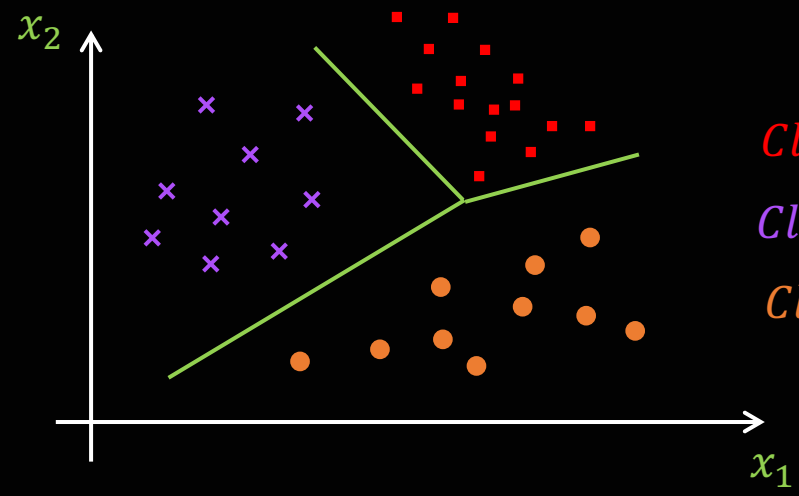
Types de classification :



Classification binaire linéaire



Classification binaire non linéaire



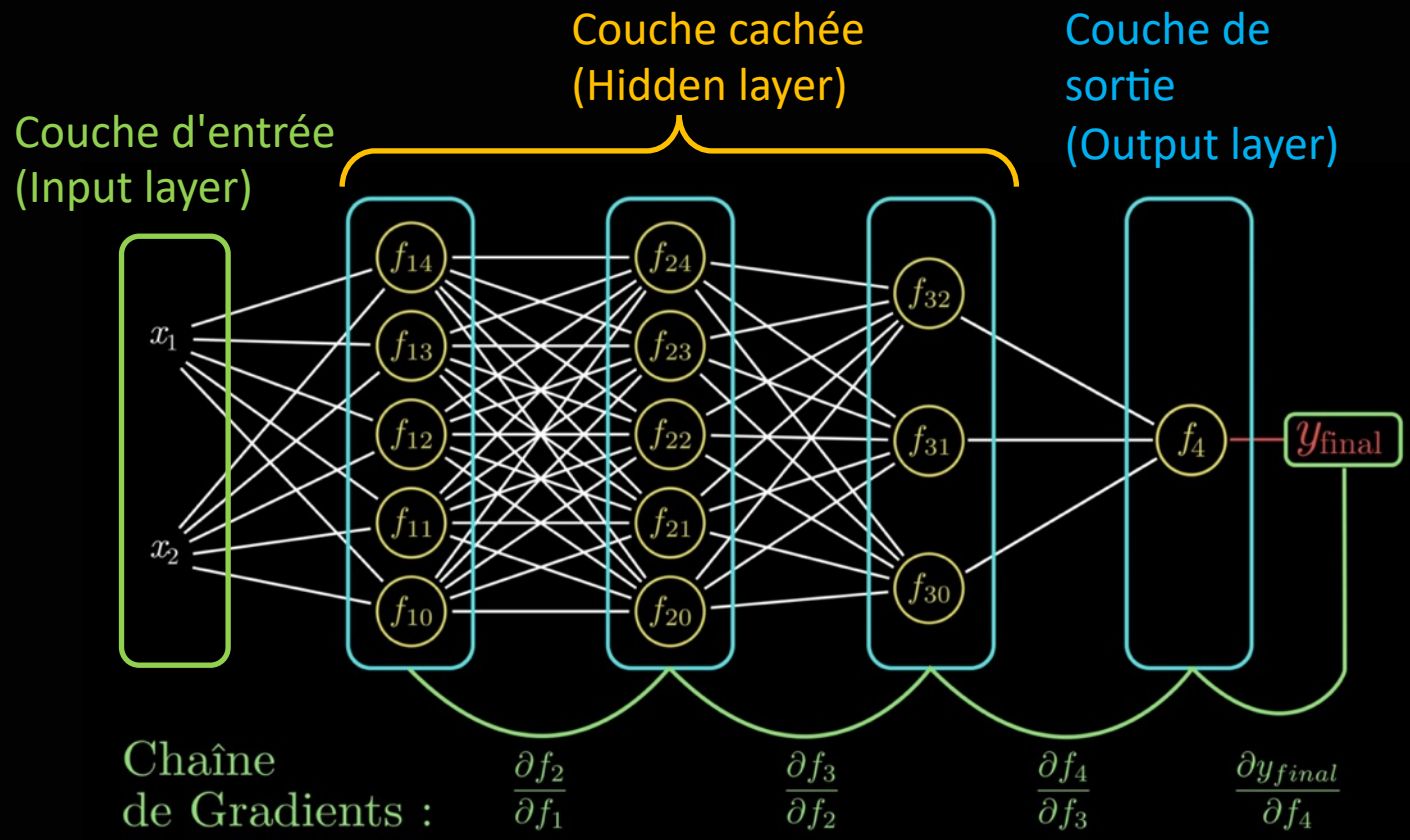
Classification multiclasse

Etapes du Deep Learning :

- Le processus d'apprentissage d'un réseau de neurone peut être décomposé en trois étapes :
    - L'entraînement : Ajustement des paramètres du modèle, choisis au départ aléatoirement, pour arriver à des prédictions de plus en plus précises (Train Set).
    - La validation : Vérification de la performance et de la possibilité de généraliser le modèle avec un retour sur l'entrainement s'il le faut (Validation Set).
    - Le testing : Test final du modèle à l'aide de données réelles (Test Set).

Définitions :

- Plutôt que de lui indiquer un algorithme d'apprentissage comme en Machine Learning, on laisse le réseau de neurones décider lui en allouant des neurones.



Entrainement des réseaux de neurones artificiels :

- Méthode pour développer un réseau de neurones :
  1. Forward Propagation (Data propagation)
  2. Loss Function (Calcul de l'erreur)
  3. Backward Propagation (Gradient Descent : Error propagation)