Python

Introduction
Les tests, les listes, les tupples, les tableaux,….
Le lien avec des fichiers de données
Les IHM avec Tkinter
Machine learning

Python

IA & Machine learning

Aujourd’hui ?


18:42
38%
person 90.5
ultralytics
HUB BETA
bottle 62.0
bottle 27.5
cup 77.7
dining table 44.8
16.5 FPS
43.0 ms
cell phone 56.1
Predict

Python

IA & Machine learning

Aujourd’hui ?

https://www.youtube.com/watch?v=FhVWKttvx6E

Motivations

- Les systèmes d’assistance : l’IA générative va venir augmenter l’efficacité des tâches pratiques, telles que la programmation ou la maintenance des machines, en générant directement le code nécessaire pour développer des solutions d’automatisation.

[source : article de bcg.com]

Motivations

- Les systèmes d’assistance : l’IA générative va venir augmenter l’efficacité des tâches pratiques, telles que la programmation ou la maintenance des machines, en générant directement le code nécessaire pour développer des solutions d’automatisation.
- Les systèmes de recommandation : l’IA générative va fournir des recommandations pour aider les opérateurs à identifier les meilleures méthodes. Elle peut notamment améliorer la maintenance prédictive en créant automatiquement des instructions étape par étape, y compris des listes de pièces de rechange requises.

[source : article de bcg.com]

Motivations

- Les systèmes d’assistance : l’IA générative va venir augmenter l’efficacité des tâches pratiques, telles que la programmation ou la maintenance des machines, en générant directement le code nécessaire pour développer des solutions d’automatisation.
- Les systèmes de recommandation : l’IA générative va fournir des recommandations pour aider les opérateurs à identifier les meilleures méthodes. Elle peut notamment améliorer la maintenance prédictive en créant automatiquement des instructions étape par étape, y compris des listes de pièces de rechange requises.
- Les systèmes autonomes : l’IA générative va permettre de s’adapter à des situations inhabituelles et de favoriser l’auto-régulation. Par exemple, lors du déploiement autonome de robots, l’IA générative permettra aux robots multimodaux de traduire les prompts des opérateurs en une séquence d’actions que le système exécutera ensuite pour effectuer les tâches de manutention.

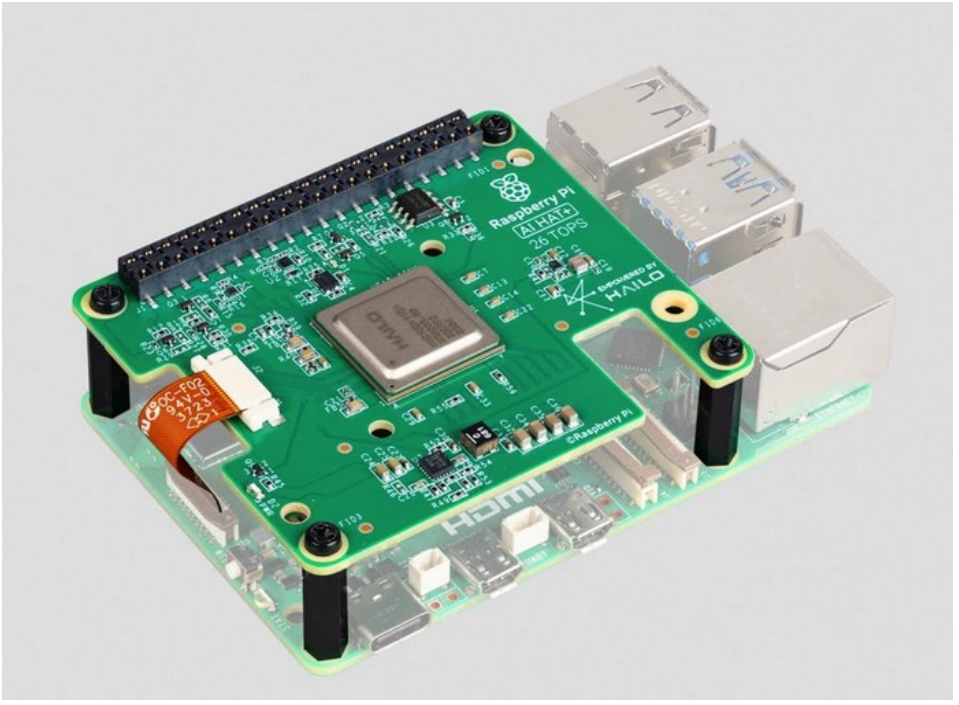
[source : article de bcg.com]

IA & Motivations

IA RPi5

NPU : Neural Processing Unit

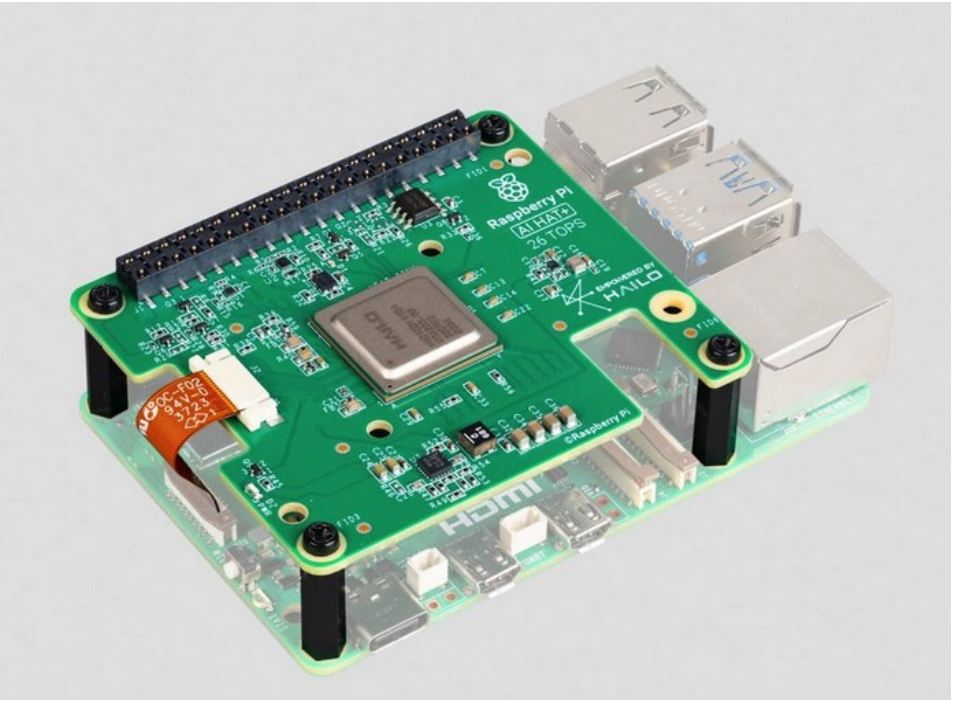
Matlab & PLC

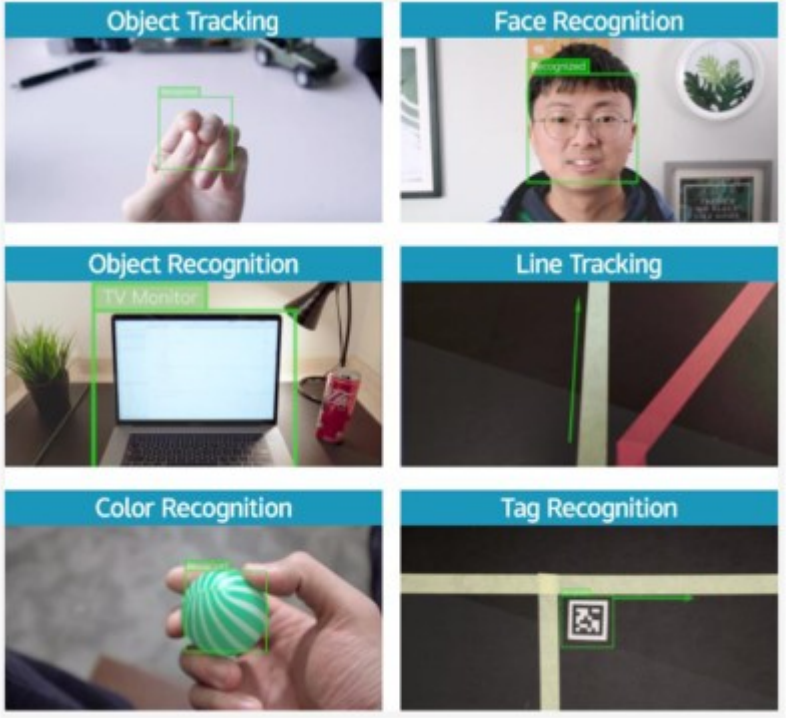


IA & Motivations

IA RPi5

ssing Unit

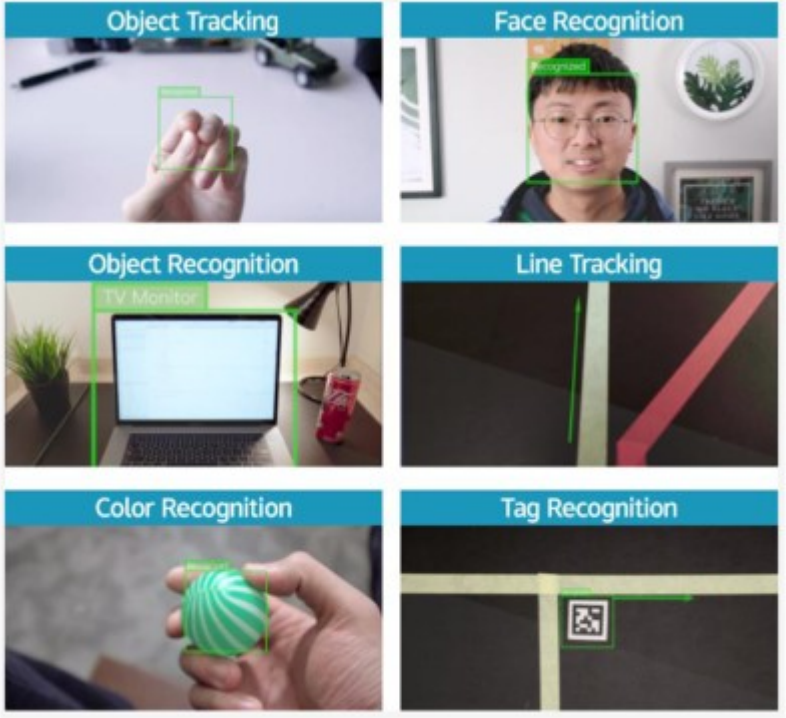




Caméra Huskylens

IA &

IA RPi5



Pourquoi avons-nous choisi la caméra Huskylens ?

Au début du projet, nous avons utilisé OpenCV pour détecter les déchets en fonction de leur forme et de leur couleur. Cependant, cela s'est avéré difficile, notamment pour distinguer les capsules de bière des bouchons. Nous avons alors décidé d'explorer l'intelligence artificielle. Nous avons d'abord testé YOLO, une technologie d'apprentissage profond, qui s'est révélée efficace mais difficile à utiliser en temps réel.

Nous nous sommes ensuite tournés vers Huskylens. Cette caméra fonctionne à l'aide d'une bibliothèque que nous avons intégrée à Arduino. Bien que nous ayons rencontré quelques difficultés techniques pour faire fonctionner la bibliothèque sans erreur, nous avons finalement réussi en explorant différentes solutions, notamment l'utilisation de PlatformIO.

Efficacité de Huskylens

Huskylens s'est avérée être un choix judicieux pour notre projet. Son processus d'apprentissage est simple : il suffit de prendre des photos des objets dans l'environnement de travail. La caméra apprend rapidement à reconnaître ces objets, ce qui en fait un outil efficace pour la détection de déchets tels que les capsules, les mégots et les bouchons. Ensuite, nous récupérons l'identifiant (ID) de l'objet appris à partir d'un code Arduino, puis nous affectons cet ID à l'objet correspondant. Enfin, nous envoyons l'objet détecté à un topic ROS.

54€

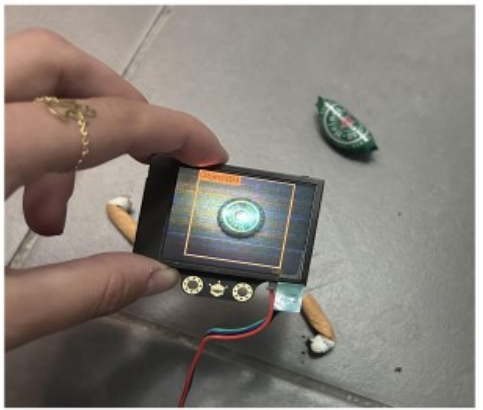


Figure 24 : Huskylens pour la détection des capsules de bière et les mégots de cigarettes

Python

Machine learning

Machine Learning

Le machine Learning, ou encore l’apprentissage automatique en français, fait partie de l’une des approches de l’intelligence artificielle.

Machine Learning

Le machine Learning, ou encore l’apprentissage automatique en français, fait partie de l’une des approches de l’intelligence artificielle.
Le machine Learning est donc une discipline scientifique centrée sur le développement, l’analyse et l’implémentation de méthodes automatisables, qui offrent la possibilité à la machine d’évoluer grâce à un processus d’apprentissage.
Il se révèle particulièrement efficace lorsqu’il s’agit d’analyser de larges ensembles de données diverses et évolutives, ce que l’on nomme communément le Big Data.

Machine Learning

Le machine Learning, ou encore l'apprentissage automatique en français, fait partie de l'une des approches de l'intelligence artificielle.
Le machine Learning est donc une discipline scientifique centrée sur le développement, l'analyse et l'implémentation de méthodes automatisables, qui offrent la possibilité à la machine d'évoluer grâce à un processus d'apprentissage.
Il se révèle particulièrement efficace lorsqu'il s'agit d'analyser de larges ensembles de données diverses et évolutives, ce que l'on nomme communément le Big Data.
Le Machine Learning a de nombreuses applications : en immobilier, en maintenance, en sécurité, ….

Machine Learning

Le machine Learning, ou encore l'apprentissage automatique en français, fait partie de l'une des approches de l'intelligence artificielle.
Le machine Learning est donc une discipline scientifique centrée sur le développement, l'analyse et l'implémentation de méthodes automatisables, qui offrent la possibilité à la machine d'évoluer grâce à un processus d'apprentissage.
Il se révèle particulièrement efficace lorsqu'il s'agit d'analyser de larges ensembles de données diverses et évolutives, ce que l'on nomme communément le Big Data.
Le Machine Learning a de nombreuses applications : en immobilier, en maintenance, en sécurité, ….
Il existe de nombreuses solutions Open Source et le langage Python, avec son principal défaut que de cacher des choses, permet justement de simplifier et de proposer des méthodes toutes faites : c'est l'utilisation de méthode (de sous programmes)

Machine Learning

- Apprentissage supervisé : la prédiction par rapports à des données d'entrée et quelques données de sortie (exemple du prix de l'immobilier / (surface, piscine, garage, climatisation, travaux, proximité des commerces), les espèces de plantes, les maladies,….)
- Apprentissage non supervisé : il n'y a pas de superviseur, pas d'étiquette : le système apprend seul
- Apprentissage par renforcement : basé sur l'expérience. On teste une action : si elle est réussie, on apprend que c'est possible, si elle est un échec, on apprend qu'il y a un obstacle (robotique par exemple)

Machine Learning

- Apprentissage supervisé : la prédiction par rapports à des données d'entrée et quelques données de sortie (exemple du prix de l'immobilier / (surface, piscine, garage, climatisation, travaux, proximité des commerces), les espèces de plantes, les maladies,….)
- Apprentissage non supervisé : il n'y a pas de superviseur, pas d'étiquette : le système apprend seul
- Apprentissage par renforcement : basé sur l'expérience. On teste une action : si elle est réussie, on apprend que c'est possible, si elle est un échec, on apprend qu'il y a un obstacle (robotique par exemple)

Apprentissages supervisés

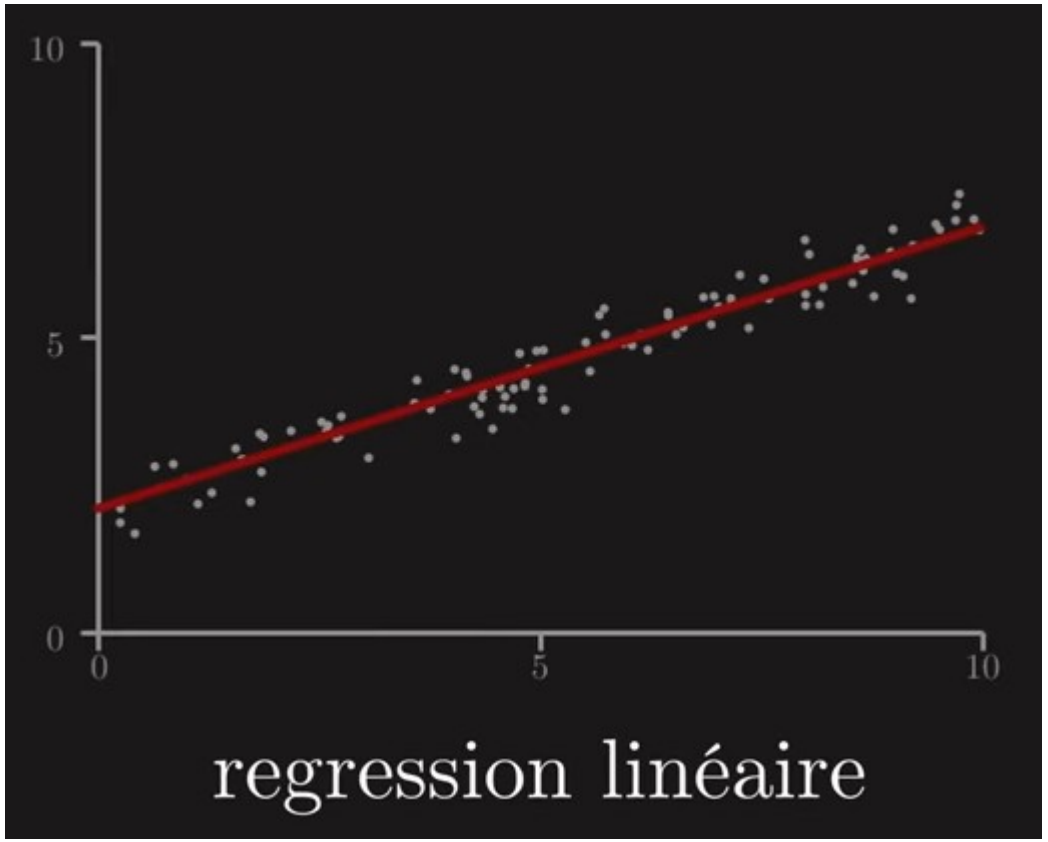
Voici quelques algorithmes d'apprentissage supervisés :

- Arbres de décision
- K Nearest Neighbours (k les plus proches voisins)
- SVM (machine à vecteur de support)
- Naïve Bayes (classification naïve bayésienne)
- Les réseaux de neurones artificiels

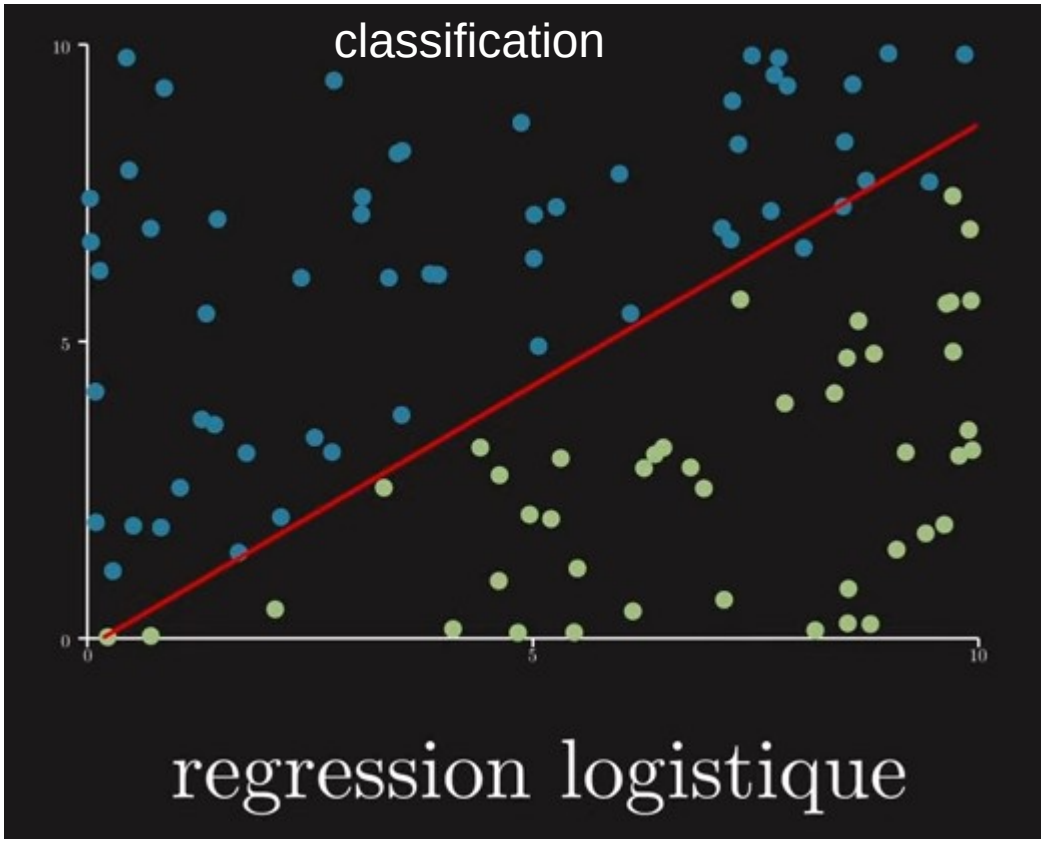
En apprentissage automatique, on distingue :

- La régression linéaire (ou non linéaire) *qui est une prédiction d'une valeur quantitative*
- La régression logistique *qui est une prédiction d'une appartenance à une classe : c'est une classification.*

Apprentissages supervisés



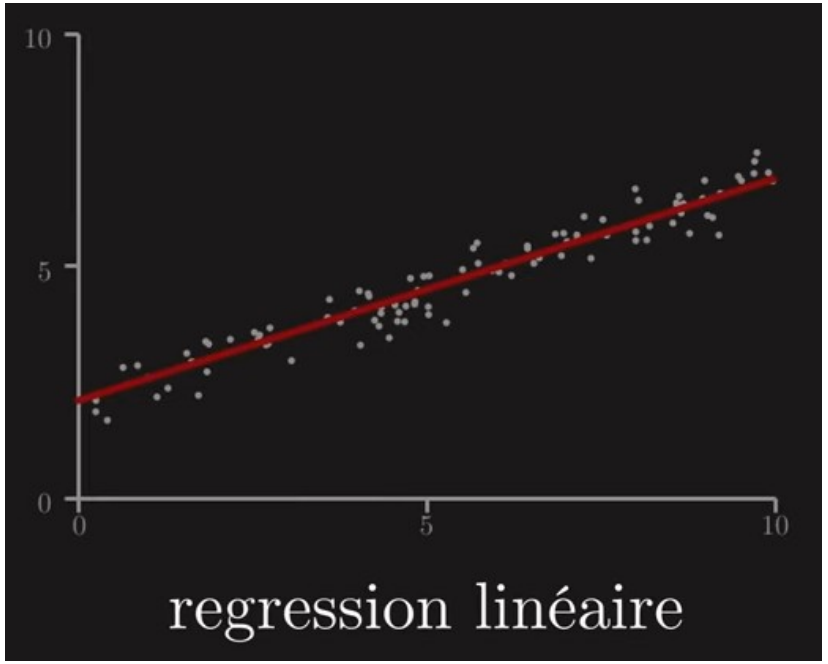
Régression linéaire (droite) ou non linéaire (...)
Exemples : Augmentation de la température de la terre, prix de l'immobilier, consommation d'énergie en fct de la température



Exemples : Reconnaissance des chiens, des chats, des tumeurs

Régression linéaire

Par exemple, basée sur la méthode de l'approximation des moindres carré



y(x) = a.x +b

$$a = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}$$

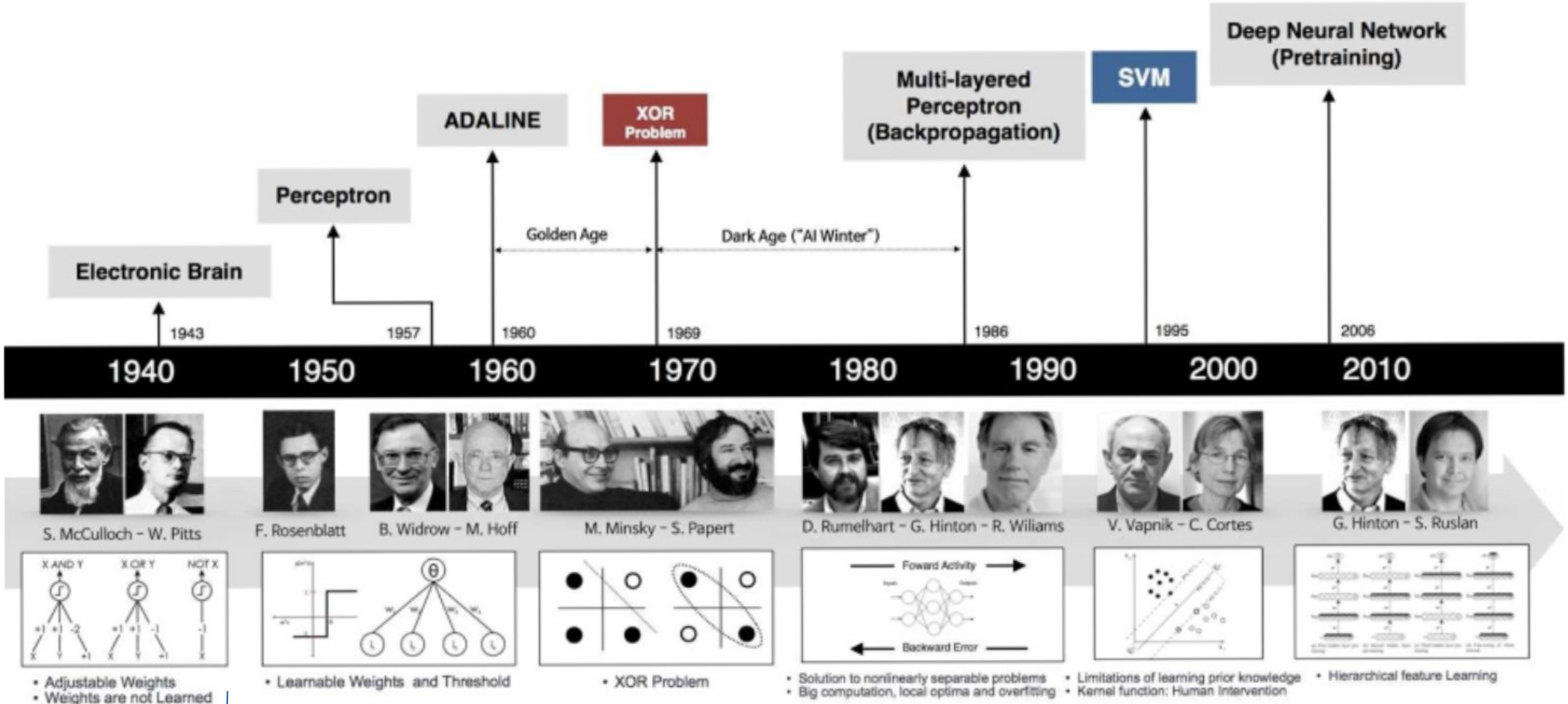
$$b = \overline{y} - a\overline{x}$$

$$r = \frac{\sum \left[(x_i - \overline{x})\,(y_i - \overline{y}) \right]}{\sqrt{\Sigma (x_i - \overline{x})^2 \; * \; \Sigma (y_i - \overline{y})^2}}$$

coefficient de corrélation

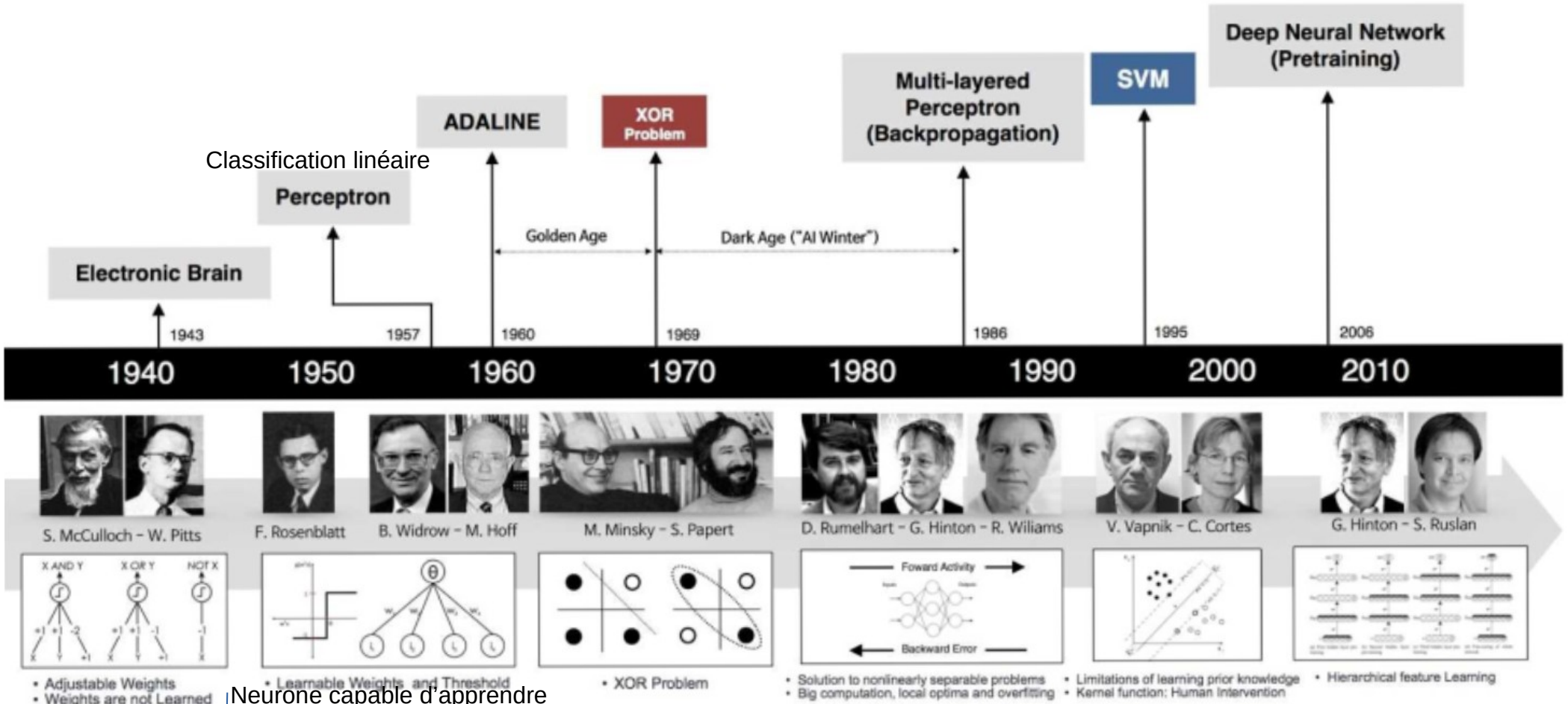
Les réseaux de neurones

Les réseaux de neurones



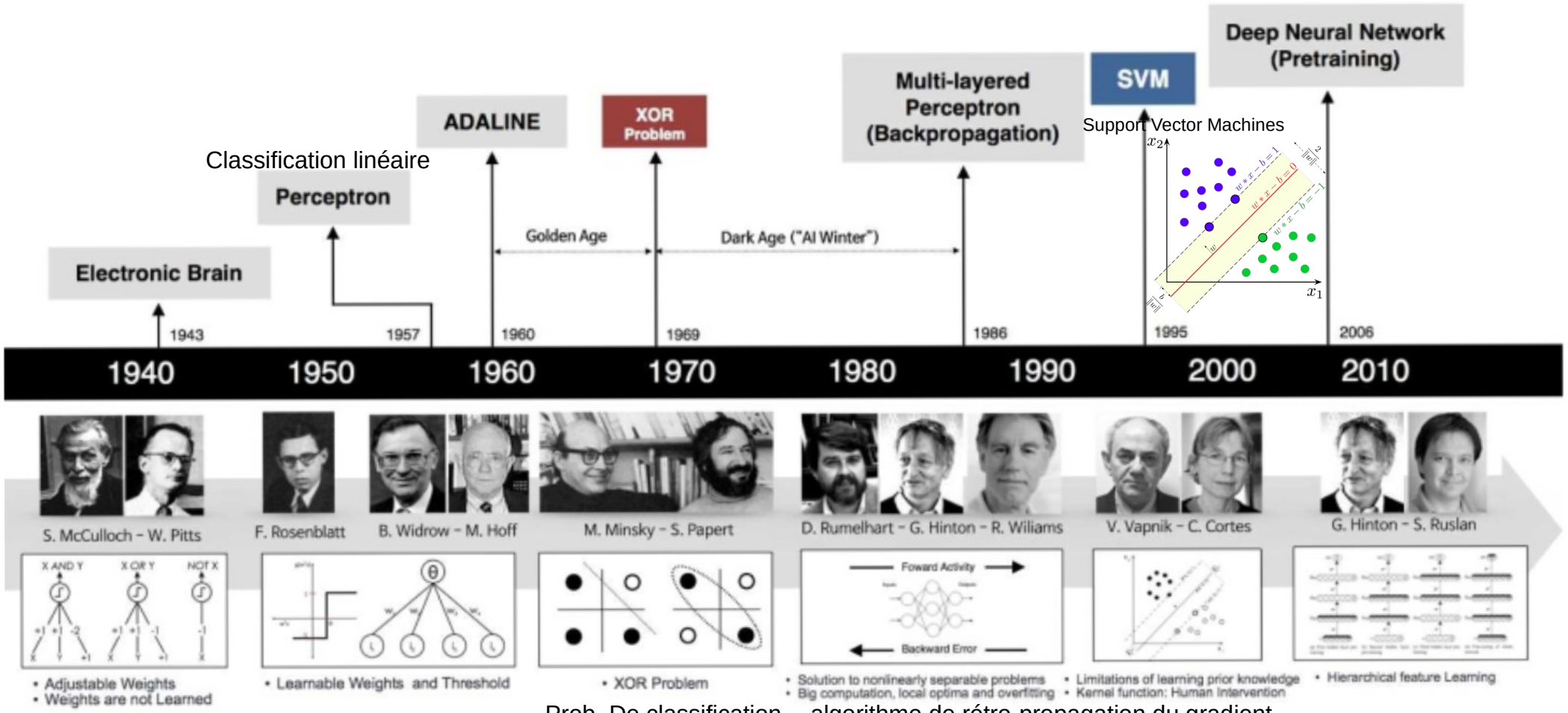
1er neurone artificiel

Les réseaux de neurones



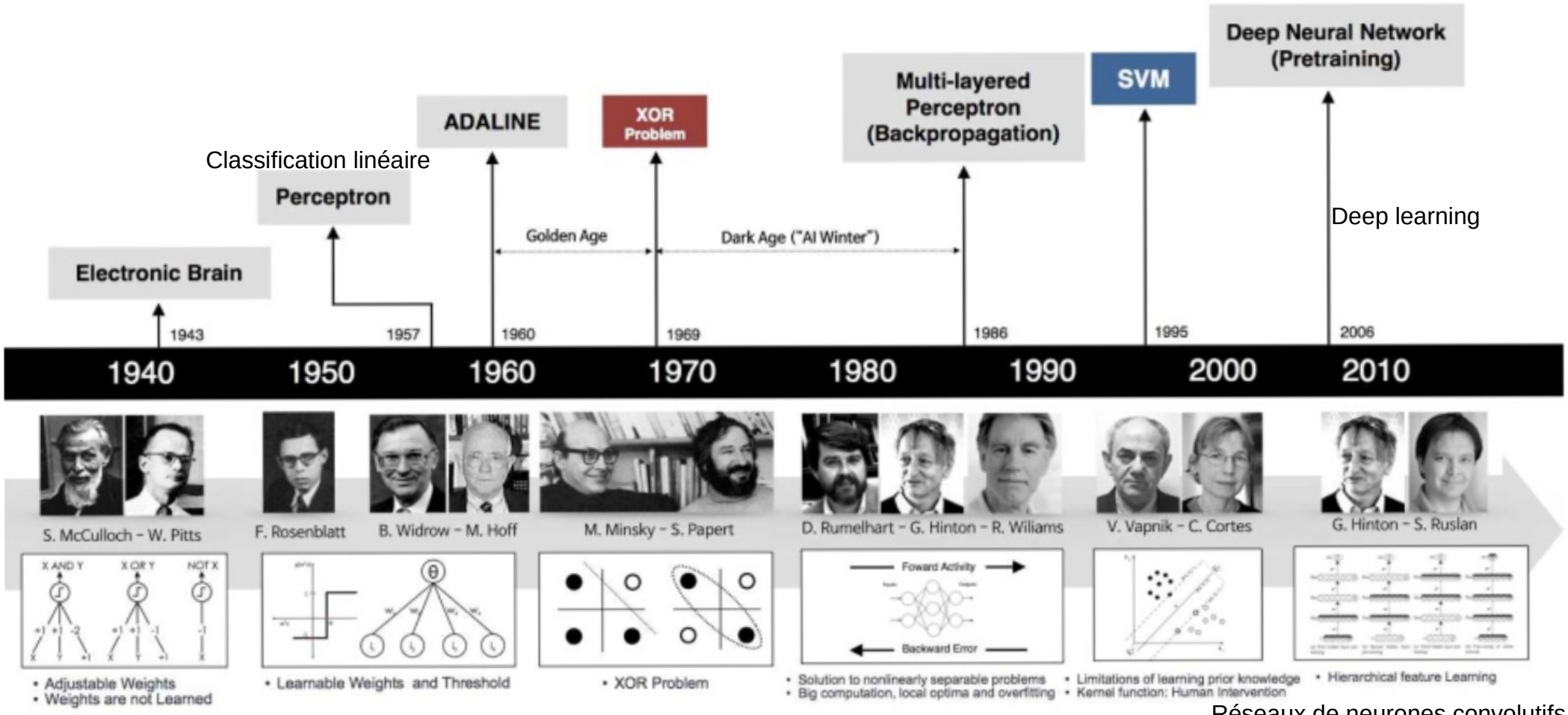
1er neurone artificiel | Neurone capable d'apprendre
Algorithme de la descente du gradient

Les réseaux de neurones



Prob. De classification algorithme de rétro-propagation du gradient

Les réseaux de neurones



Réseaux de neurones convolutifs

Les réseaux de neurones

Principe de prédicteur

Les réseaux de neurones

Principe de prédicteur pour la régression linéaire

Aujourd’hui : $ = 1,5 x €

Les réseaux de neurones

Principe de prédicteur pour la régression linéaire

Aujourd’hui : $ = 1,5 x €.
1ère itération : Demain : on prédit $ = 1,5 x € mais, $ = 1,1 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

Les réseaux de neurones

Principe de prédicteur pour la régression linéaire

Aujourd'hui : $ = 1,5 x €.
1ère itération : Demain : on prédit $ = 1,5 x € mais, $ = 1,1 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

2nd itération : Après-demain : on prédit $ = 1,1 x € mais, $ = 1,2 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

Les réseaux de neurones

Principe de prédicteur pour la régression linéaire

Aujourd'hui : $ = 1,5 x €.
1ère itération : Demain : on prédit $ = 1,5 x € mais, $ = 1,1 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

2nd itération : Après-demain : on prédit $ = 1,1 x € mais, $ = 1,2 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

3ème itération : Après-après-demain : on prédit $ = 1,2 x € mais, $ = 1,5 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

Les réseaux de neurones

Principe de prédicteur pour la régression linéaire

Aujourd'hui : $ = 1,5 x €.
1ère itération : Demain : on prédit $ = 1,5 x € mais, $ = 1,1 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

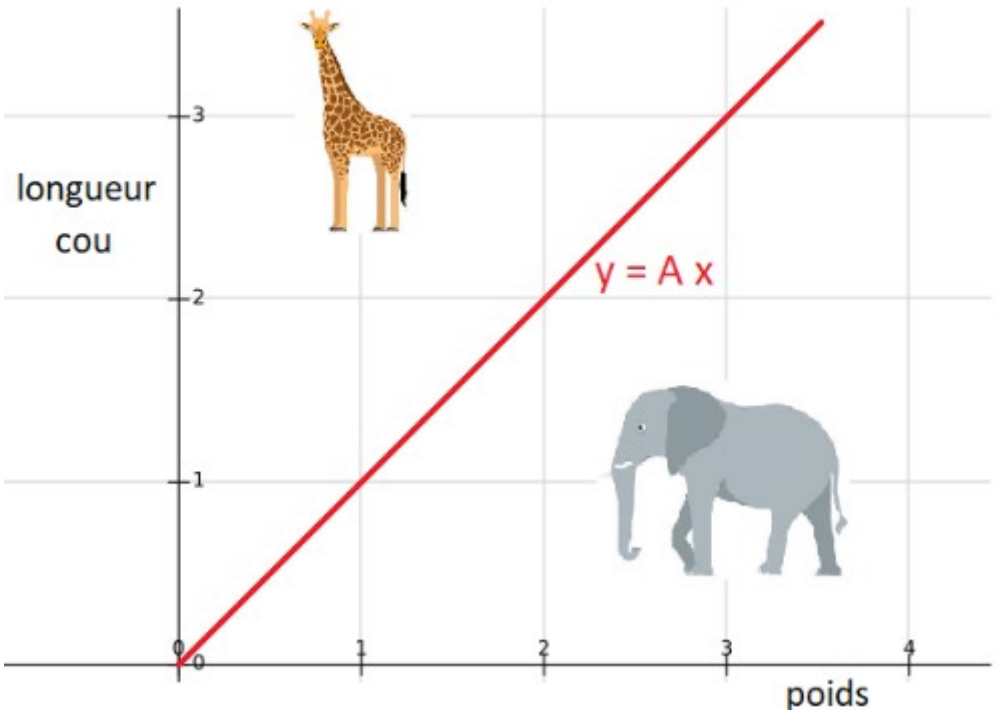
2nd itération : Après-demain : on prédit $ = 1,1 x € mais, $ = 1,2 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

3ème itération : Après-après-demain : on prédit $ = 1,2 x € mais, $ = 1,5 x €
→ le prédicteur est évalué sur base de l'erreur qu'il commet,
→ le prédicteur est corrigé afin d'améliorer son comportement

….

Les réseaux de neurones

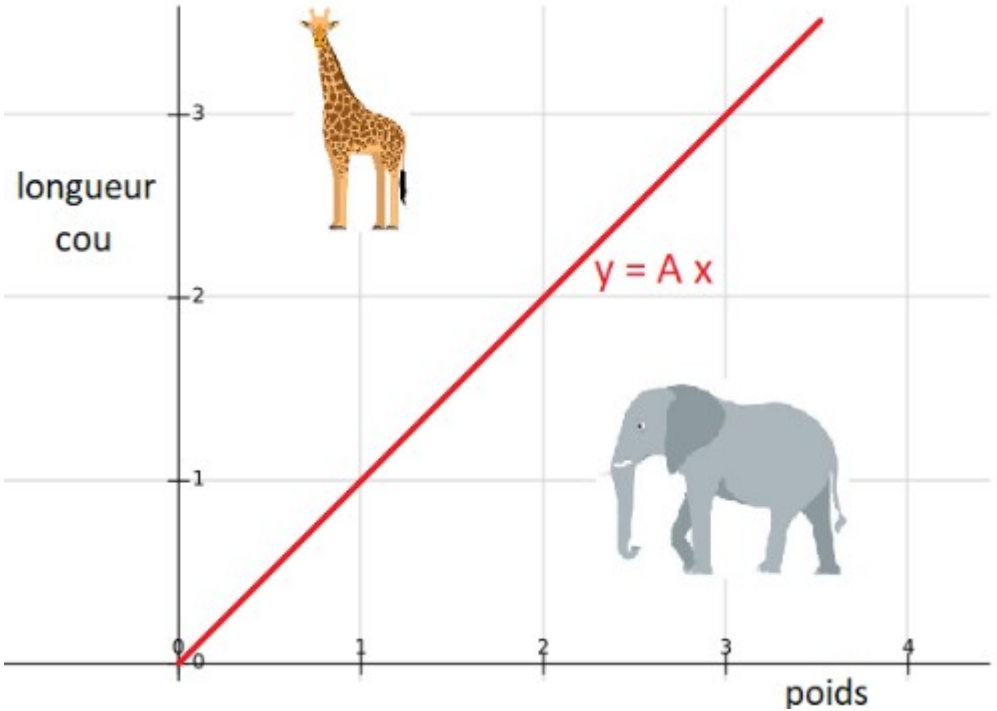
Principe de prédicteur pour la régression logistique



y (longueur du cou) = A . x (poids)

Les réseaux de neurones

Principe de prédicteur pour la régression logistique



$$\text{sortie du classificateur} = \begin{cases} \text{girafe} & \text{si } y_s > A\,x_s \\ \text{éléphant} & \text{si } y_s < A\,x_s \end{cases}$$

= opération de seuillage

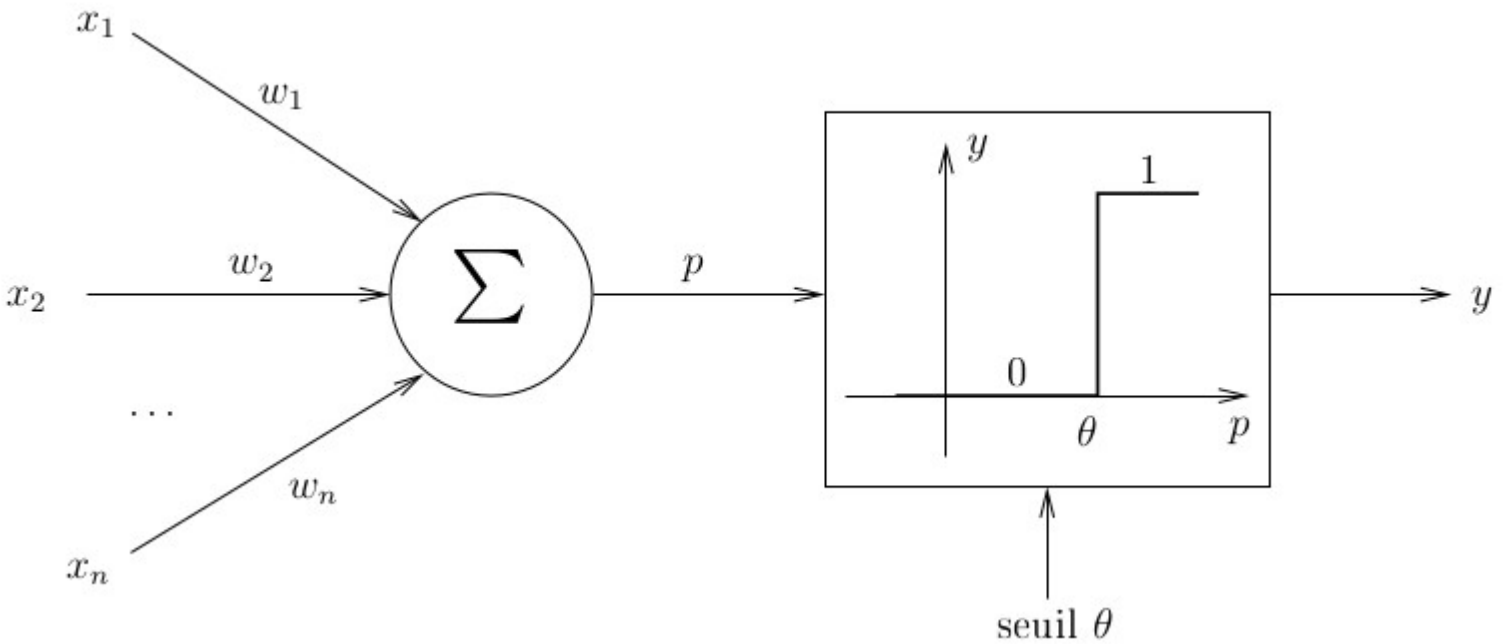
y (longueur du cou) = A . x (poids)

Les réseaux de neurones

Le neurone

Les réseaux de neurones

Le neurone

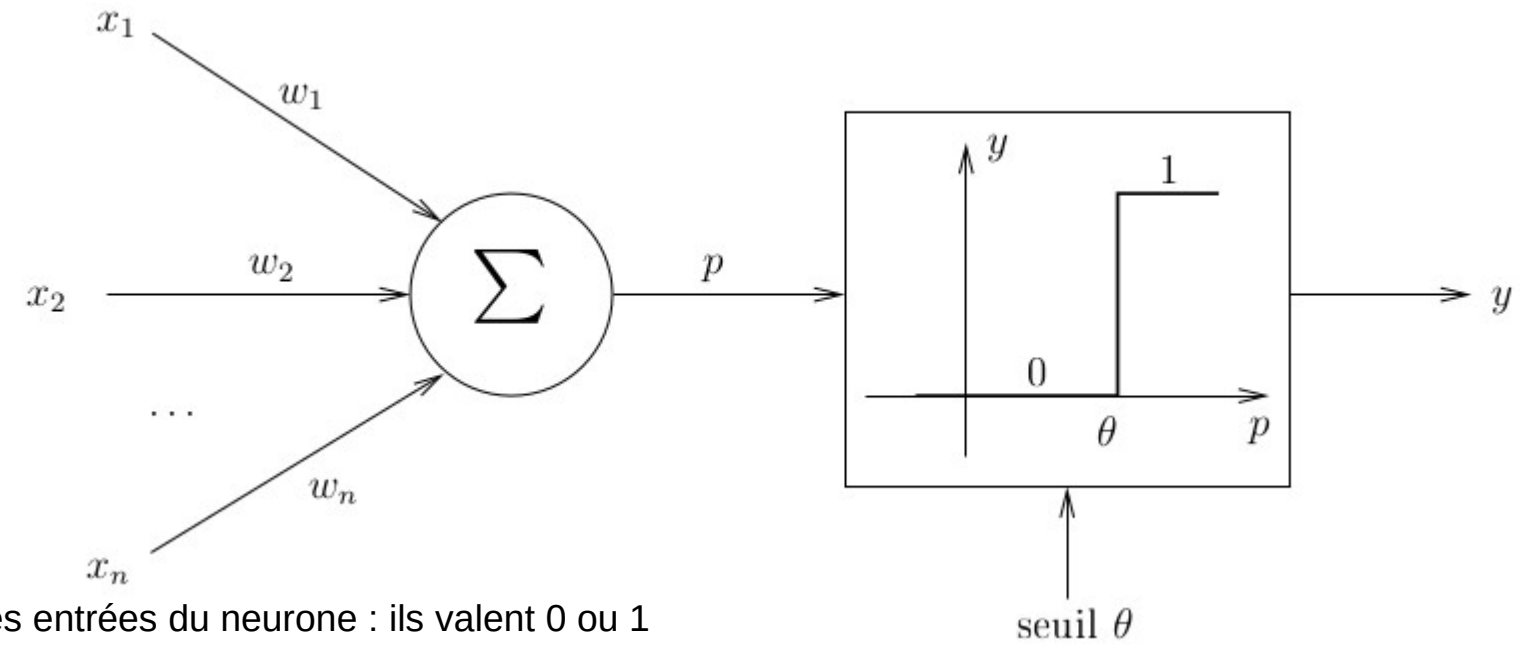


Le neurone calcule la somme pondérée de ses entrées puis il la compare par rapport à un seuil.

- Si la somme pondérée est supérieure à ce seuil, alors le neurone s'active et sort la valeur 1
- Si la somme pondérée est inférieure à ce seuil, alors le neurone se désactive et sort la valeur 0

Les réseaux de neurones

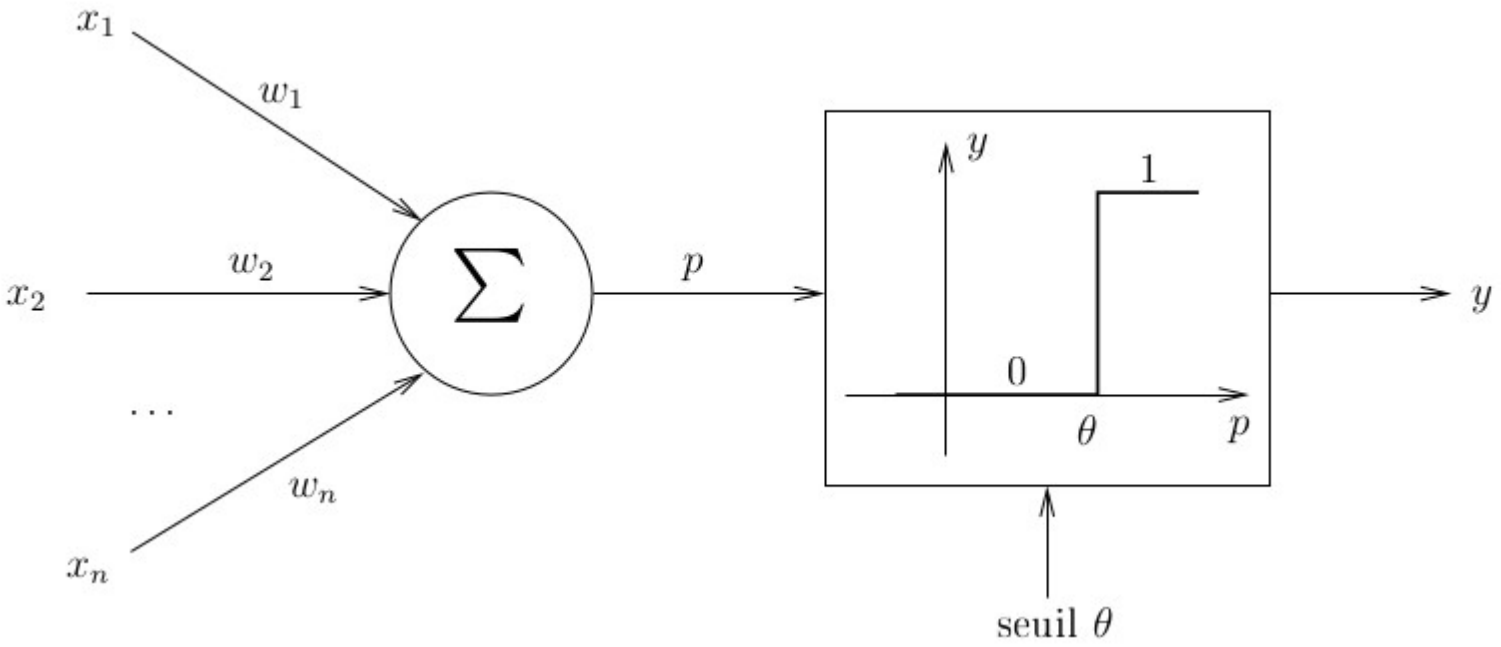
Le neurone



x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

Le neurone

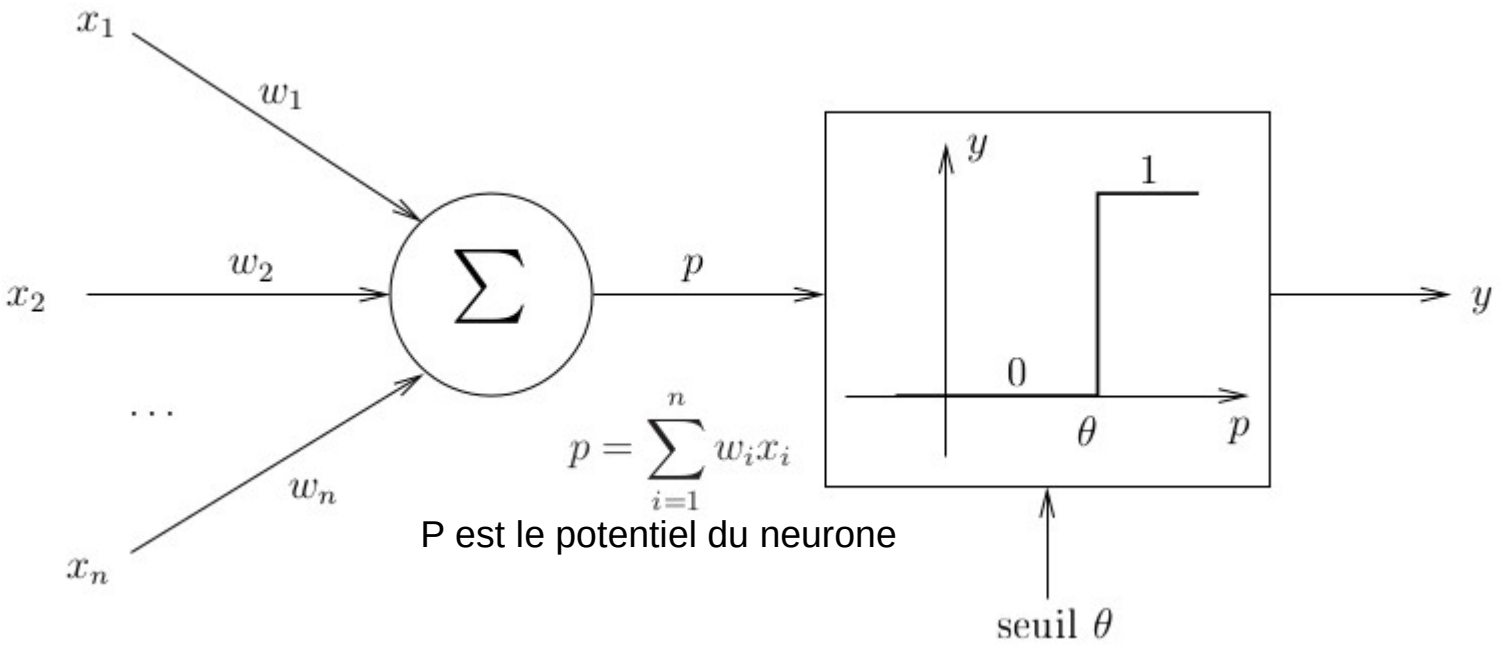


ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

Le neurone



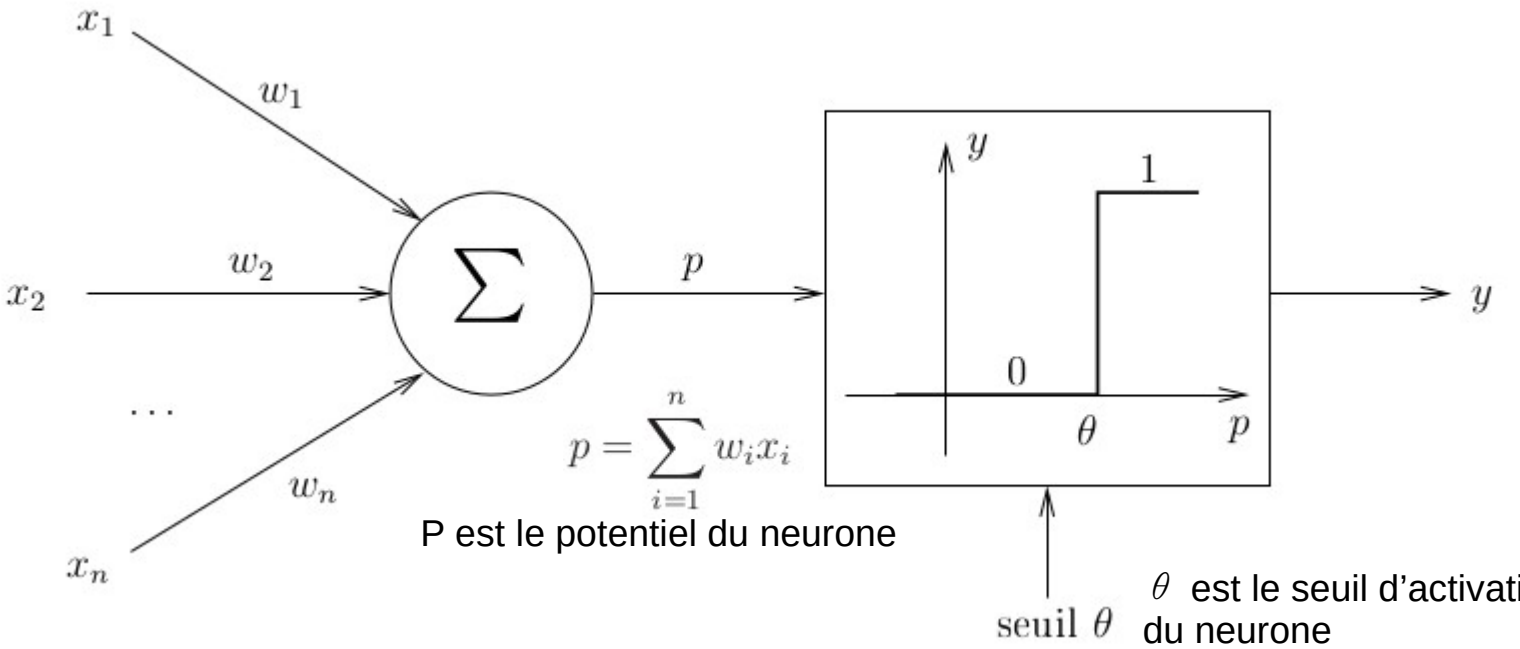
P est le potentiel du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

Le neurone



P est le potentiel du neurone

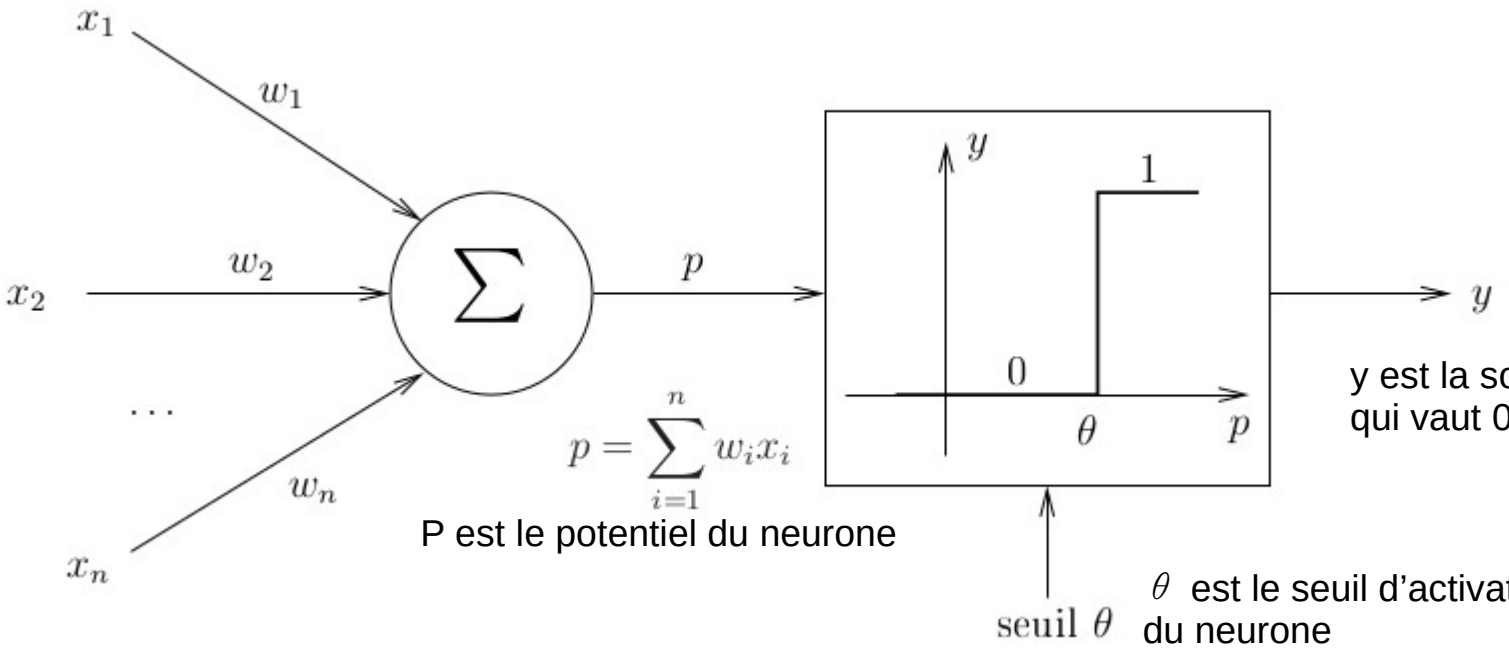
θ est le seuil d'activation (ou le biais) du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

Le neurone



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

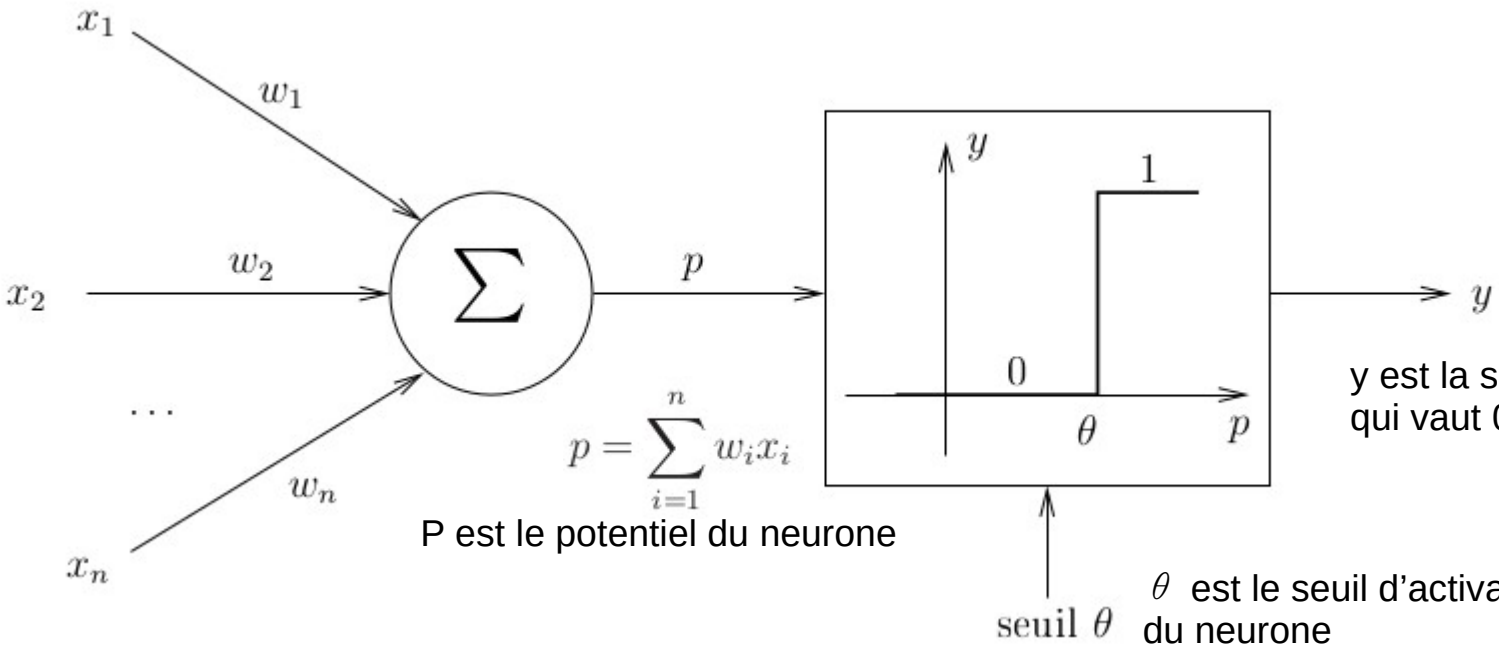
θ est le seuil d'activation (ou le biais) du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

Le neurone



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

θ est le seuil d'activation (ou le biais) du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

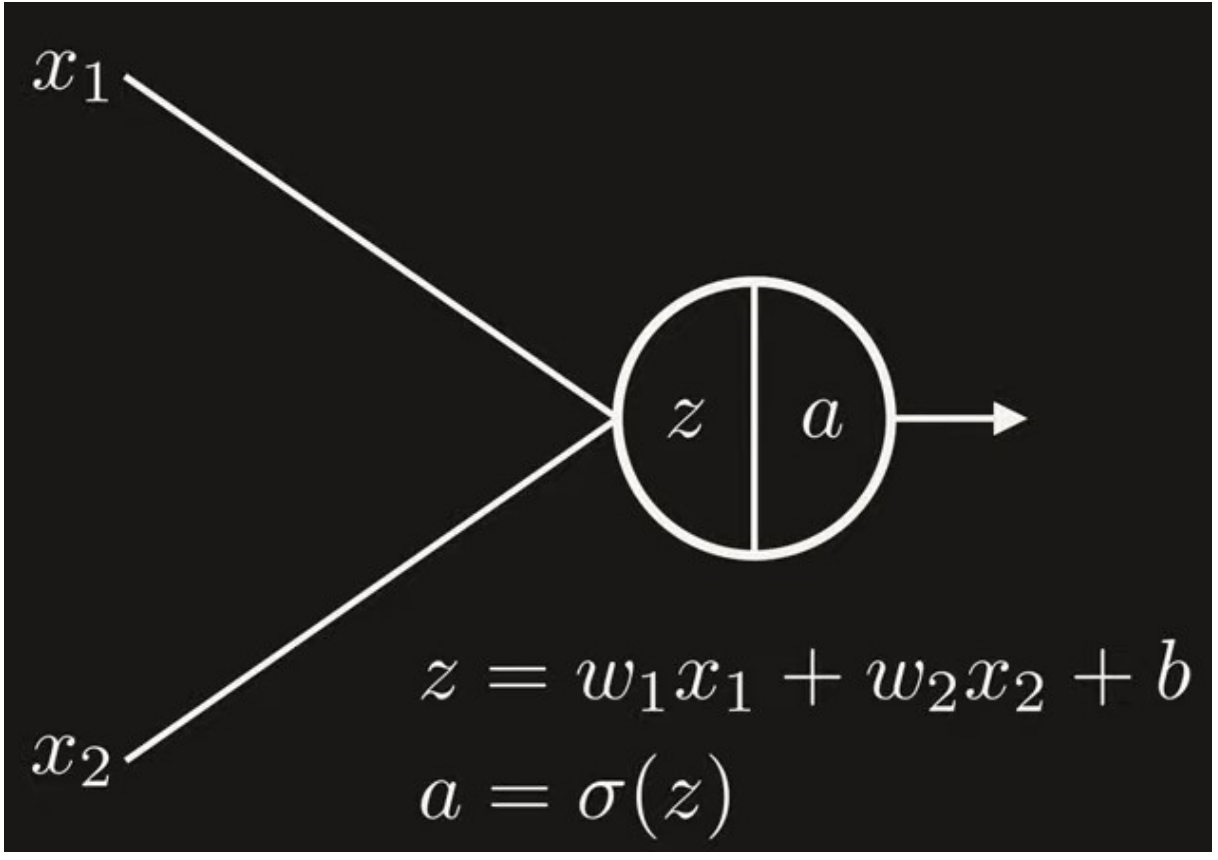
Traiter exemple : x_1, x_2, x_3 = 1 ; 0 ; 1 et ω_1, ω_2, ω_3 = 0,5 ; 1 ; 0,2 et θ = 0,5

Les réseaux de neurones

Le neurone

Voir le site https://playground.tensorflow.org
→ Changer la répartition des données
→ Ajouter des couches cachées
→ Ajouter des neurones

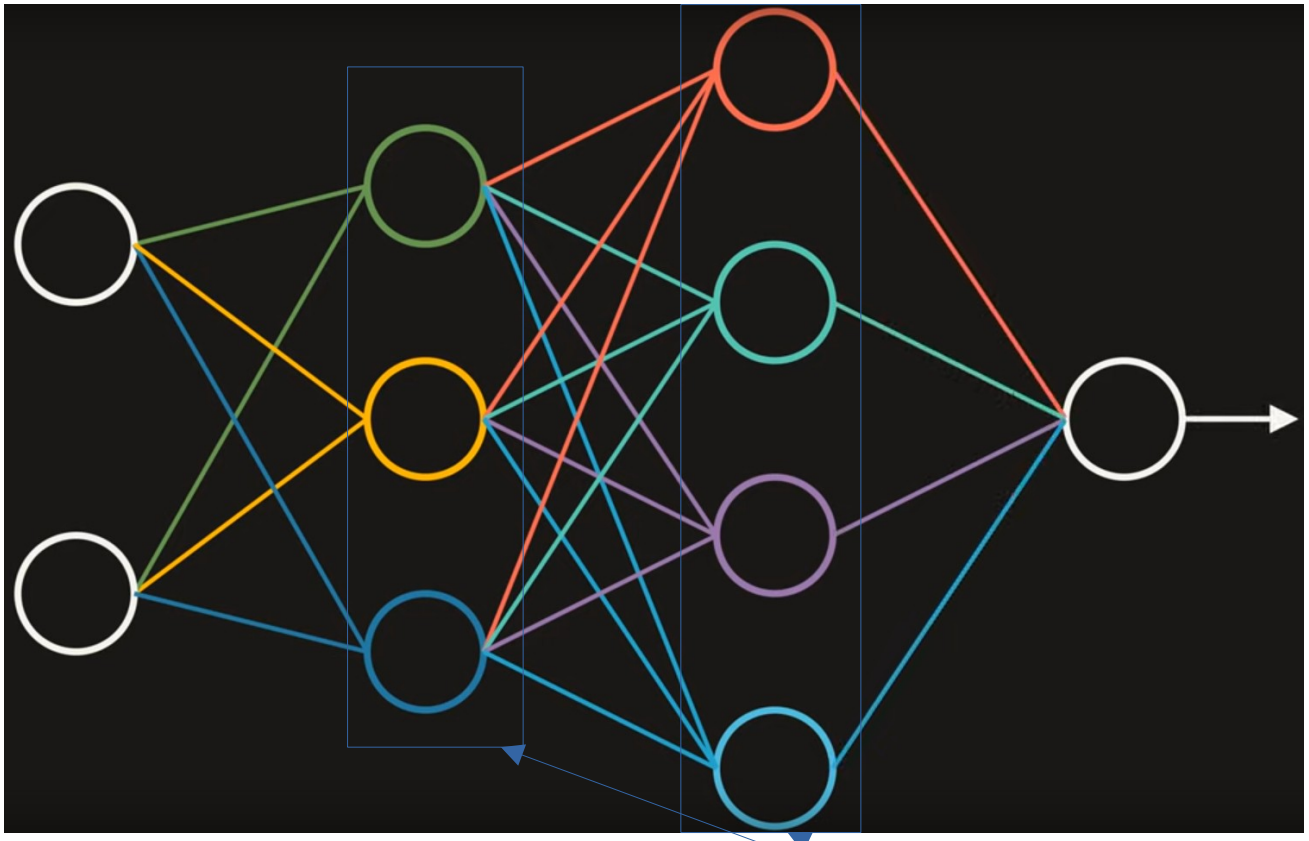
Les réseaux de neurones



b est le biais qui correspond au seuil d'activation θ

σ est la fonction seuil (seuil à 0)

Les réseaux de neurones

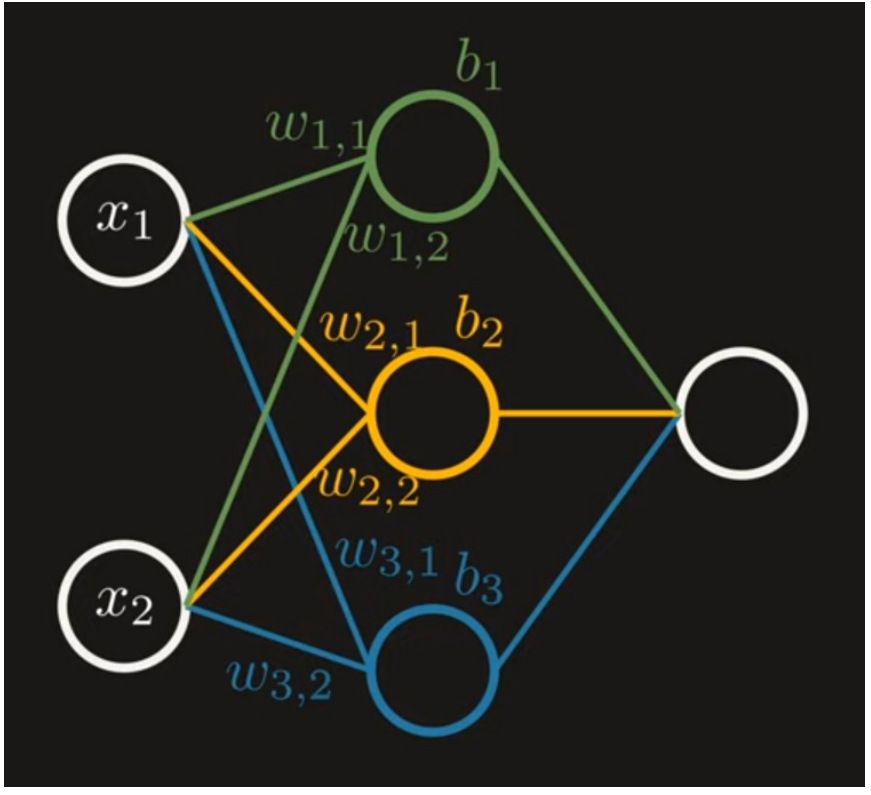


La conception d’un réseau de neurones : choix

Couches de neurones

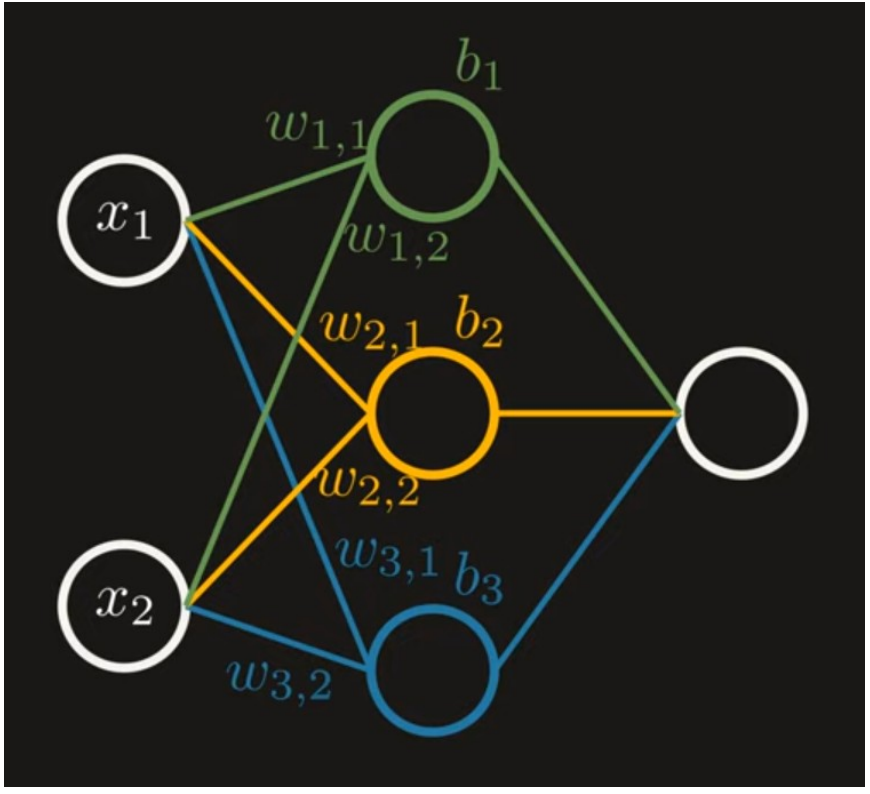
Les réseaux de neurones

Convention de nommage



$\omega_{1,2}$ est le poids synaptique du neurone 1, à l'entrée2
b1 est la place du neurone dans ce réseau

Les réseaux de neurones



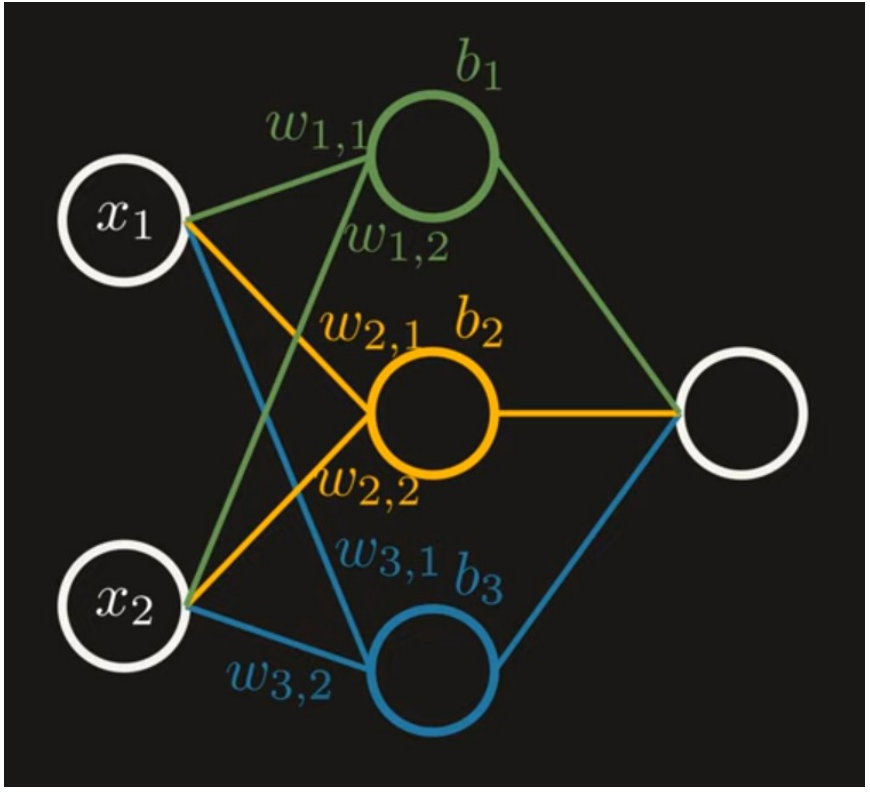
$$z_1 = w_{1,1}x_1 + w_{1,2}x_2 + b_1$$
$$a_1 = \sigma(z_1)$$

$$z_2 = w_{2,1}x_1 + w_{2,2}x_2 + b_2$$
$$a_2 = \sigma(z_2)$$

$$z_1 = w_{3,1}x_1 + w_{3,2}x_2 + b_3$$
$$a_1 = \sigma(z_1)$$

$\omega_{1,2}$ est le poids synaptique du neurone 1, à l'entrée 2
b_1 est la place du neurone dans ce réseau

Les réseaux de neurones



Expression générique :

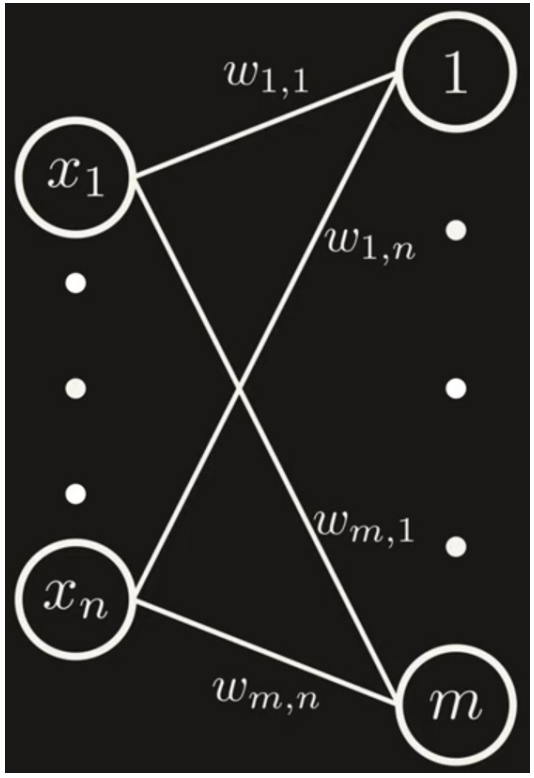
$$z_i = w_{i,1}x_1 + w_{i,2}x_2 + b_i$$

$$a_i = \sigma(z_i)$$

Avec i = 1 ou 2 ou 3

$\omega_{i,2}$ est le poids synaptique du neurone i, à l'entrée 2
b_i est la place du neurone dans ce réseau

Les réseaux de neurones



Expression générique :

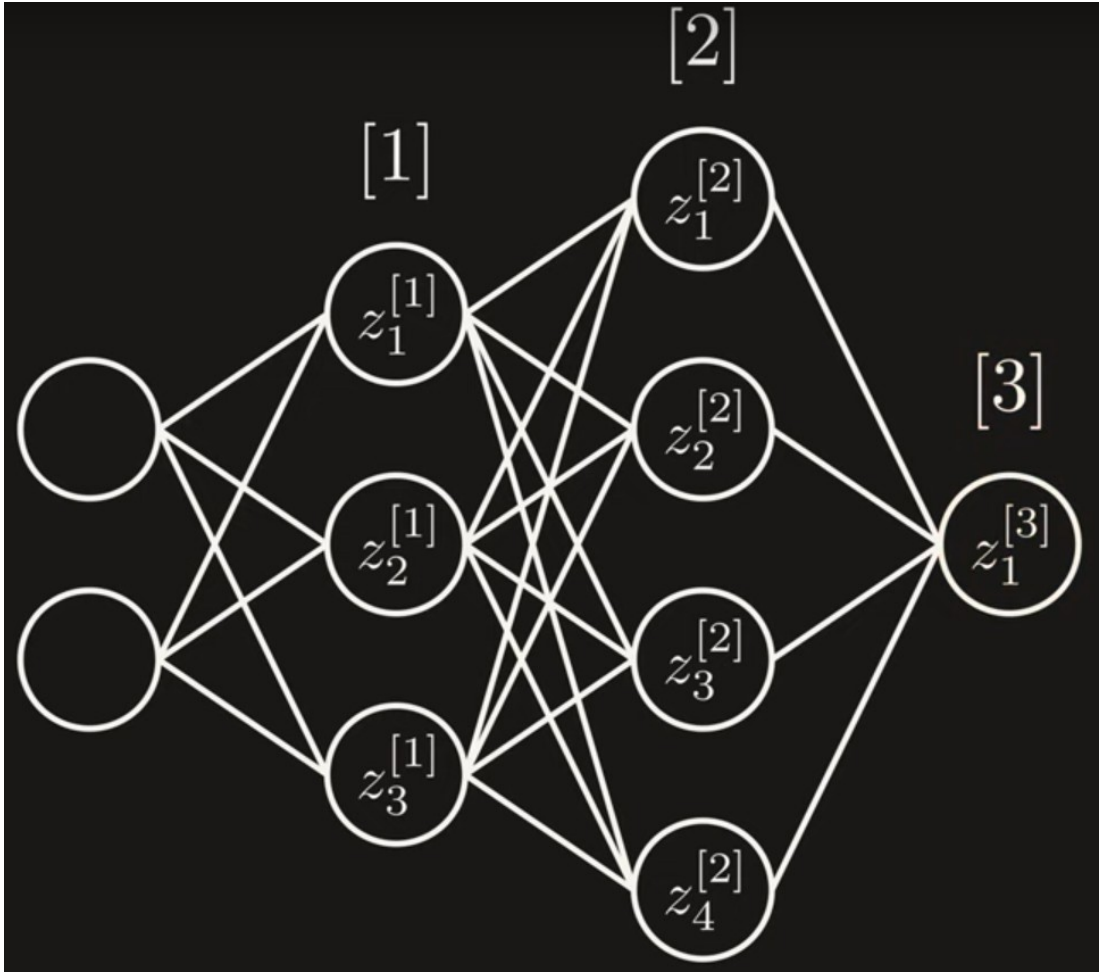
$$z_i = \sum_{j=1}^{n} w_{i,j} x_j + b_i$$

$$a_i = f(z_i)$$

Avec $\omega_{i,j}$ | i : numéro du neurone de sortie
j : numéro de neurone d'entrée

$\omega_{i,j}$ est le poids synaptique du neurone i, à l'entrée j
b_i est la place du neurone dans ce réseau

Les réseaux de neurones



Expression générique :

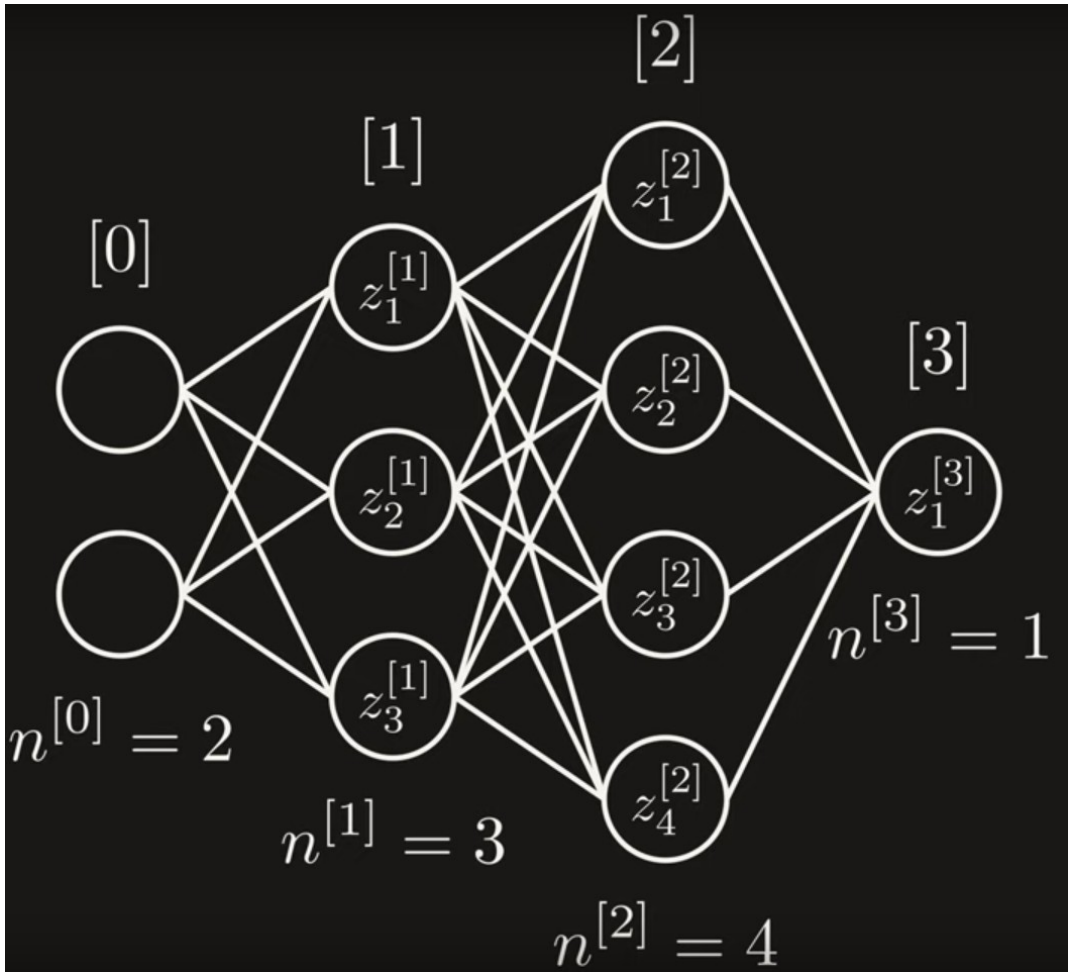
$$z_i^{[l]} = \sum_{j=1}^{n} w_{i,j}^{[l]} a_j^{[l-1]} + b_i^{[l]}$$

$$a_i^{[l]} = f^{[l]}(z_i^{[l]})$$

Avec $\omega_{i,j}$ | i : numéro du neurone de sortie
j : numéro de neurone d'entrée

Et [i] : numéro de la couche du neurone

Les réseaux de neurones



Expression générique :

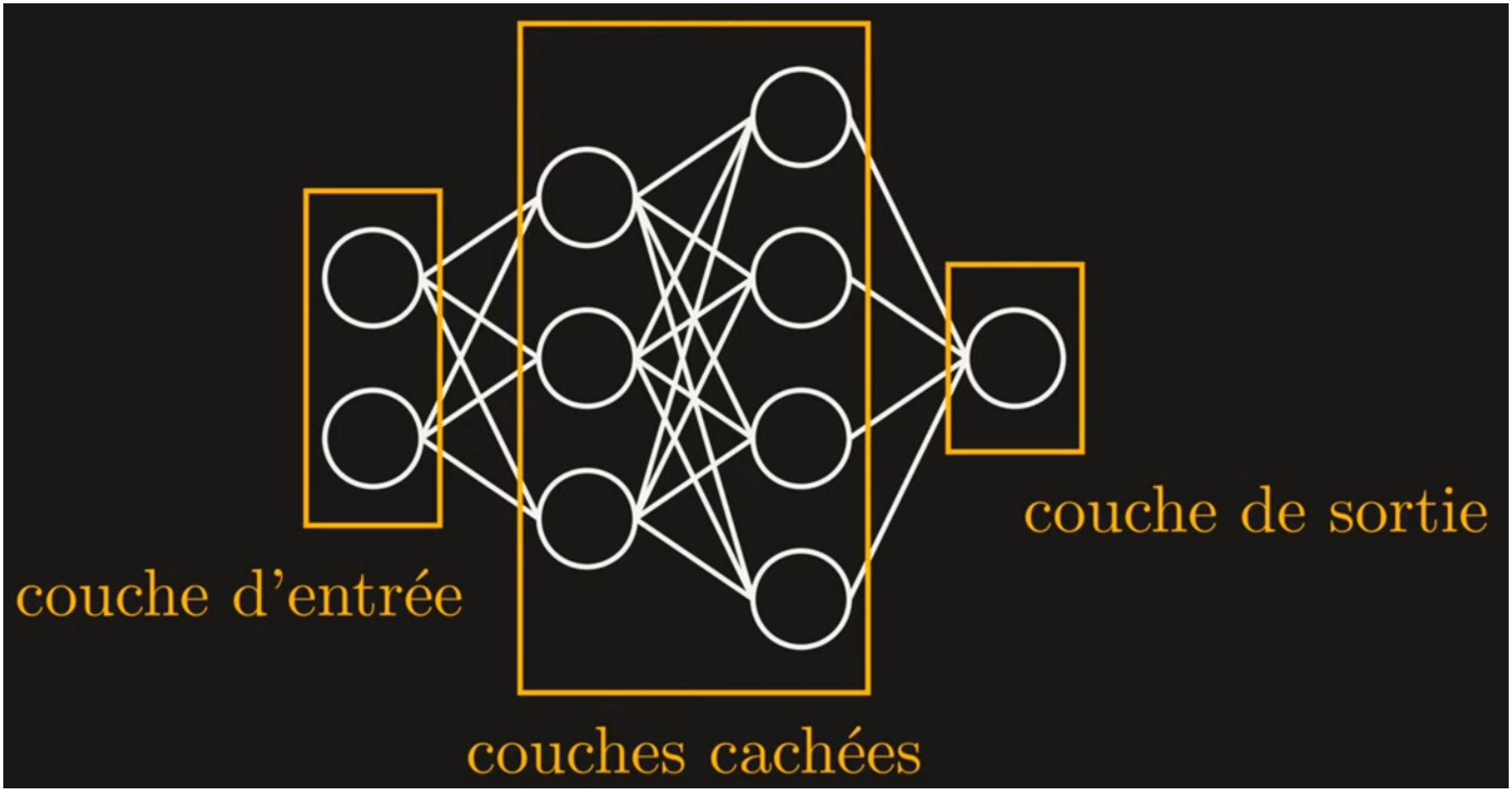
$$z_i^{[l]} = \sum_{j=1}^{n^{[l-1]}} w_{i,j}^{[l]} a_j^{[l-1]} + b_i^{[l]}$$

$$a_i^{[l]} = f^{[l]}(z_i^{[l]})$$

Avec $\omega_{i,j}$ | i : numéro du neurone de sortie
j : numéro de neurone d'entrée

Et [i] : numéro de la couche du neurone

Les réseaux de neurones



Les réseaux de neurones

initialiser les $w_{i,j}^{[l]}$ et $b_i^{[l]}$ quelconques

pour chaque epoch de 1 à N:

forward pass avec tous les exemples

calcul du coût J

backward pass : calcul des $\mathtt{dw}_{i,j}^{[l]}$ et $\mathtt{db}_i^{[l]}$

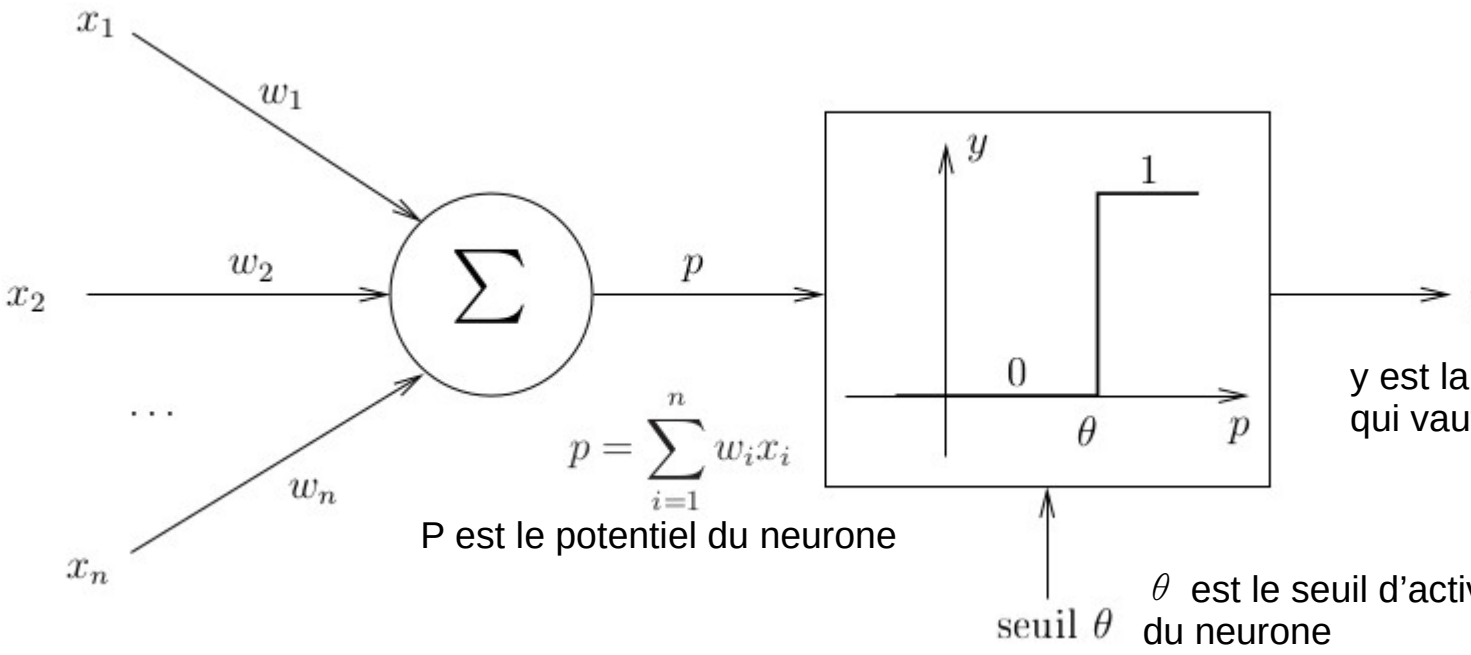
$$w_{i,j}^{[l]} = w_{i,j}^{[l]} - \alpha \mathtt{dw}_{i,j}^{[l]} \qquad \forall l, i, j$$

$$b_i^{[l]} = b_i^{[l]} - \alpha \mathtt{db}_i^{[l]} \qquad \forall l, i$$

Les réseaux de neurones

Comment apprennent-ils ?

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

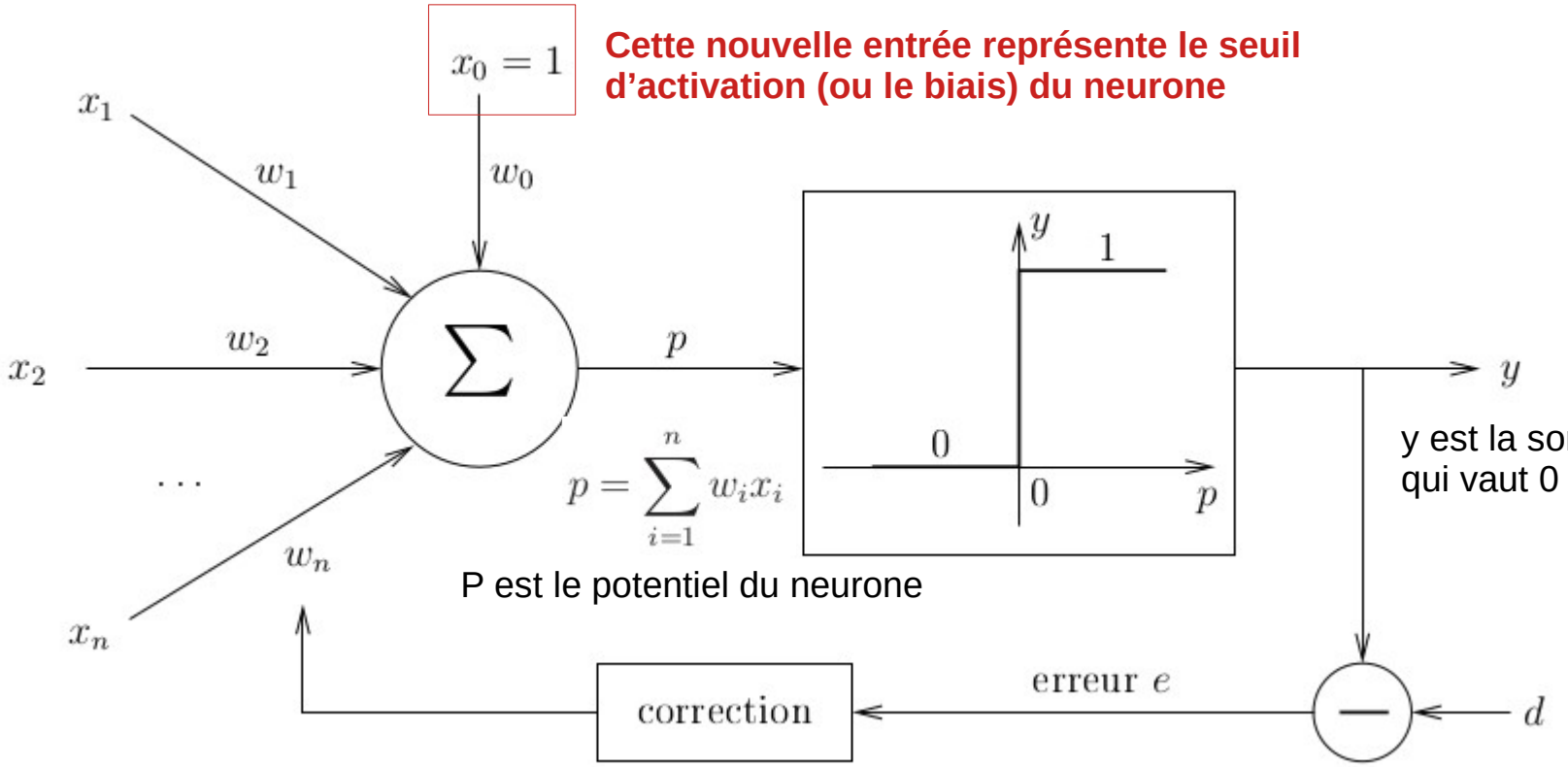
P est le potentiel du neurone

θ est le seuil d'activation (ou le biais) du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones



Cette nouvelle entrée représente le seuil d'activation (ou le biais) du neurone

y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

k	x_1	x_2	d
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

La fonction logique ET

Comment les réseaux de neurone apprennent-ils cette fonction ?

Les réseaux de neurones

k	x_1	x_2	d
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

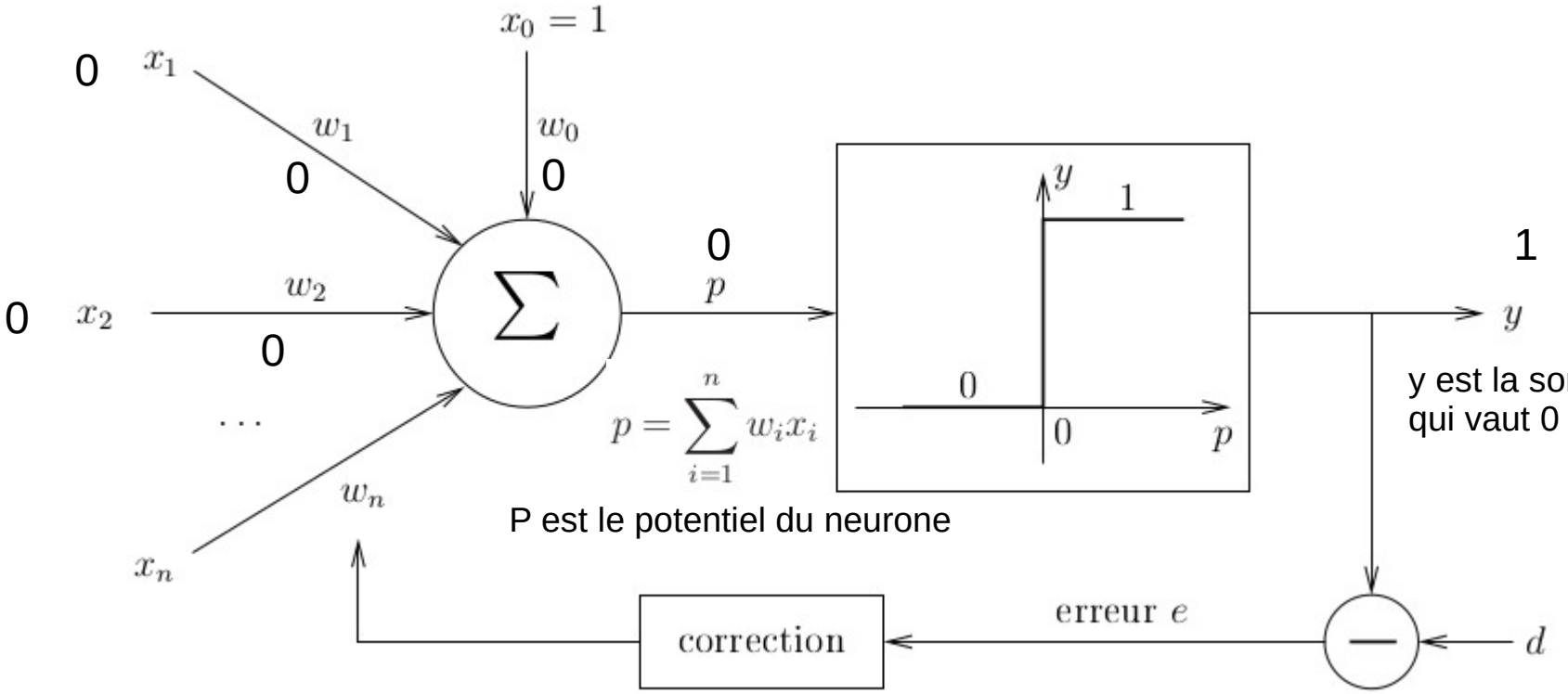
La fonction logique ET

Comment les réseaux de neurone apprennent-ils cette fonction ?

Ici, le réseau dispose de 2 entrées x_1 et x_2 auxquelles on adjoint une entrée fictive x_0 qui sera en permanence égale à 1. Le réseau disposera donc de 3 éléments de mémoire, c'est-à-dire les poids synaptiques ω_0 , ω_1 et ω_2 que nous initialisons à 0 pour l'exemple

K = 1

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

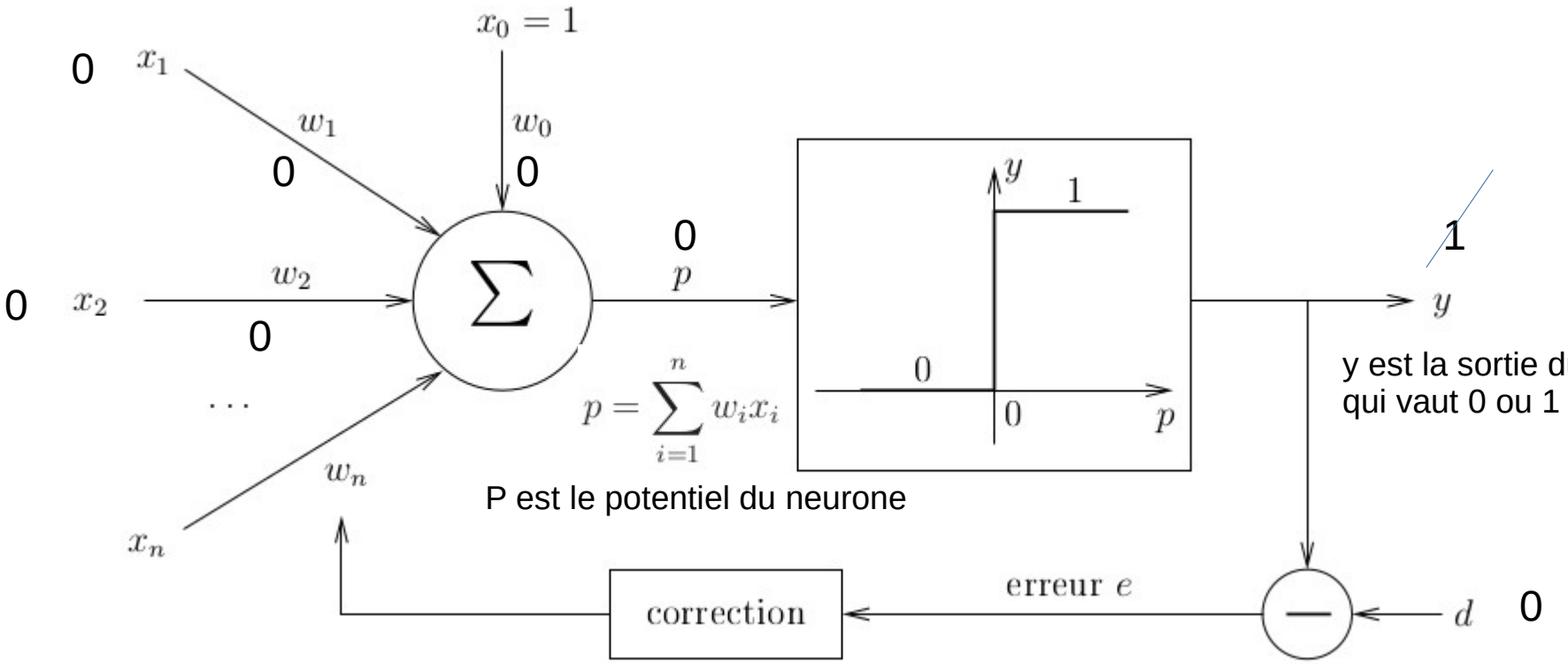
P est le potentiel du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

Les réseaux de neurones

K = 1



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

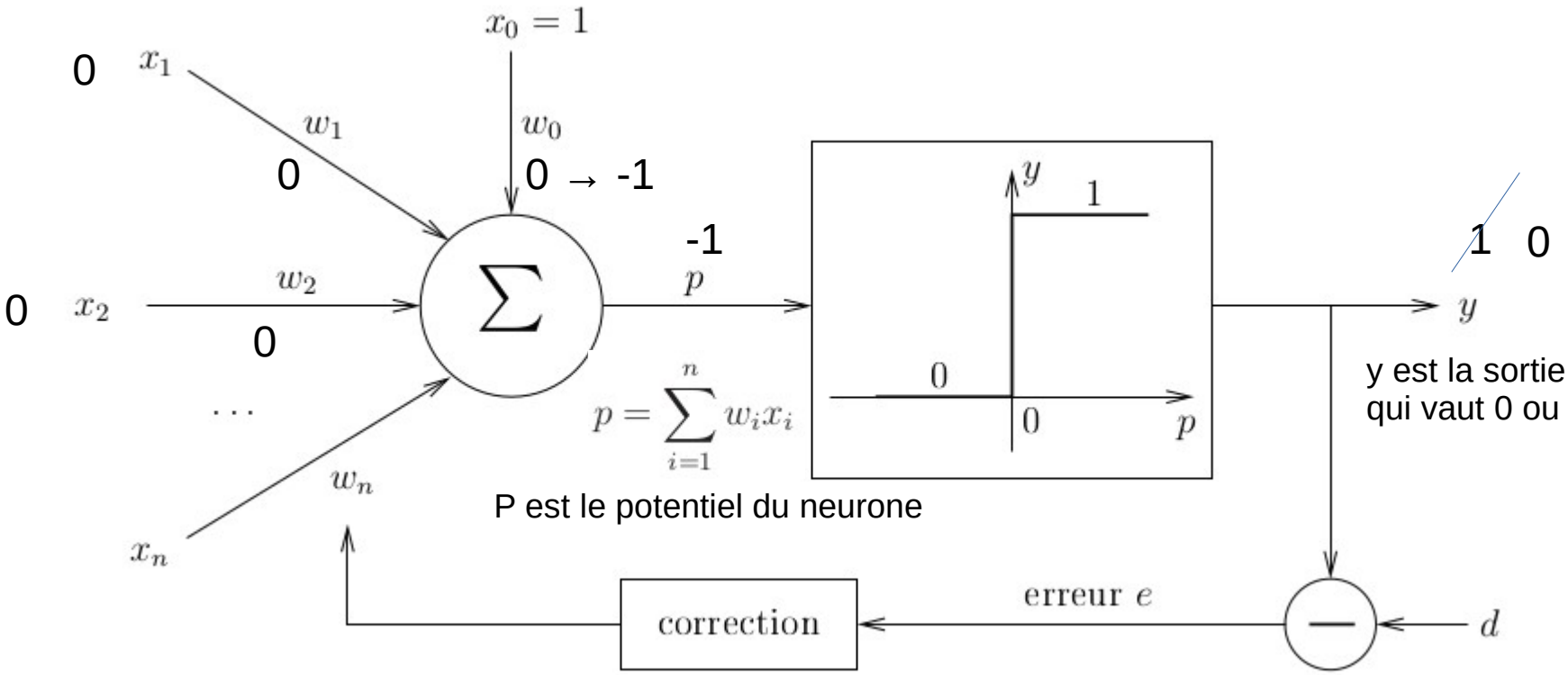
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs++

Les réseaux de neurones

K = 1



1 0

y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

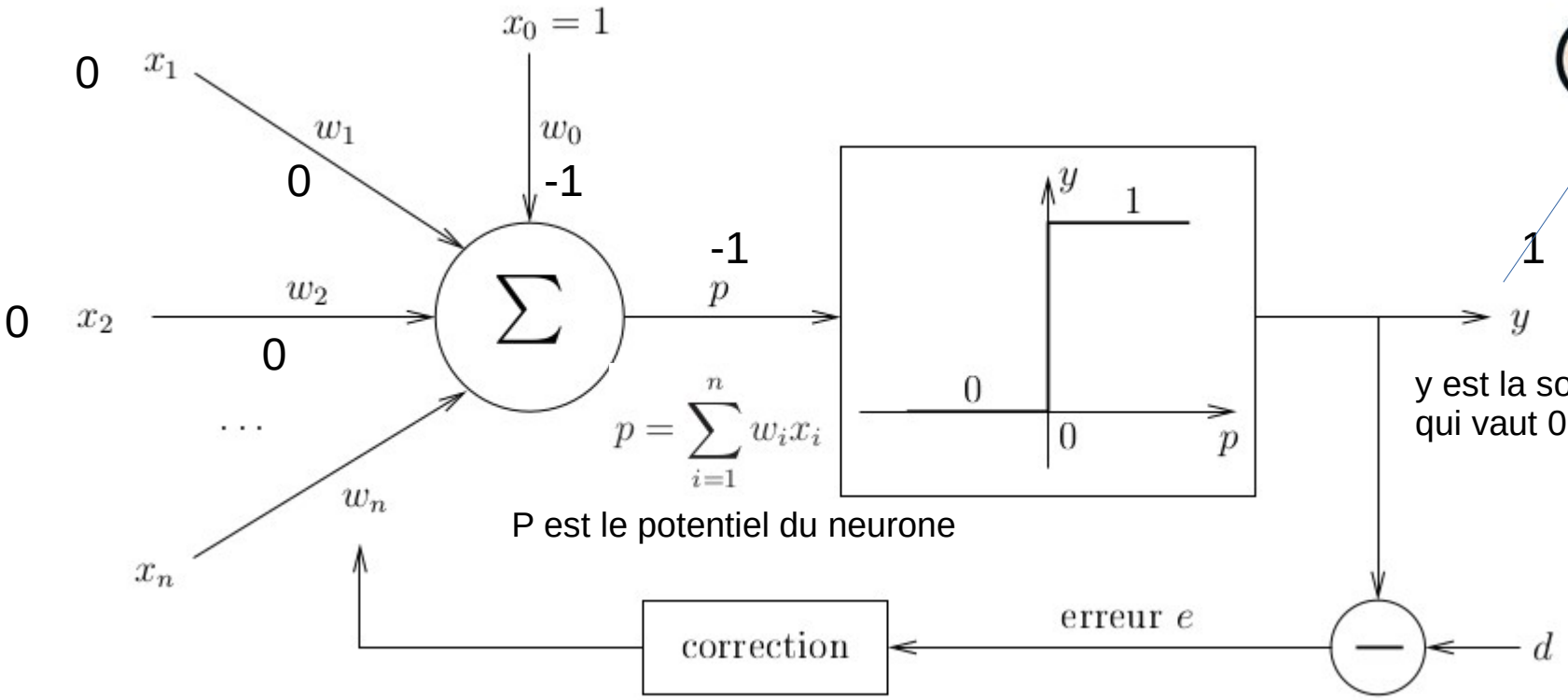
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 1

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

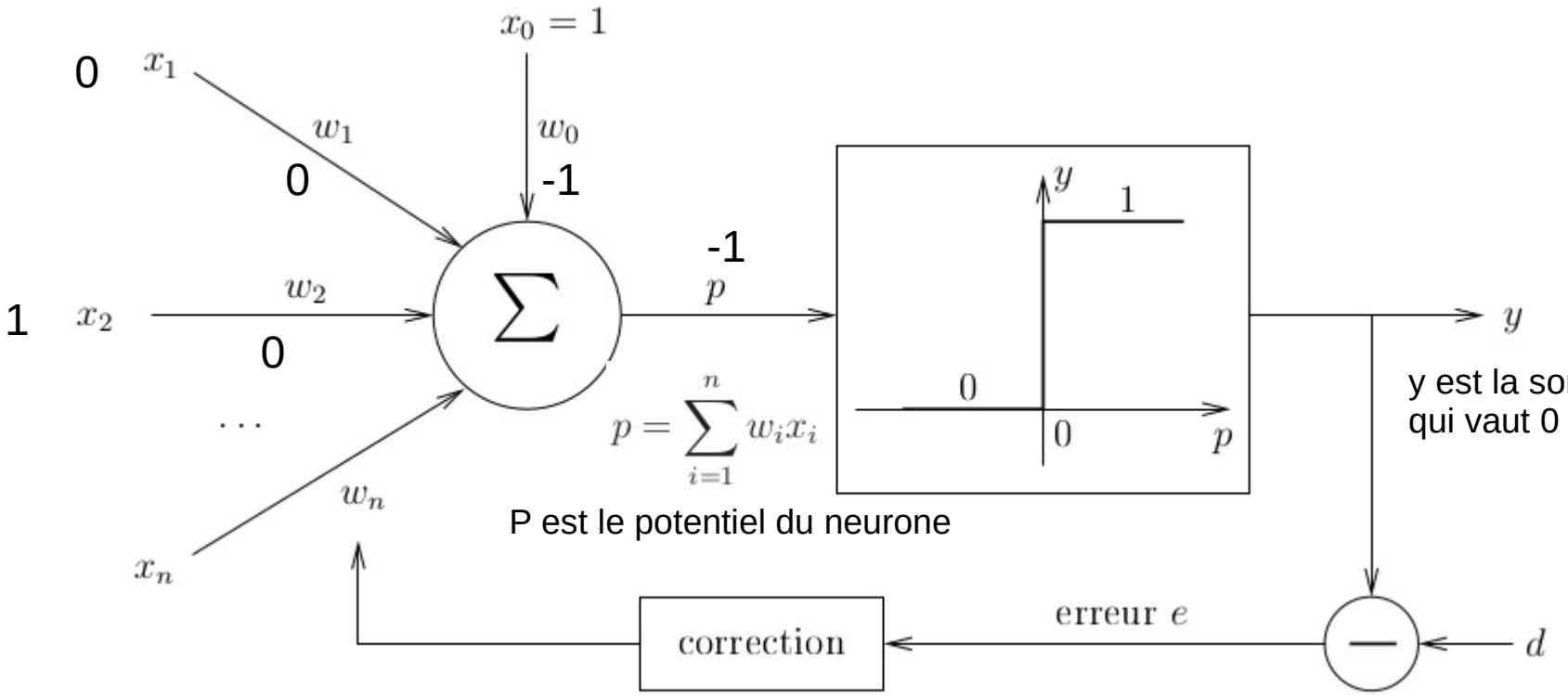
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 2

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

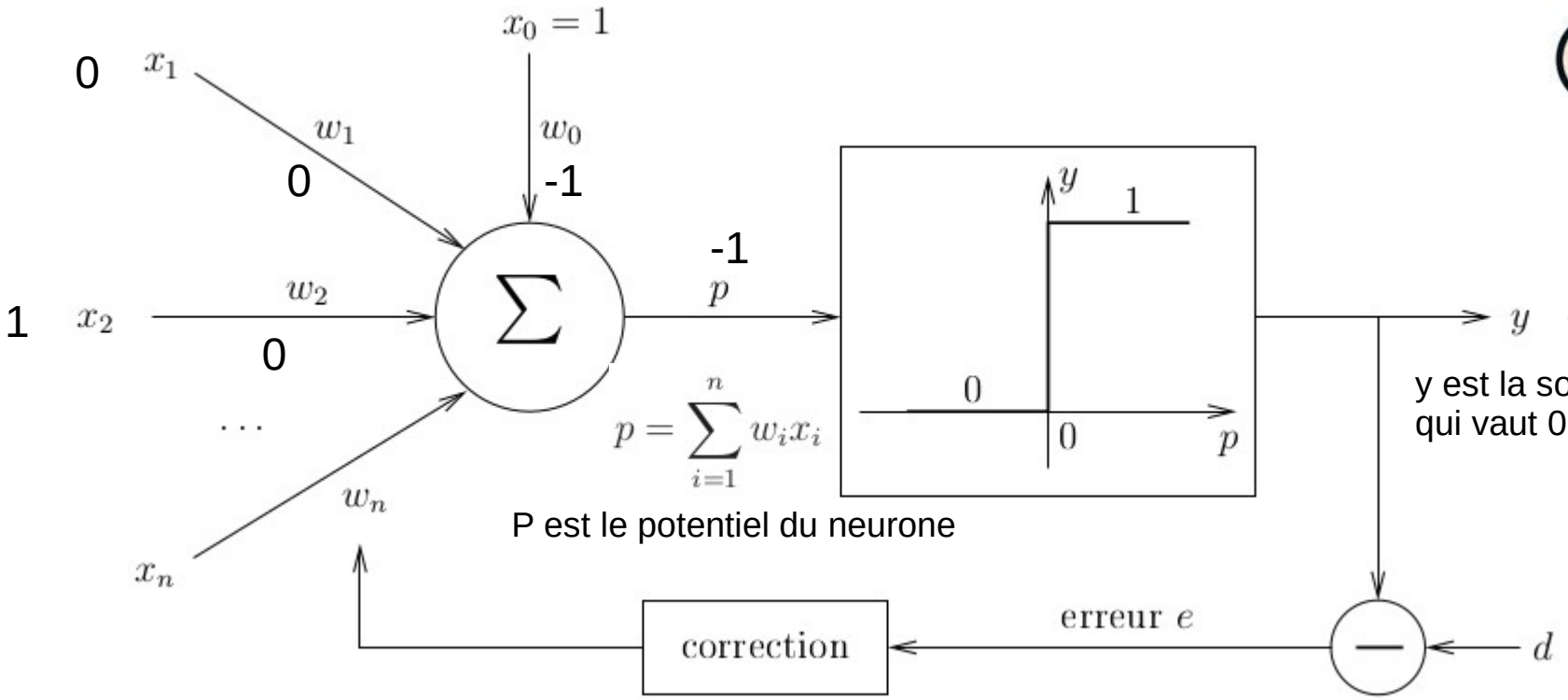
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 2

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

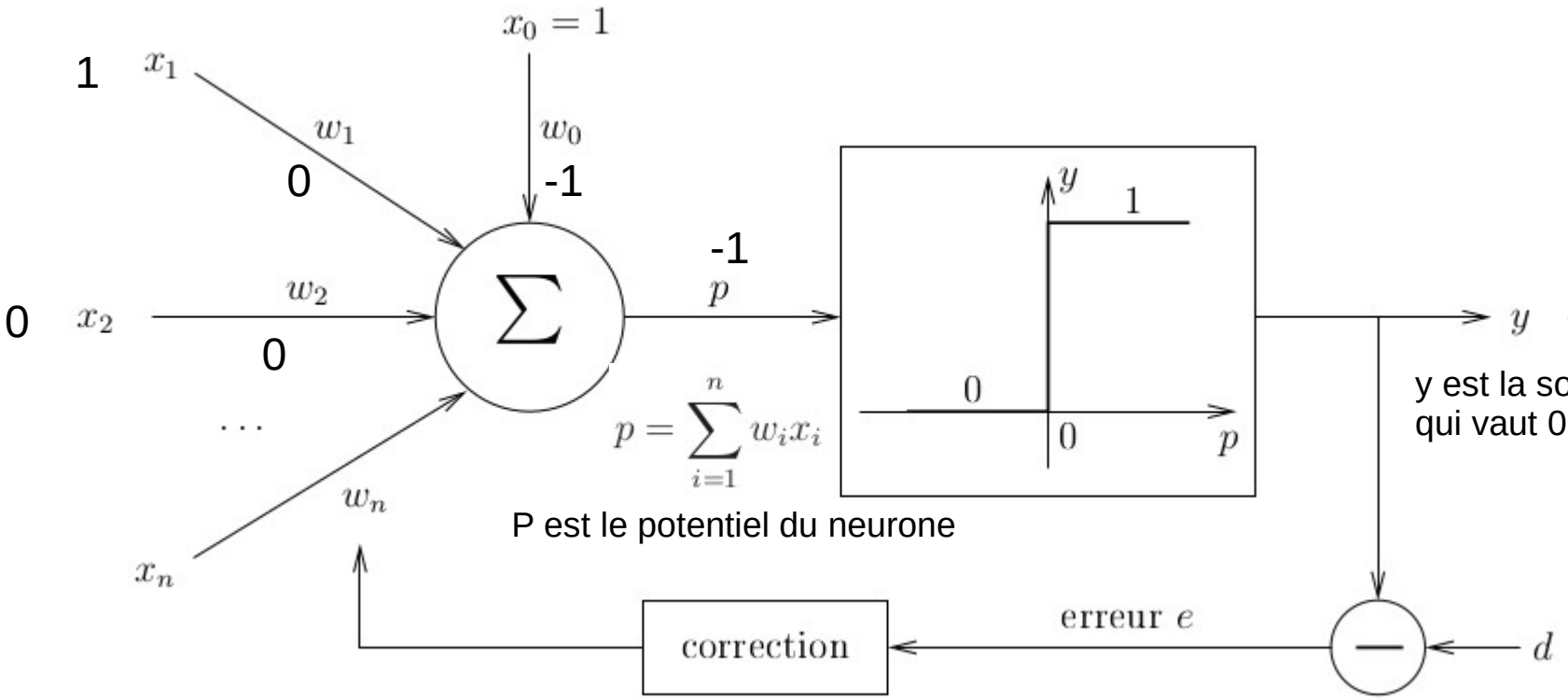
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 3

Les réseaux de neurones



P est le potentiel du neurone

y est la sortie du neurone qui vaut 0 ou 1

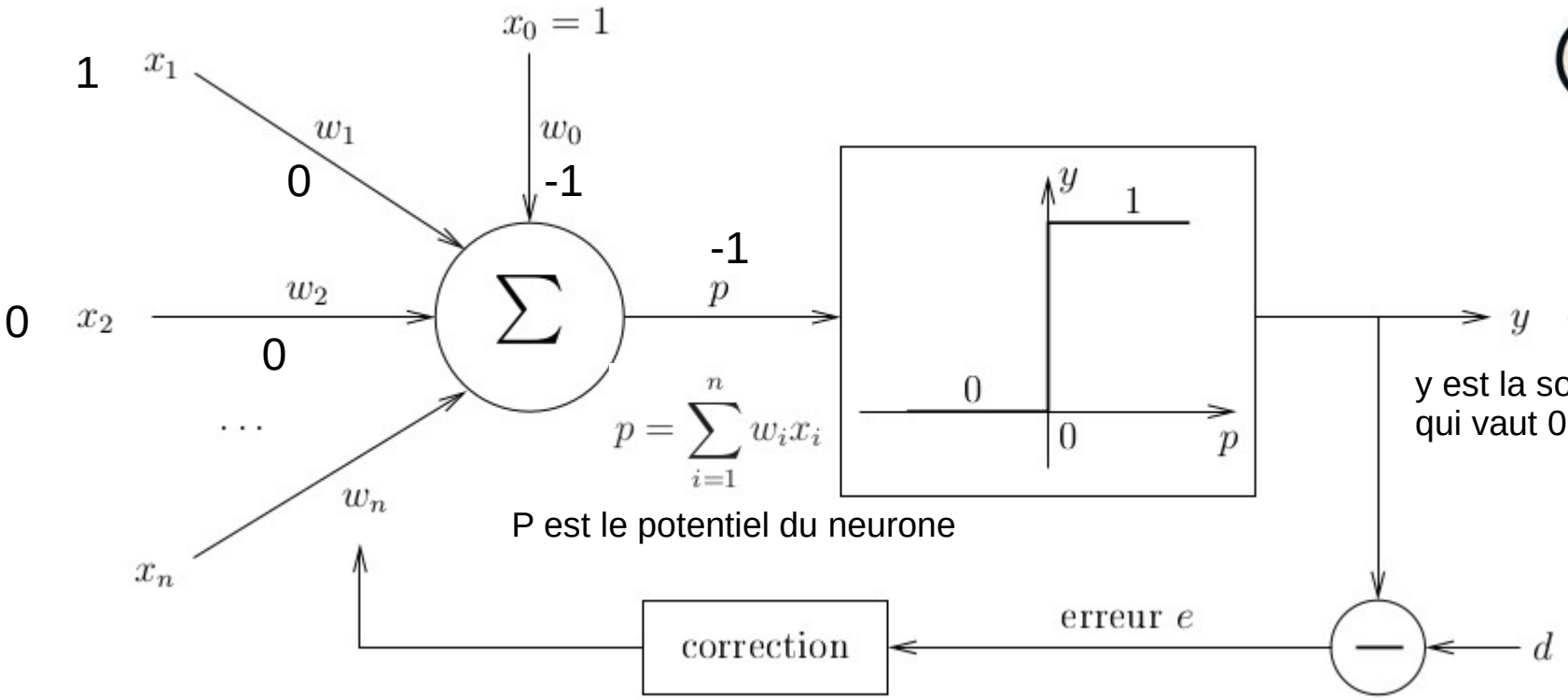
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 3

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

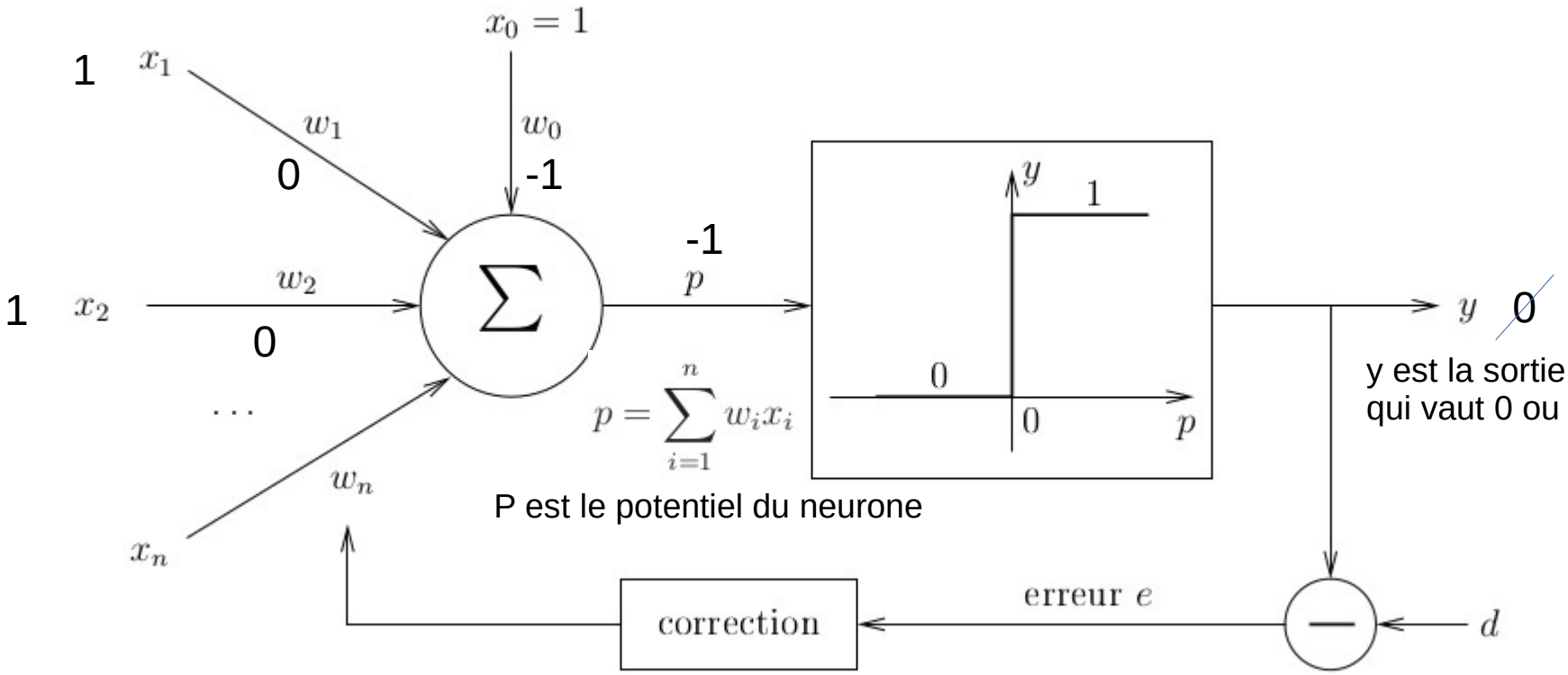
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=1

K = 4

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

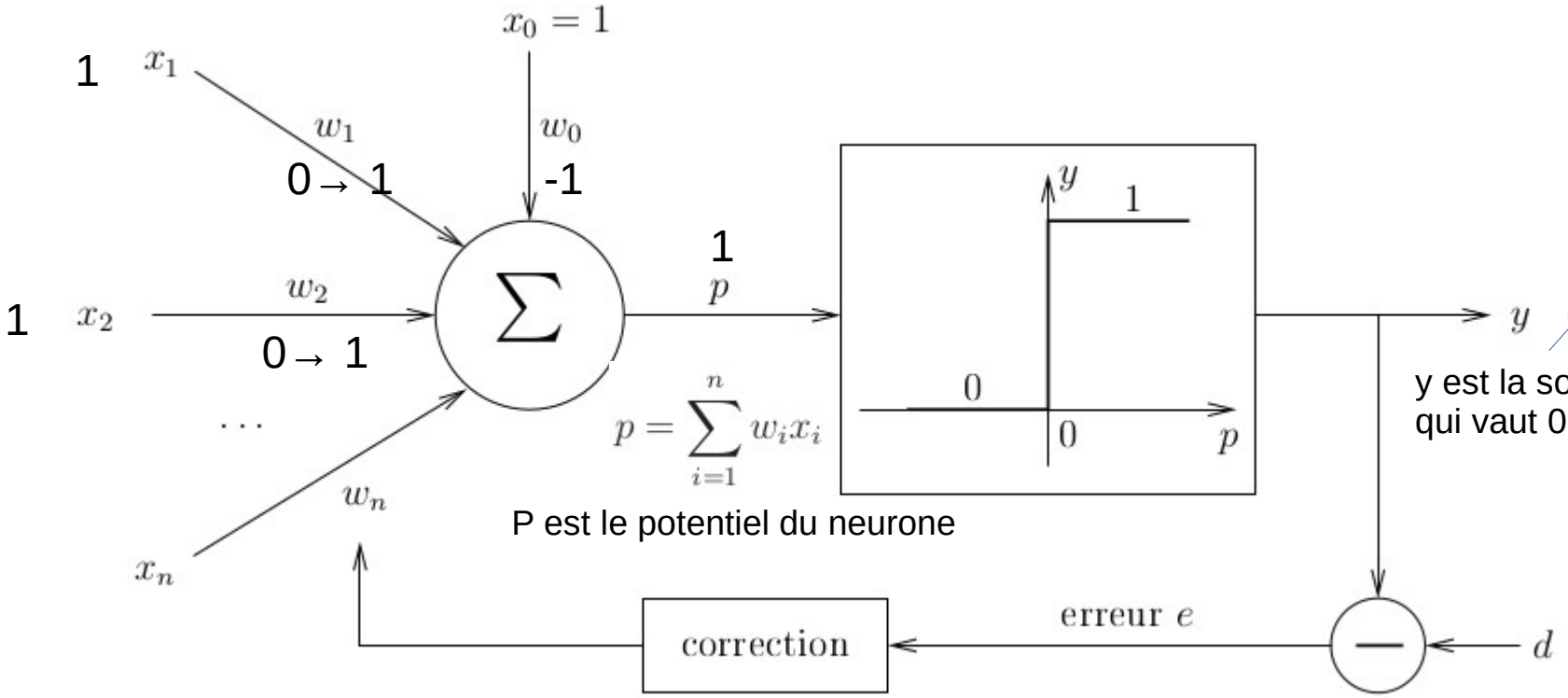
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs++

Les réseaux de neurones

K = 4



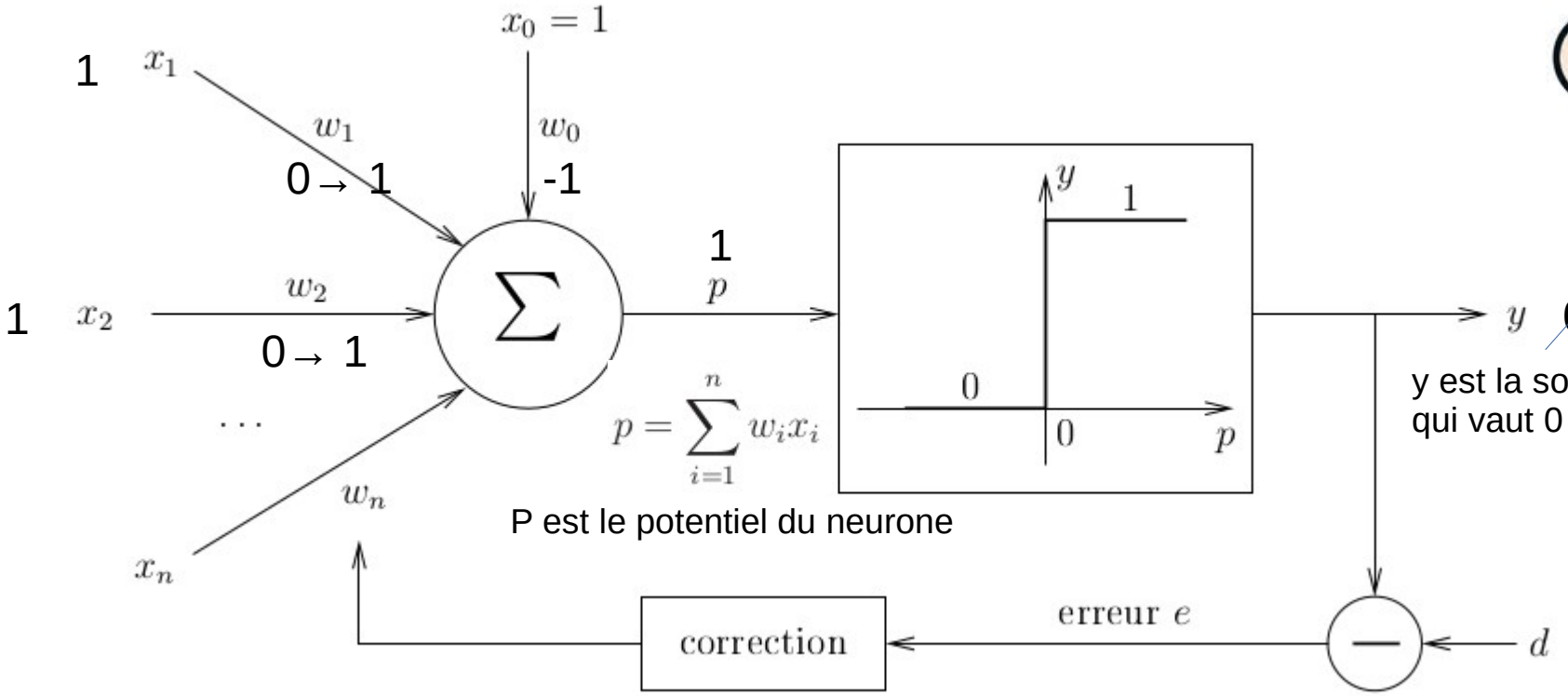
ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=2

K = 4

Les réseaux de neurones



y est la sortie du neurone qui vaut 0 ou 1

P est le potentiel du neurone

ω_1, ω_2, ω_3 sont les poids synaptiques : ils peuvent être positifs ou négatifs

x_1, x_2, x_3 sont les entrées du neurone : ils valent 0 ou 1

nb_erreurs=2

Les réseaux de neurones

Nous venons de faire la 1ère passe.

Le nombre d’erreurs = 2

Les réseaux de neurones

Nous venons de faire la 1ère passe.

Le nombre d’erreurs = 2

Nous allons refaire des passes afin d’obtenir un nombre d’erreurs = 0

Les réseaux de neurones

Nous venons de faire la 1ère passe.

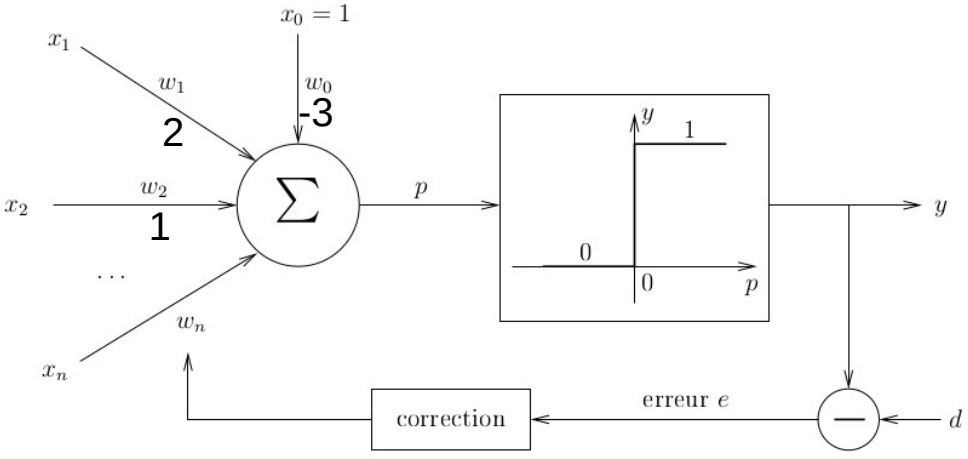
Le nombre d’erreurs = 2

Nous allons refaire des passes afin d’obtenir un nombre d’erreurs = 0

Pour cela, nous allons modifier les poids synaptique ω_0, ω_1, et de ω_2

k	x_1	x_2	d
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

La fonction logique ET



Il faut 6 passes pour apprendre la fonction ET

Itération	k	w_0	w_1	w_2	x_0	x_1	x_2	y	d	e	$nbErreurs$
1	1	0	0	0	1	0	0	1	0	**-1**	2
	2	**-1**	**0**	**0**	1	0	1	0	0	0	
	3	-1	0	0	1	1	0	0	0	0	
	4	-1	0	0	1	1	1	0	1	**1**	
2	1	**0**	**1**	**1**	1	0	0	1	0	**1**	3
	2	**-1**	**1**	**1**	1	0	1	1	0	**-1**	
	3	**-2**	**1**	**0**	1	1	0	0	0	0	
	4	-2	1	0	1	1	1	0	1	**1**	
3	1	**-1**	**2**	**1**	1	0	0	0	0	0	3
	2	-1	2	1	1	0	1	1	0	**-1**	
	3	**-2**	**2**	**0**	1	1	0	1	0	**-1**	
	4	**-3**	**1**	**0**	1	1	1	0	1	**1**	
4	1	**-2**	**2**	**1**	1	0	0	0	0	0	2
	2	-2	2	1	1	0	1	0	0	0	
	3	-2	2	1	1	1	0	1	0	**-1**	
	4	**-3**	**1**	**1**	1	1	1	0	1	**1**	
5	1	**-2**	**2**	**2**	1	0	0	0	0	0	1
	2	-2	2	2	1	0	1	1	0	**-1**	
	3	**-3**	**2**	**1**	1	1	0	0	0	0	
	4	-3	2	1	1	1	1	1	1	0	
6	1	-3	2	1	1	0	0	0	0	0	0
	2	-3	2	1	1	0	1	0	0	0	
	3	-3	2	1	1	1	0	0	0	0	
	4	-3	2	1	1	1	1	1	1	0	

Mon cours

TP4.pdf
www.snoeck.fr
→ ADESFA – Python
→ Cameroun : Le sujet du TP & tous les programmes utilisés

YOLO

You Only Look Once

YOLO

YOLO (You Only Look Once), a popular object detection and image segmentation model, was developed by Joseph Redmon and Ali Farhadi at the University of Washington. Launched in 2015, YOLO quickly gained popularity for its high speed and accuracy.

1) YOLOv2, released in 2016, improved the original model by incorporating batch normalization, anchor boxes, and dimension clusters.
2) YOLOv3, launched in 2018, further enhanced the model's performance using a more efficient backbone network, multiple anchors and spatial pyramid pooling.
3) YOLOv4 was released in 2020, introducing innovations like Mosaic data augmentation, a new anchor-free detection head, and a new loss function.
4) YOLOv5 further improved the model's performance and added new features such as hyperparameter optimization, integrated experiment tracking and automatic export to popular export formats.

YOLO

5) YOLOv6 was open-sourced by Meituan in 2022 and is in use in many of the company's autonomous delivery robots.
6) YOLOv7 added additional tasks such as pose estimation on the COCO keypoints dataset.
7) YOLOv8 released in 2023 by Ultralytics. YOLOv8 introduced new features and improvements for enhanced performance, flexibility, and efficiency, supporting a full range of vision AI tasks,
8) YOLOv9 introduces innovative methods like Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN).
9) YOLOv10 is created by researchers from Tsinghua University using the Ultralytics Python package. This version provides real-time object detection advancements by introducing an End-to-End head that eliminates Non-Maximum Suppression (NMS) requirements.
10) YOLO11 NEW: Ultralytics' latest YOLO models delivering state-of-the-art (SOTA) performance across multiple tasks, including detection, segmentation, pose estimation, tracking, and classification, leverage capabilities across diverse AI applications and domains.

This app is available only on the App Store for iPhone and iPad.

Ultralytics YOLO 4+

Simpler. Smarter. Further.

Ultralytics LLC

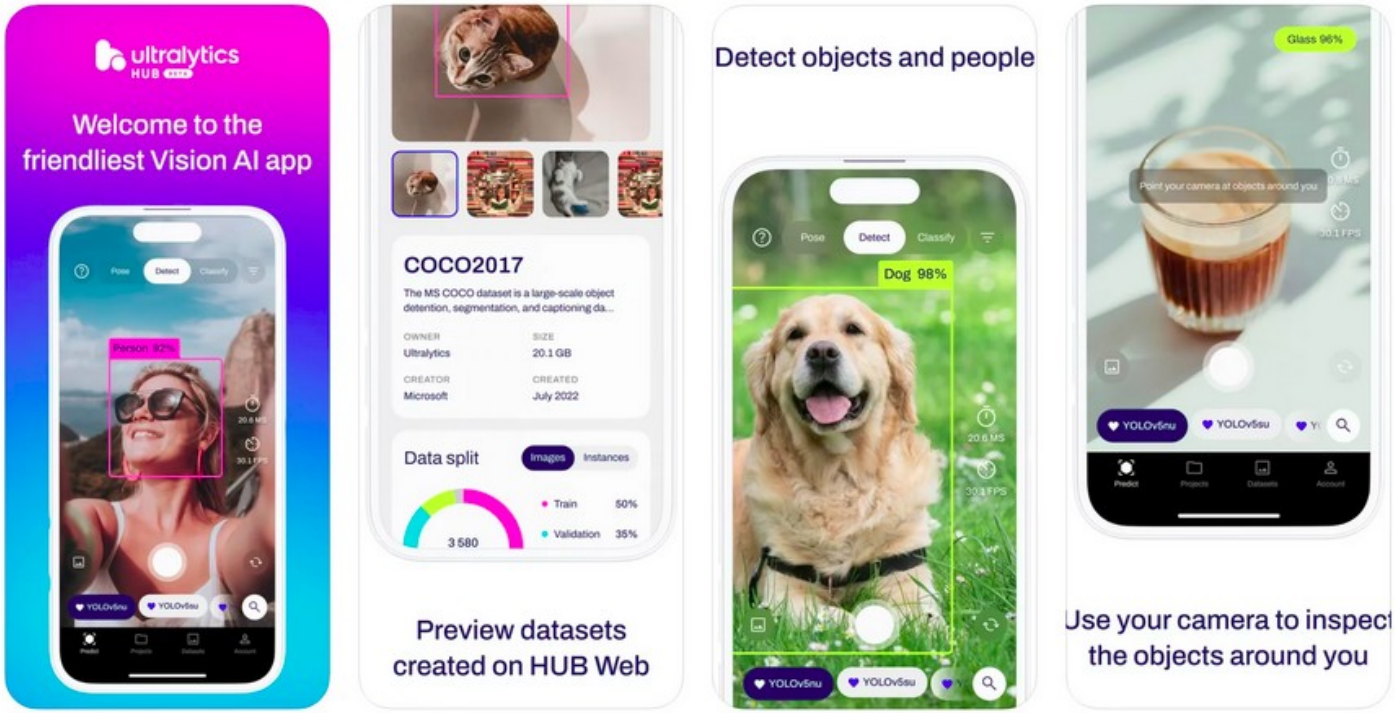
★★★★★ 4.3 • 54 Ratings

Free

Screenshots

iPhone iPad

Ultralytics Android App: Real-time Object Detection with YOLO Models



GET IT ON Google Play

```
sudo apt-get update
sudo apt install python3-pip
sudo pip install scipy
sudo pip install ultralytics opencv-python matplotlib numpy neptune
```

Pip : gestion des packages python (package installer for Python)
Scipy : algorithmes fondamentaux pour les scientifiques
Ultralytics : c’est Ultralytics YOLO...
Opencv-python : algorithmes fondamentaux pour la vision
Matplotlib : algorithmes fondamentaux pour les mathématiciens
Numpy : algorithmes fondamentaux pour les scientifiques
Neptune : site intéressant (mais pas obligatoire) pour conserver les images

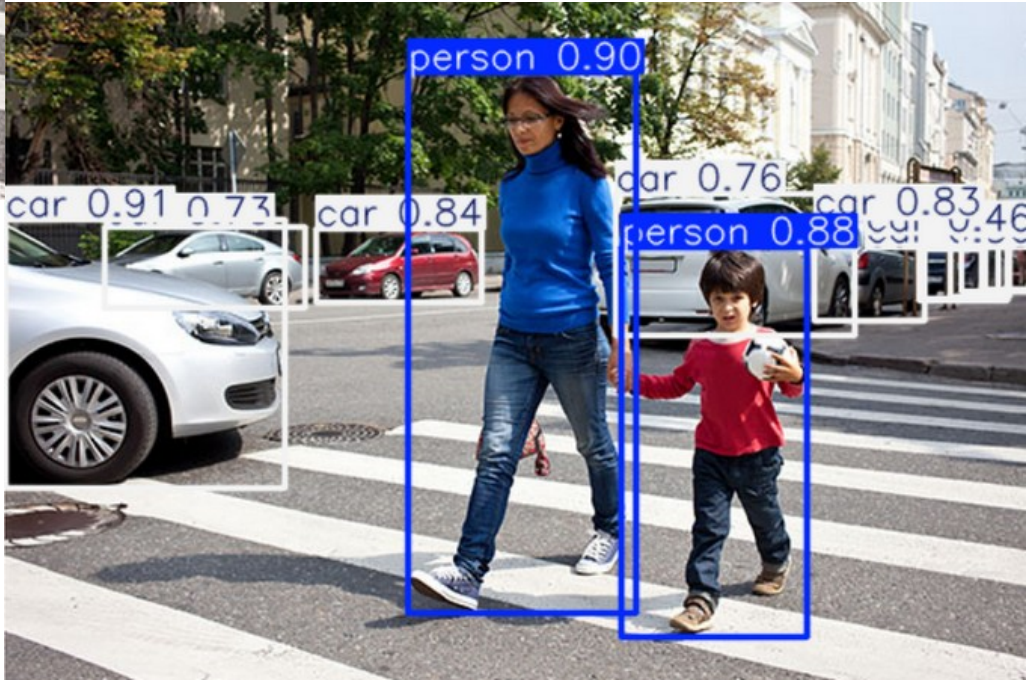
```python
import os

import cv2
import matplotlib.pyplot as plt
import neptune
from ultralytics import YOLO
```

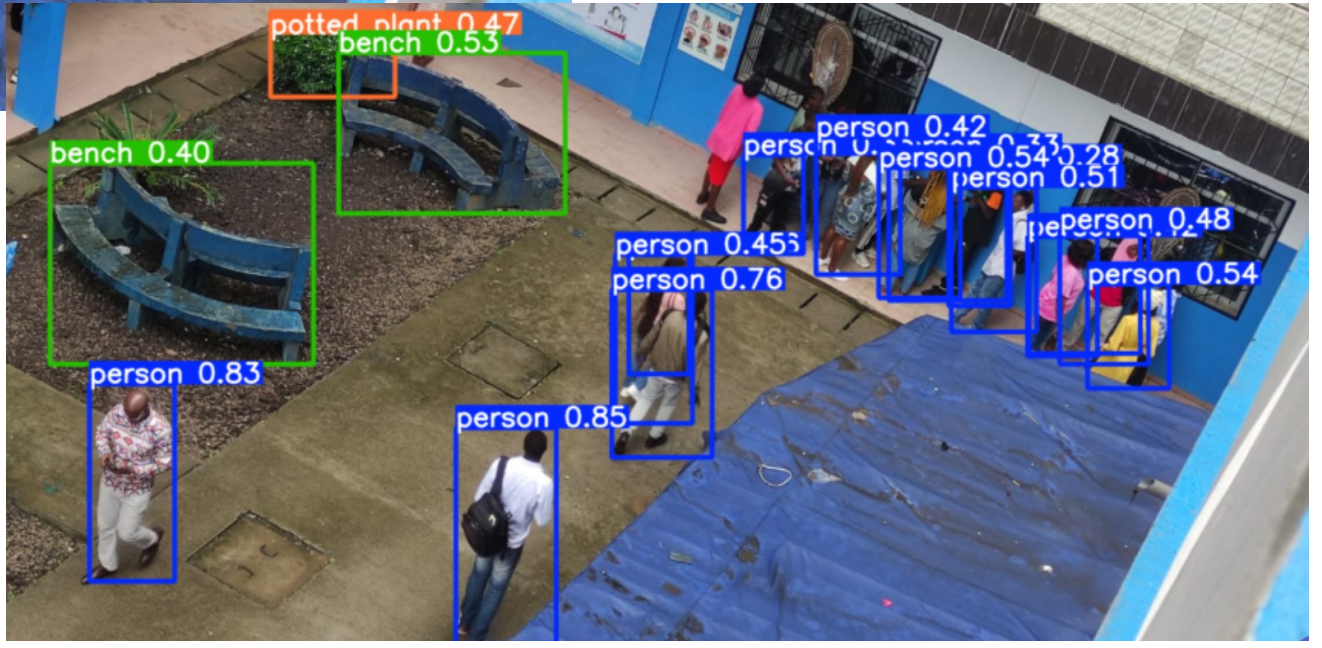
```python
import os

import cv2
import matplotlib.pyplot as plt
import neptune
from ultralytics import YOLO

#pas obligatoire
def init_run(tags=None):
    run = neptune.init_run(
         project="olivier-test/testYOLO11",
         api_token="eyJhcGlfYWRkcmVzcyI6Imh0dHBzOi8vYXBwLm5lcHR1bmUuYWkiLCJhcGlfdXJsIjoiaH

        #project="bexgboost/project",
        #api_token=os.getenv("NEPTUNE_API_TOKEN"),
        tags=tags,
    )
    return run
```

```python
import os

import cv2
import matplotlib.pyplot as plt
import neptune
from ultralytics import YOLO

#pas obligatoire
def init_run(tags=None):
    run = neptune.init_run(
         project="olivier-test/testYOLO11",
         api_token="eyJhcGlfYWRkcmVzcyI6Imh0dHBzOi8vYXBwLm5lcHR1bmUuYWkiLCJhcGlfdXJsIjoiaH

        #project="bexgboost/project",
        #api_token=os.getenv("NEPTUNE_API_TOKEN"),
        tags=tags,
    )
    return run


MODEL_NAME = "yolo11n.pt" #choix de la version YOLO (ici, la version 11/11) et le type de version (ici n ou s ou m ou l ou x): à

model = YOLO(MODEL_NAME)

#pas obligatoire
run = init_run(['yolo-detection'])

# Log model configuration
run["model/task"] = model.task
run["model/name"] = MODEL_NAME
```

```python
MODEL_NAME = "yolo11n.pt" #choix de la version YOLO (ici, la version 11/11) et le type de version (ici n ou s ou m ou l ou x): à

model = YOLO(MODEL_NAME)

#pas obligatoire
run = init_run(['yolo-detection'])

# Log model configuration
run["model/task"] = model.task
run["model/name"] = MODEL_NAME

img1_path = "../Images/crosswalk.jpg"
img2_path = "../Images/pedestrians.jpg"
img3_path = "../Images/INUBIL.jpg"
img3_path = "../Images/ChatChien.jpg"

results = model(img3_path)
#results = model.train(data="coco128.yaml", epochs=5, device="cpu")

#best_model = YOLO("runs/detect/train/weights/best.pt")
#results = best_model(img1_path)
```

```python
MODEL_NAME = "yolo11n.pt" #choix de la version YOLO (ici, la version 11/11) et le type de version (ici n ou s ou m ou l ou x): à

model = YOLO(MODEL_NAME)

#pas obligatoire
run = init_run(['yolo-detection'])

# Log model configuration
run["model/task"] = model.task
run["model/name"] = MODEL_NAME

img1_path = "../Images/crosswalk.jpg"
img2_path = "../Images/pedestrians.jpg"
img3_path = "../Images/INUBIL.jpg"
img3_path = "../Images/ChatChien.jpg"

results = model(img3_path)
#results = model.train(data="coco128.yaml", epochs=5, device="cpu")

#best_model = YOLO("runs/detect/train/weights/best.pt")
#results = best_model(img1_path)

# Plot and log the results
fig, ax = plt.subplots(figsize=(12, 8))

ax.imshow(cv2.cvtColor(results[0].plot(), cv2.COLOR_BGR2RGB))
ax.axis("off")

# Upload the image to Neptune
run["predictions/sample"].upload(neptune.types.File.as_image(fig))

plt.show()
```


person 0.90
car 0.91
car 0.84
car 0.76
car 0.83
person 0.88


potted plant 0.47
bench 0.53
bench 0.40
person 0.42
person 0.54
person 0.51
person 0.48
person 0.54
person 0.76
person 0.83
person 0.85


dog 0
dog 0.76

Sources

https://docs.ultralytics.com/

https://playground.tensorflow.org

Alexandre TL sur youtube pour les images

Introduction aux réseaux de neurones artificiels.pdf

https://neptune.ai

```
import os


import cv2
import matplotlib.pyplot as plt
import neptune
from ultralytics import YOLO


def init_run(tags=None):
  run = neptune.init_run(
                              project="oliv ah O11",
            api_token="eyJhcGlfYWRkc ahahah M6Ly9hcHAubmVwdHVuZS5haSIsImFwaV9rZXkiOiI1MjQ1N2E0NC1kNTA5LTRkNGEtYfQ==",


      #project="bexgboost/project",
      #api_token=os.getenv("NEPTUNE_API_TOKEN"),
      tags=tags,
  )
  return run


MODEL_NAME = "yolo11n.pt"


model = YOLO(MODEL_NAME)


run = init_run(['yolo-detection'])


# Log model configuration
run["model/task"] = model.task
run["model/name"] = MODEL_NAME


img1_path = "../Images/crosswalk.jpg"
img2_path = "../Images/pedestrians.jpg"
img3_path = "../Images/INUBIL.jpg"
img3_path = "../Images/ChatChien.jpg"


results = model(img3_path)
#results = model.train(data="coco128.yaml", epochs=5, device="cpu")
```

Avec une photo

```
best_model = YOLO("runs/detect/train/weights/best.pt")


results = best_model(img1_path)


# Plot and log the results
fig, ax = plt.subplots(figsize=(12, 8))


ax.imshow(cv2.cvtColor(results[0].plot(), cv2.COLOR_BGR2RGB))
ax.axis("off")


# Upload the image to Neptune
run["predictions/sample"].upload(neptune.types.File.as_image(fig))


plt.show()
```

Avec la caméra

```
import os


import cv2
import matplotlib.pyplot as plt
import neptune
from ultralytics import YOLO


def init_run(tags=None):
  run = neptune.init_run(
                          project="olivier-test/tO11",
            api_token="eyJhcGlfYWRkcmVzcyI6Imh0dHBzOi8vYXBwLm5lcHR1bsIjoiaHR0cHM6Ly9hcHAubmVwdHVuZS5haSIsImFwaV9rZXkiOiI1MjQ1N2E0NC1kNTA5LTRkNGEtYTY5Zi0wMmM0N2RmYmIzODkifQ==",


      #project="bexgboost/project",
      #api_token=os.getenv("NEPTUNE_API_TOKEN"),
      tags=tags,
  )
  return run


MODEL_NAME = "yolo11n.pt"


model = YOLO(MODEL_NAME)


run = init_run(['yolo-detection'])


# Log model configuration
run["model/task"] = model.task
run["model/name"] = MODEL_NAME


img1_path = "../Images/crosswalk.jpg"
img2_path = "../Images/pedestrians.jpg"
img3_path = "../Images/INUBIL.jpg"
img3_path = "../Images/ChatChien.jpg"


results = model(img3_path)
#results = model.train(data="coco128.yaml", epochs=5, device="cpu")
```