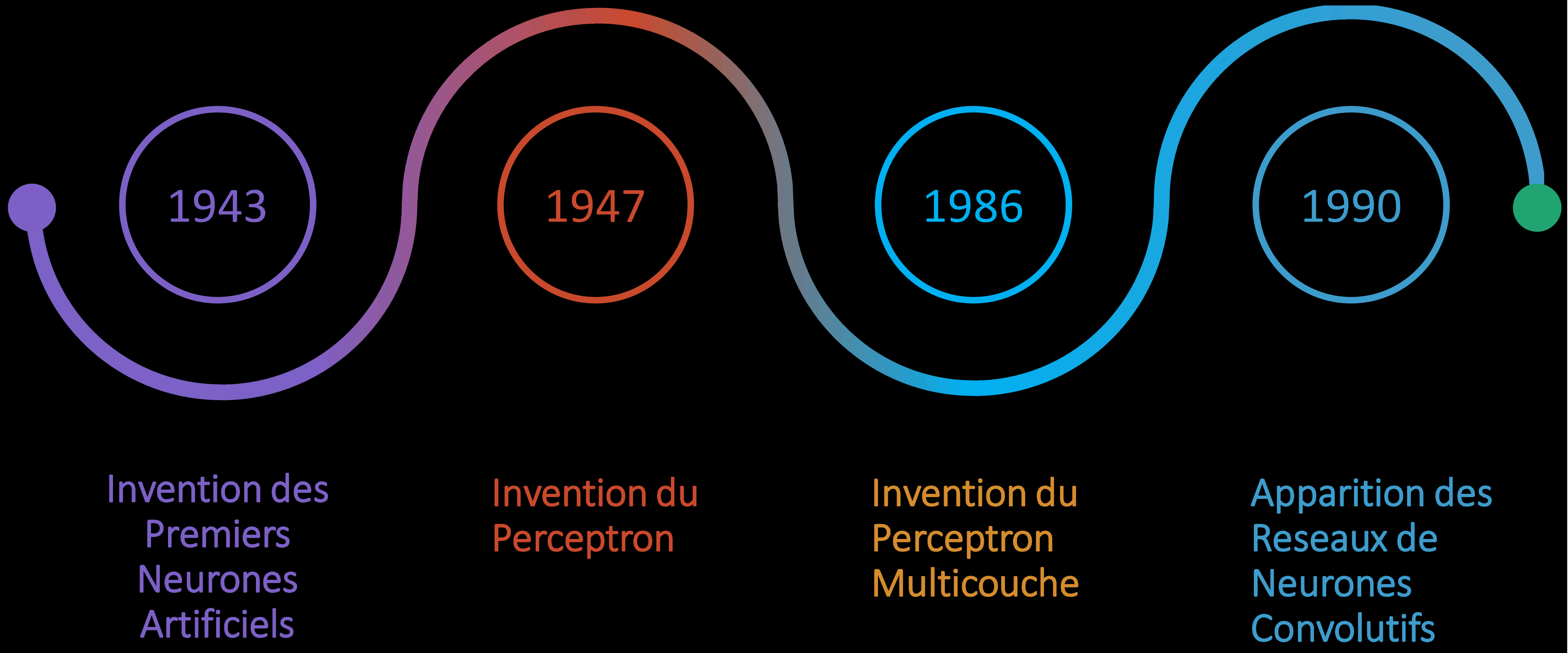
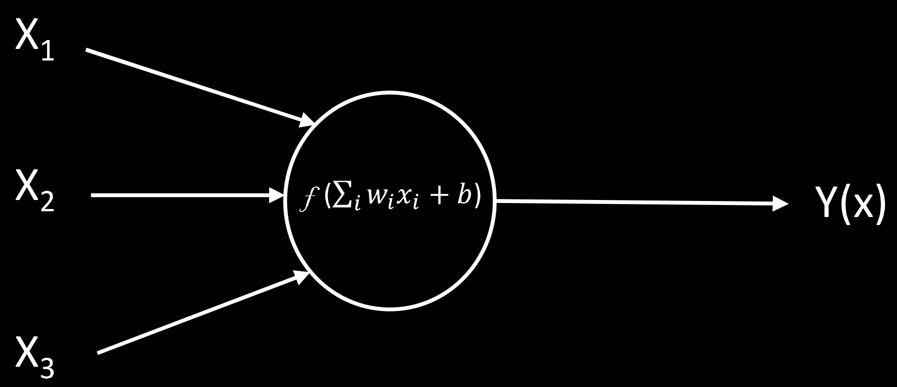
Evolution du Deep Learning



Le Perceptron

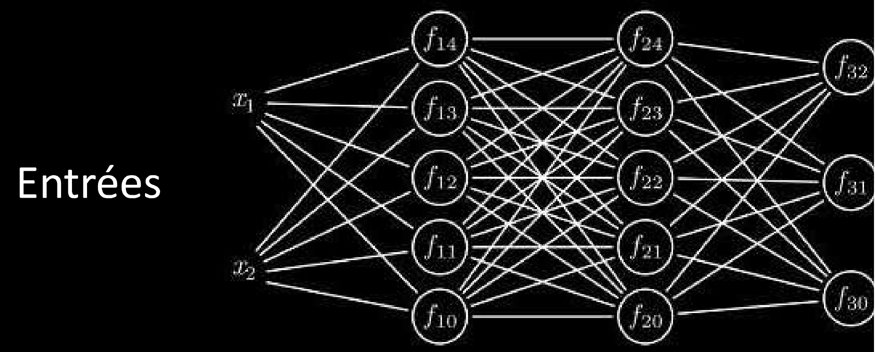
1. Neurone Formel et le perceptron Multicouche.
2. Fonction d’activation, en particulier la function sigmoide (Logistic
3. Fonction Cout et minimisation de l’erreur.
4. La Descente du gradient.
5. Algorithme de la descente de gradient.

Neurone formel



X_i : Entrées
f : Fonction d'activation
Y : Sortie

Perceptron multicouche



f_{ij} : Fonction d'activation

Fonction d'activation

Sigmoid

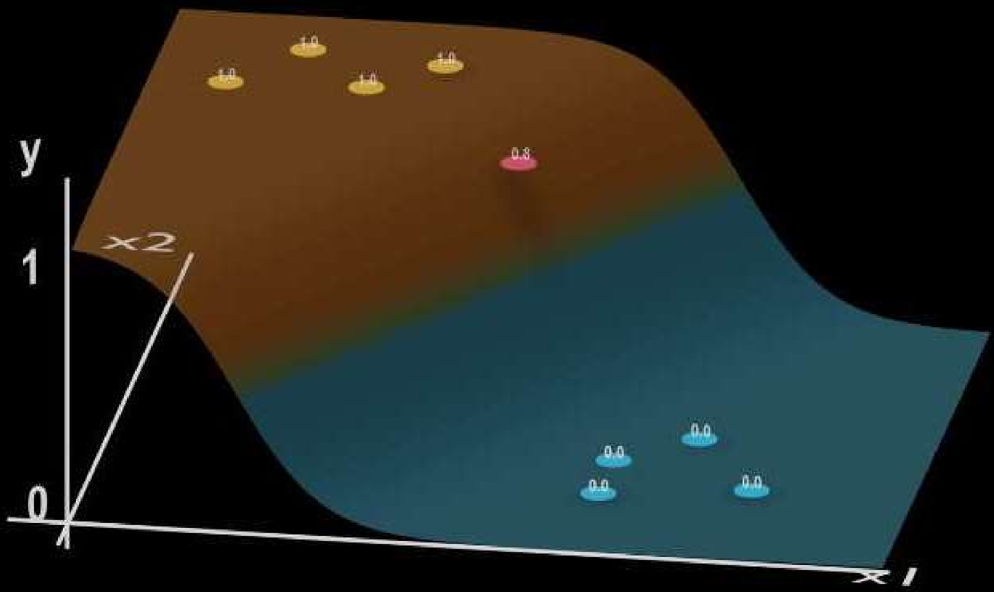
$$f(x) = \frac{1}{1 + e^{-x}}$$

Relu

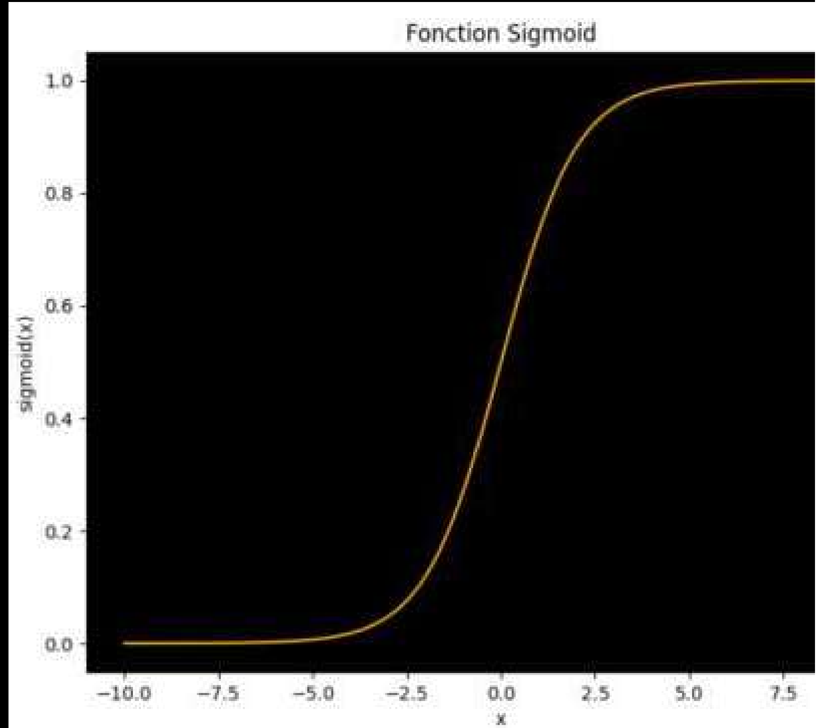
$$f(x) = \max(0, x)$$

Heaviside

$$H(x) = \begin{cases} 1, Si\ x > 0 \\ \frac{1}{2}, Si\ x = 0 \\ 0, Si\ x < 0 \end{cases}$$



Source : Machine Learnia



Minimisation de l'erreur

Pour minimiser les erreurs de notre modèle, nous utiliserons la fonction de coût ***Log Loss:***

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

$$\begin{cases} a_i : la\ probabilité\ d'etre\ dans\ la\ cl \\ \qquad\qquad la\ sortie\ i \\ y_i : la\ classe\ de\ l'echantillon \end{cases}$$

Cette formule est dérivée de la définition statistique de ***Log-vraisemblance*** :

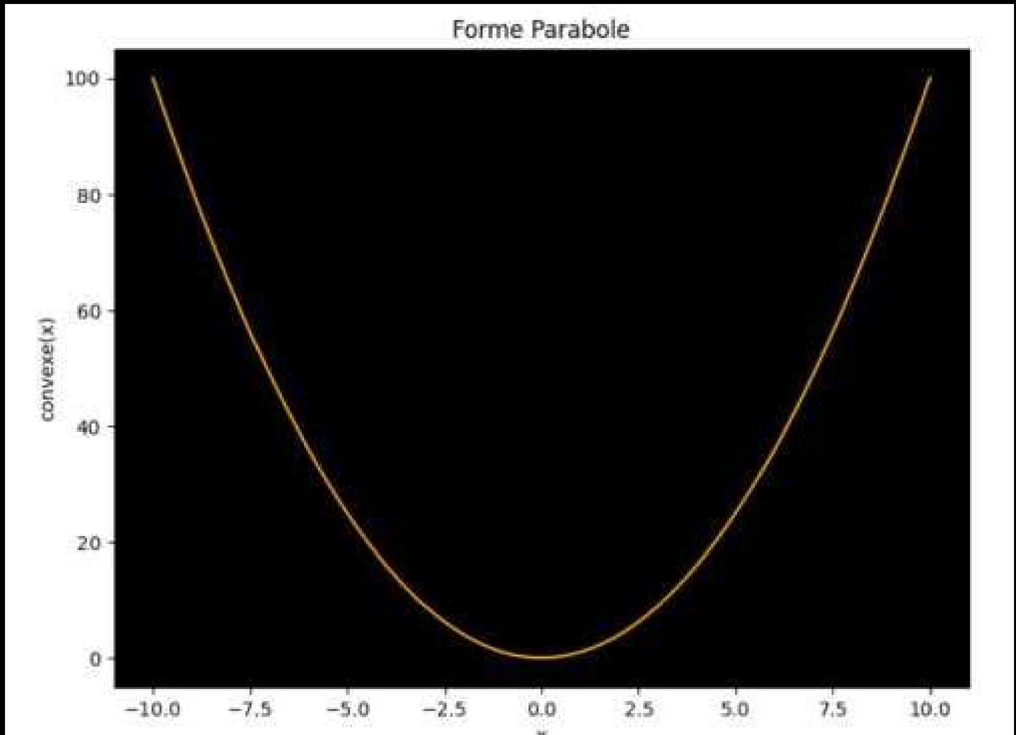
$$\boxed{\log(\prod_{i=1}^{m} P(X = x_i)) = \log(\prod_{i=1}^{m} a_i^{x_i} * (1 - a_i)^{1-x_i})}$$

$$avec\ x_i \in \{0,1\}$$

En effet, pour maximiser la vraisemblance, nous minimisons la fonction de coût car il existe de nombreux algorithmes d'optimisation pour minimiser les fonctions.

L’ algorithme de la descente du gradient

Il s’agit d’un algorithme de minimisation qui consiste à ajuster les parametres W et le biais b afin de minimiser
coût. On calcule ainsi le gradient de la fonction coût



- Si $\frac{\partial F}{\partial x} > 0$, la fonction F augmente lorsque x augme
- Sinon, la fonction F diminue lorsque x augmente

Dans notre cas, nous chercherons à atteindre un
minimum global pour la fonction de coût. Nous dé
donc la fonction par rapport aux paramètres, et n
augmenterons la valeur de ces paramètres en for
signe de la dérivée jusqu'à atteindre le minimum.

La formule conçue pour ce travail est :

$$\boxed{W_{i+1} = W_i - \mu * \frac{\partial L}{\partial W_i}}$$

En resumé

Neurone Formel

$$z = \sum_{i=1}^{m} w_i x_i + w_0$$

$$y(sortie) = \begin{cases} 0\,, Si\ z < 0 \\ 1, Si\ z \geq 0 \end{cases}$$

Perceptron

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

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

$$W_{i+1} = W_i - \mu * \frac{\partial L}{\partial W_i}$$

Vectorisation des équations pour des donné larges

1) Vectorisation du modele:

$$\begin{cases} X = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \\ W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \\ B = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \end{cases} \Rightarrow \quad Z = \begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(m)} \end{bmatrix} = \begin{bmatrix} w_1 * x_1^{(1)} + w_2 * x_2^{(1)} + \cdots + w_n * x_n^{(1)} + b \\ \vdots \\ w_1 * x_1^{(m)} + w_2 * x_2^{(m)} + \cdots + w_n * x_n^{(m)} + b \end{bmatrix} =$$

2) Vectorisation de la fonction d'activation:

$$A = \begin{bmatrix} \sigma(z^{(1)}) \\ \vdots \\ \sigma(z^{(m)}) \end{bmatrix} = \sigma\left(\begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(m)} \end{bmatrix}\right) = \sigma(Z)$$

$$avec : \sigma(z) = \frac{1}{1 + e^{-z}} \quad (fonction\ log$$

3) Vectorisation de la fonction cout:

Rappelons que :

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

$$L = -\frac{1}{m} * Somme\_des\_lignes(y \underbrace{*}_{} \log(A) + (1 - y) * \log(1 - A)) \quad , avec \begin{cases} y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \\ A = \begin{bmatrix} \sigma(z^{(1)}) \\ \vdots \\ \sigma(z^{(m)}) \end{bmatrix} \end{cases}$$

*Produit terme à terme

4) Vectorisation de l'algorithme de la descente de gradient:

$$W = W - \mu * \frac{\partial L}{\partial W}$$

$$B = B - \mu * \frac{\partial L}{\partial B}$$

$$\begin{cases} W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \\ B = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \\ \frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix} \\ \frac{\partial L}{\partial B} = \begin{bmatrix} \frac{\partial L}{\partial b} \\ \vdots \\ \frac{\partial L}{\partial b} \end{bmatrix} \end{cases}$$

Calcule des gradients

1) Calcule de $\frac{\partial L}{\partial w_i}$:

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

alors $L = -\frac{1}{m} * \sum_{i=1}^{m} y_i * \log\left(\frac{1}{1+e^{-z_i}}\right) + (1 - y_i) * \log\left(1 - \frac{1}{1+e^{-z_i}}\right)$

alors $L = -\frac{1}{m} * \sum_{i=1}^{m} (- y_i * \log(1 + e^{-z_i})) + (1 - y_i) * (-\log(1 + e^{-z_i}) - z_i)$

Donc $L = -\frac{1}{m} * \sum_{i=1}^{m} - \log(e^{z_i} + 1) + y_i z_i$

Finalement : $\boxed{\frac{\partial L}{\partial w_j} = -\frac{1}{m} * \sum_{i=1}^{m} (y_i - a_i) \times x_j^{(i)}}$

2) Calcule de $\frac{\partial L}{\partial b}$:

De façon analogue : $\boxed{\frac{\partial L}{\partial b} = -\frac{1}{m} * \sum_{i=1}^{m} (y_i - a_i)}$

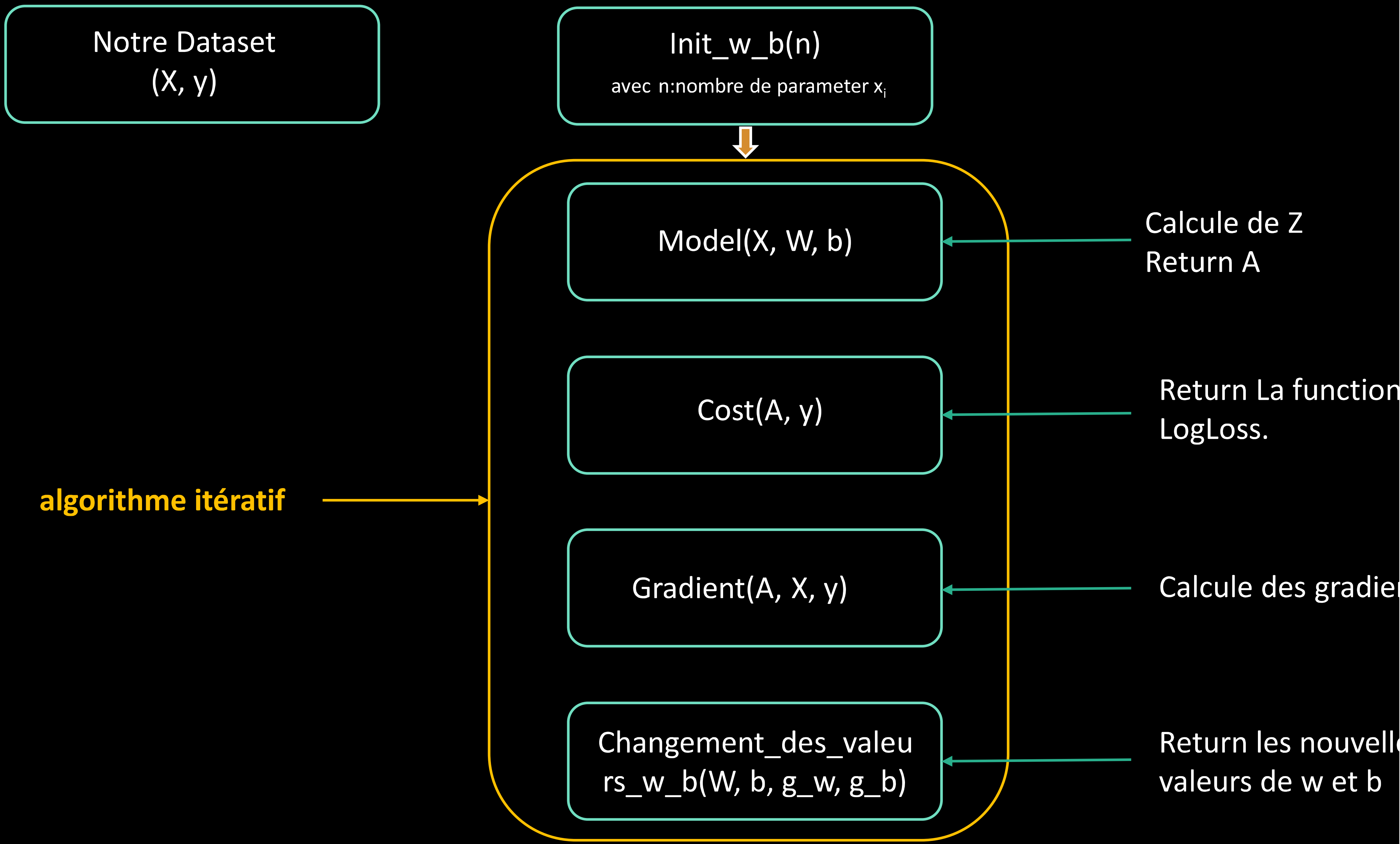
Ainsi sous une forme vectorisée :

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} (y_i - a_i) \times \boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{i})} \\ \vdots \\ \sum_{i=1}^{m} (y_i - a_i) \times \boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{i})} \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{1})} & \dots & \boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{m})} \\ \vdots & \vdots & \vdots \\ \boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{1})} & \dots & \boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{m})} \end{bmatrix} . \left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \right) = -\frac{1}{m} X^T . (y - A)$$

De meme:

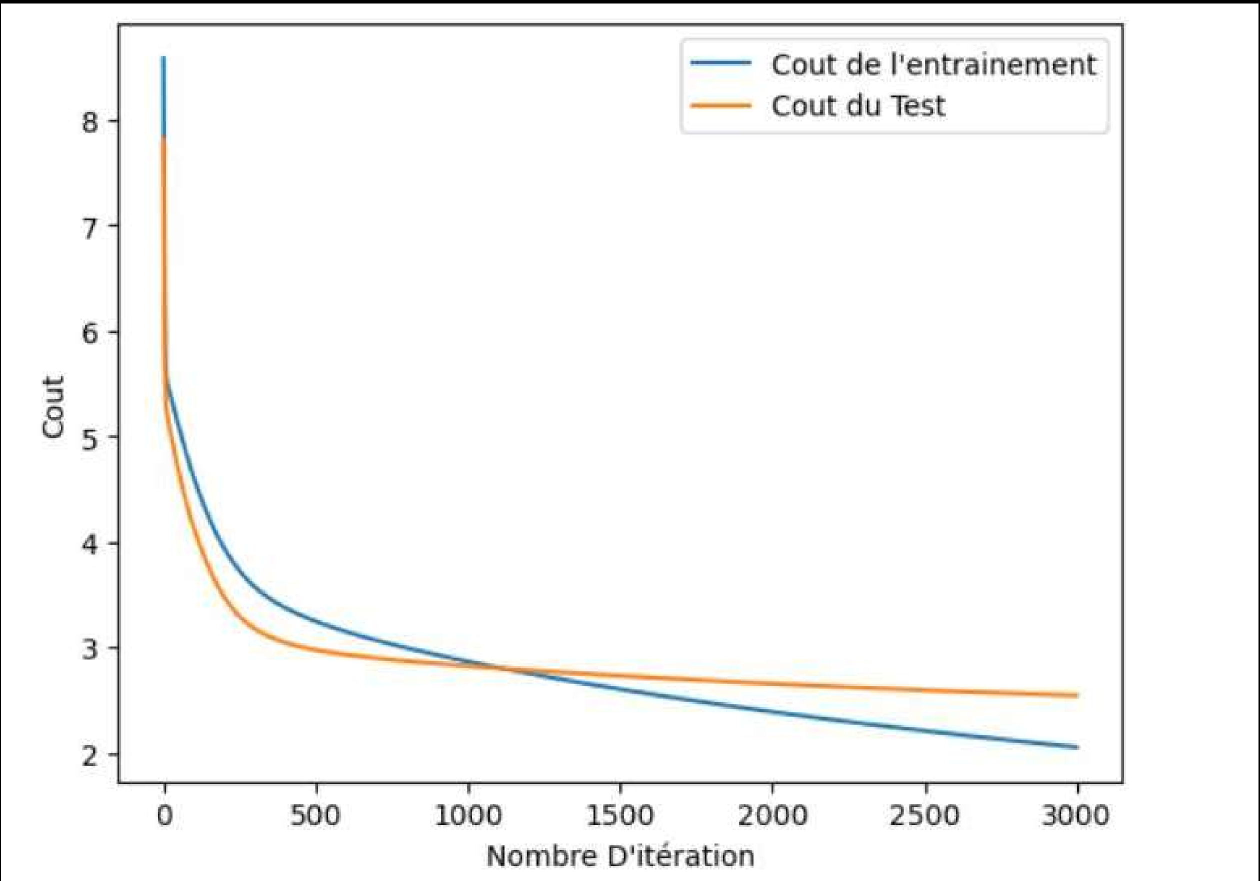
$$\frac{\partial L}{\partial B} = \begin{bmatrix} \frac{\partial L}{\partial b} \\ \vdots \\ \frac{\partial L}{\partial b} \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} (y_i - a_i) \\ \vdots \\ \sum_{i=1}^{m} (y_i - a_i) \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} \sum (y - A) \\ \vdots \\ \sum (y - A) \end{bmatrix}$$

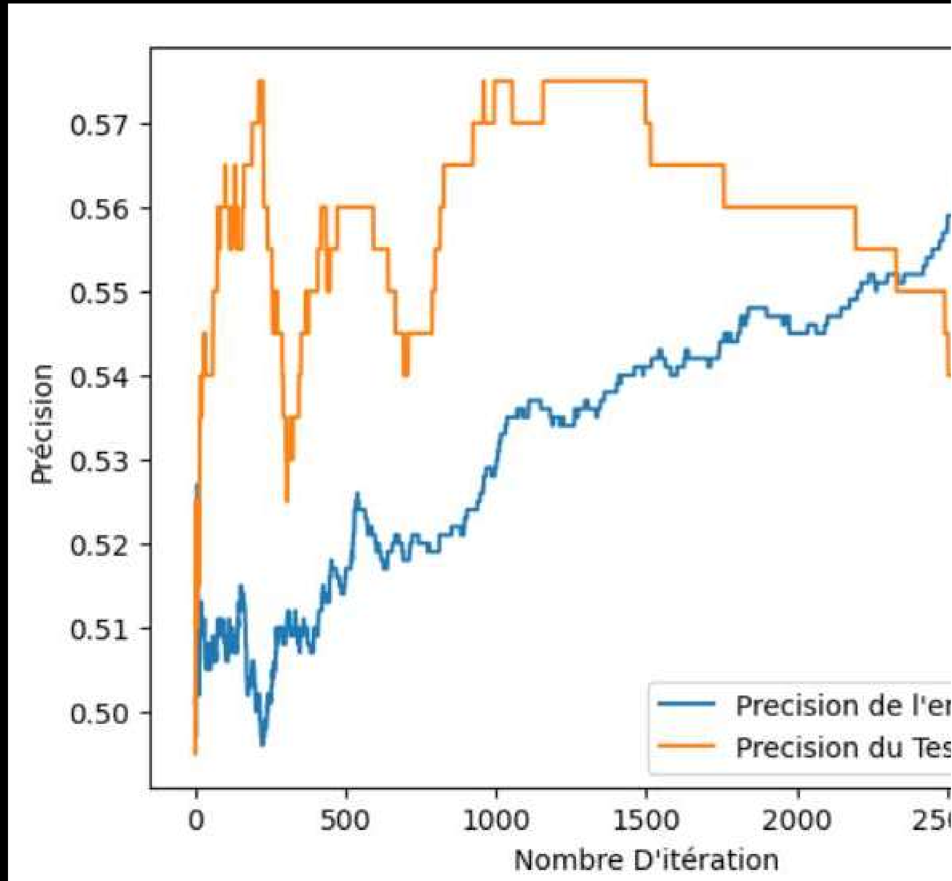
Programmation du neurone artificiel



Application : Chat ou Chien ?

Résultat:

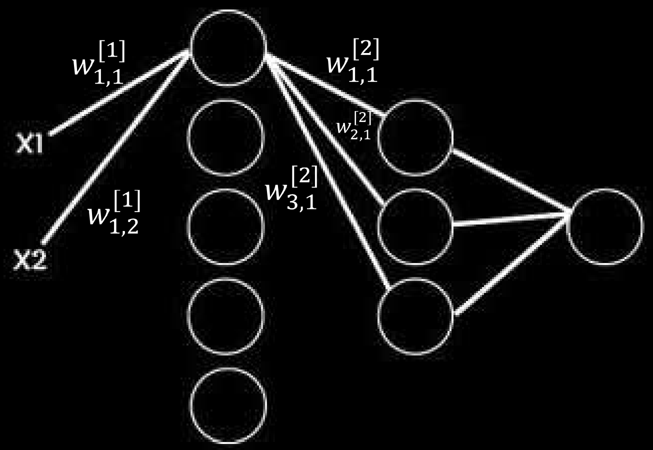




Vers les réseaux de neurones

L’insuffisance du perceptron tout seul nous oblige à travailler avec plusieurs neurones connectés entre eux pour faire des taches plus complexes et plus précis :

1) Forward Propagation:



Notation:

$W_{i,j}^{[\alpha]}$: *poid associé au neurone i et provenant de l'entrée j et α le numero de la couche*

Véctorisation des couches d’un réseau de neurone:

Fonction d’agrégation:

$$\begin{cases} Z^{[1]} = W^{[1]}.X + B^{[1]} \\ Z^{[\alpha]} = W^{[\alpha]}.A^{[\alpha-1]} + B^{[\alpha]} \end{cases}$$

Fonction d’activation:

$$A^{[\alpha]} = \frac{1}{1+ e^{-Z^{[\alpha]}}}$$

2) Back Propagation:

On procède de la meme façon que pour un perceptron, sauf qu'ici pour actualiser le poids de connexion $w_{j,q}^{h}$ du neurone j de la couche h-1 vers le neurone q de la couche h, nous devons calculer $\frac{\partial L}{\partial w_{j,q}^{h}}$. Pour ce faire, nous allons appliquer le théoreme de dérivation des fonctions composées (Règle de la chaine).

D'après les calules faits sur la fonction cout d'un perceptron, on obtient :

Cas de C = 2:

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}^{[2]}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{W}^{[2]}}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{B}^{[2]}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{B}^{[2]}}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}^{[1]}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{A}^{[1]}} \times \frac{\partial \boldsymbol{A}^{[1]}}{\partial \boldsymbol{Z}^{[1]}} \times \frac{\partial \boldsymbol{Z}^{[1]}}{\partial \boldsymbol{W}^{[1]}}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{B}^{[1]}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{A}^{[1]}} \times \frac{\partial \boldsymbol{A}^{[1]}}{\partial \boldsymbol{Z}^{[1]}} \times \frac{\partial \boldsymbol{Z}^{[1]}}{\partial \boldsymbol{B}^{[1]}}$$

Avec:

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}} = \sum \frac{-\boldsymbol{y}}{\boldsymbol{A}} + \frac{\boldsymbol{1} - \boldsymbol{y}}{\boldsymbol{A}}$$

$$\frac{\partial A}{\partial Z} = A \times (1 - A)$$

$$\frac{\partial Z}{\partial W} = A$$

On constate alors une répetition de calcule, on utilisera alors la <u>mémoïsation</u> pour optimiser nos calcules.

Notation:

[C] : numé
couche.

Notation N
pour les ve

On pose :

$$base^{[2]} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}}$$

$$base^{[1]} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{A}^{[2]}} \times \frac{\partial \boldsymbol{A}^{[2]}}{\partial \boldsymbol{Z}^{[2]}} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{A}^{[1]}} \times \frac{\partial \boldsymbol{A}^{[1]}}{\partial \boldsymbol{Z}^{[1]}} = \boldsymbol{base}^{[2]} \times \frac{\partial \boldsymbol{Z}^{[2]}}{\partial \boldsymbol{A}^{[1]}} \times \frac{\partial \boldsymbol{A}^{[1]}}{\partial \boldsymbol{Z}^{[1]}} = \boldsymbol{W}^{[2]^{T}} \times \boldsymbol{base}^{[2]} \times \boldsymbol{A}^{[1]} \times (\boldsymbol{1} -$$

Cas général:

En prenant en considération la technique de mémoïsation, on obtient les équations suivantes pour une c

$$\frac{\partial L}{\partial W^{[C]}} = \frac{1}{m} \times base^{[C]} . A^{[C-1]^{T}}$$: la somme est inclue dans le produit matricielle

$$\frac{\partial L}{\partial B^{[C]}} = \frac{1}{m} \times base^{[C]}$$

$$base^{[C-1]} = W^{[C]^{T}} \times base^{[C]} \times A^{[C-1]} \times \left(1 - A^{[C-1]}\right)$$: technique de mémoïsation.

Convolution

1)Operation de convolution: (Padding = ‘same’ ou ‘valid’, Stride = (x, y), kernel = taille du filtre)

0	0	0	0	0
0	2	50	20	0
0	20	18	20	0
0	20	20	20	0

Portion d’une image

*

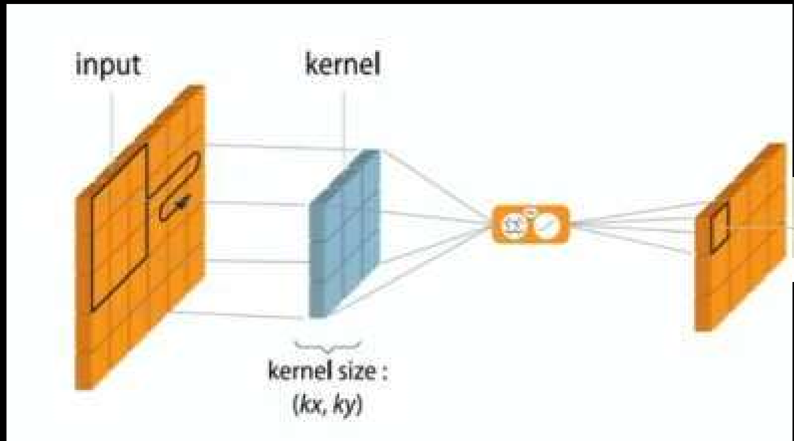
-1	2	2
2	6	2
2	2	-1

Filtre (Kernel)

=

134		

2)Neurone de Convolution:



$y = \sigma(\text{input} \otimes \text{kernel} + b)$

Source : youtube, @CNRS - Formation FIDLE

Maxpooling

1)Operation de maxpooling: (Stride = (x, y), kernel = taille du filtre)

0	0	0	0
0	2	50	20
0	20	18	20
0	20	20	20

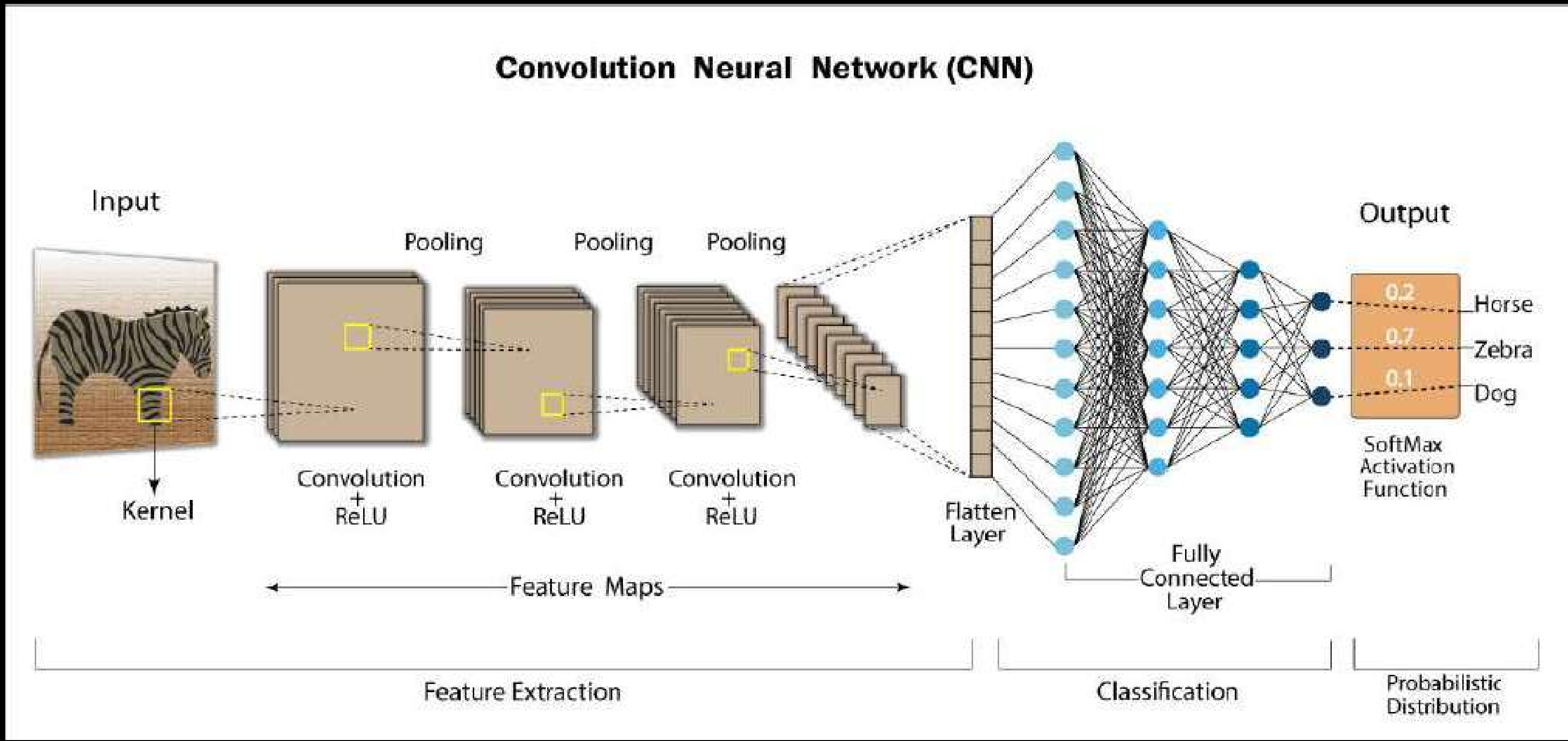
$\xrightarrow{\text{2x2 maxpooling}}$

2	50
20	20

2)Neurone de MaxPooling :

$$\vec{X} \longrightarrow \boxed{maxpooling(\vec{X})} \longrightarrow \vec{Y}$$

Reconnaissance d’image



Structure type d’un réseau de classification d’images
Source : https://www.analyticsvidhya.com/

Résultat d’entrainement d’un reseau de neurone convolutif sur la dataset EMNIST