Deep learning par la pratique

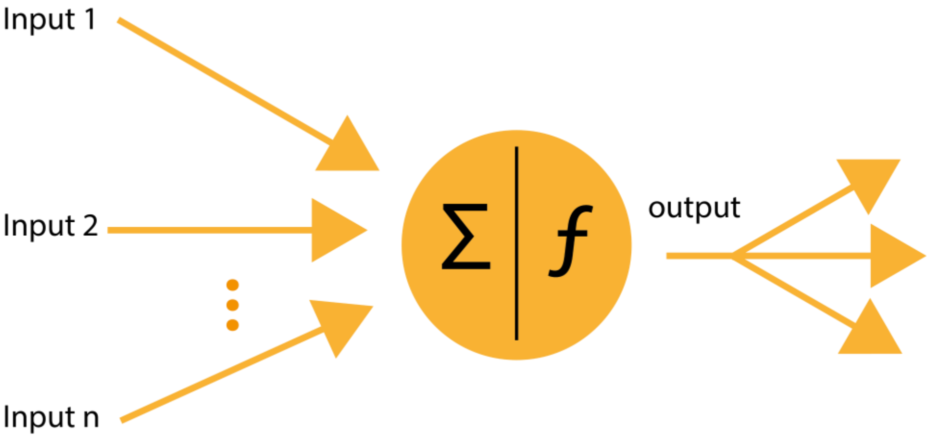
Leçon 1 : Votre premier réseau

Présenté par **Morgan Gautherot**

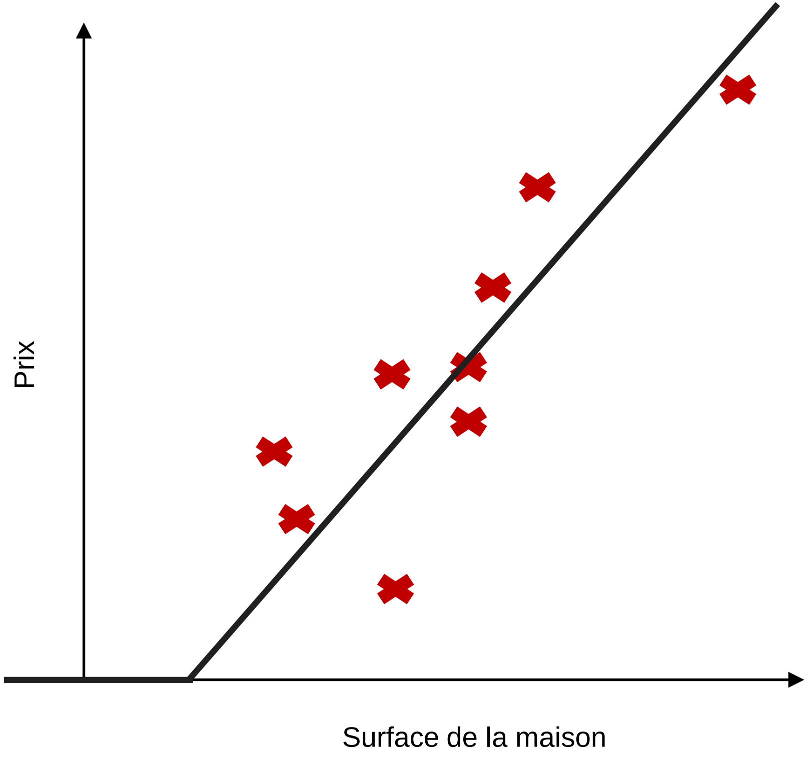
Le neurone artificiel

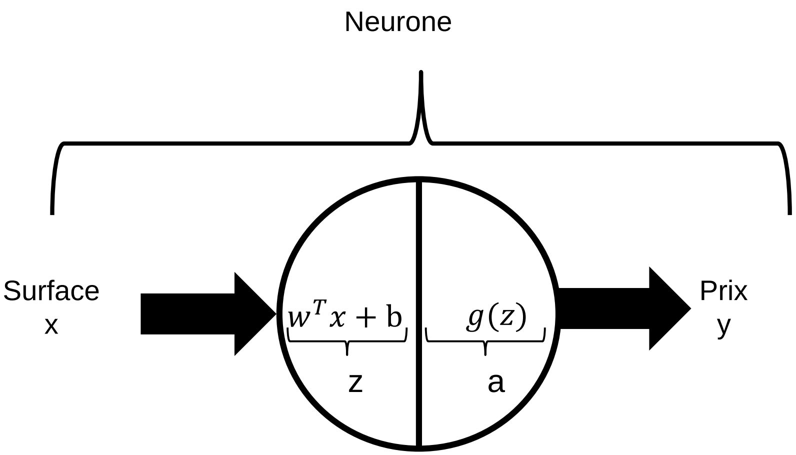
Neurone biologique

Neurone artificiel



Qu'est-ce qu'un réseau neuronal ?

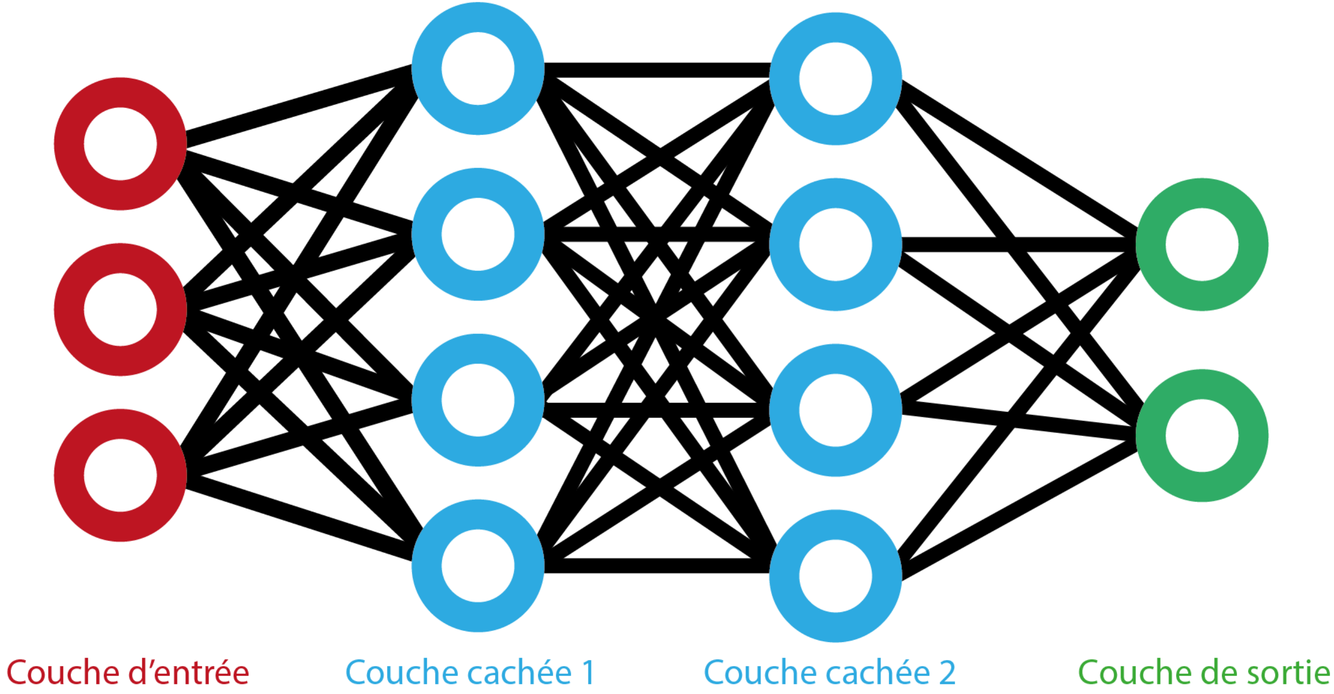




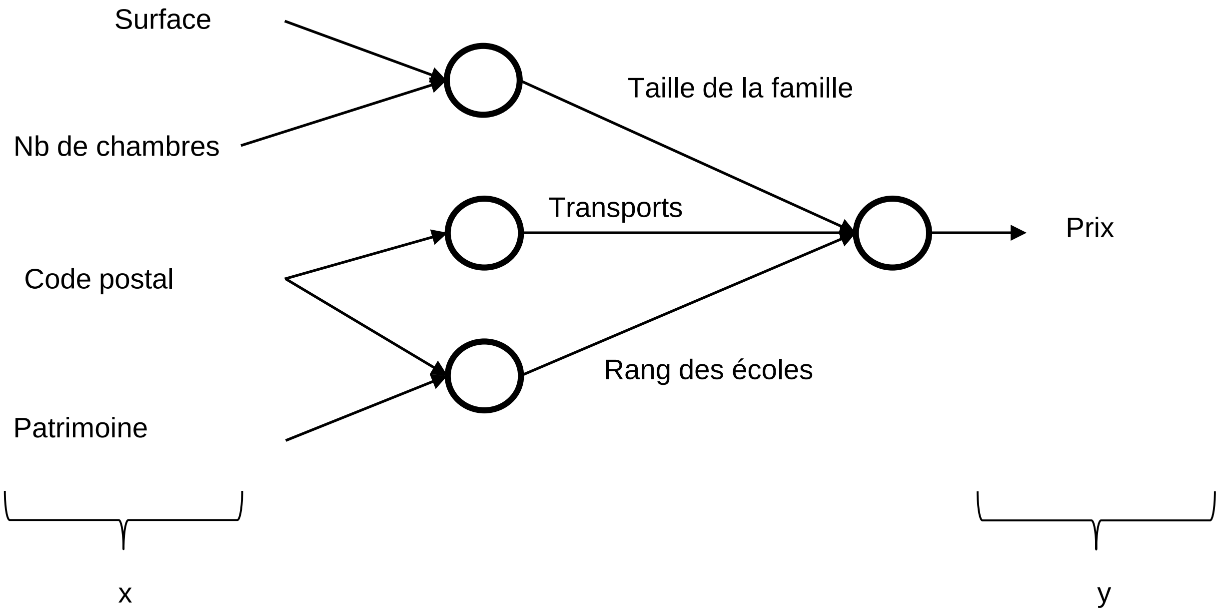
Un réseau neuronal est construit
en connectant de nombreux neurones

Fonction d’activation : $\mathrm{g}(z) = \max(0, z)$

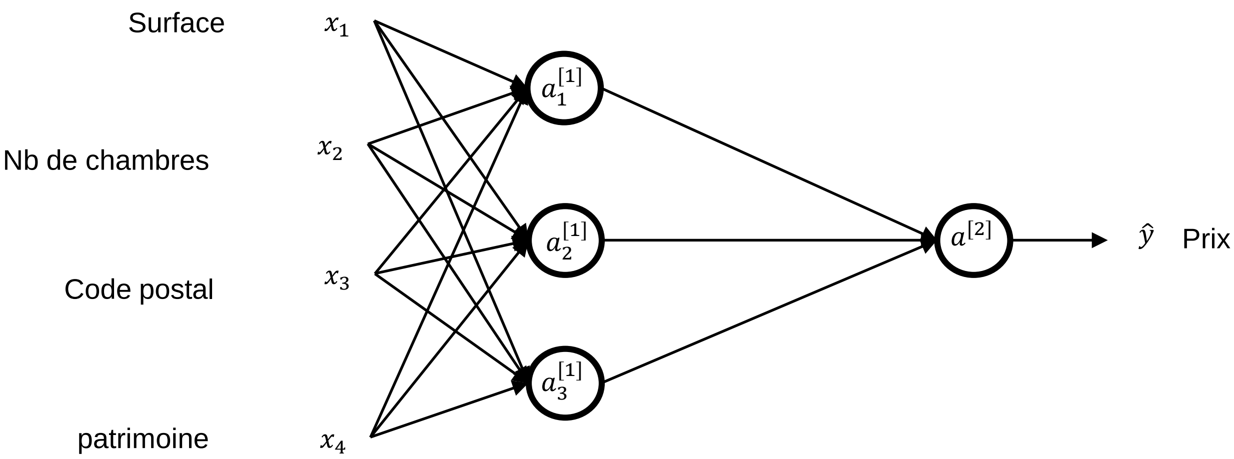
Réseau neuronal



Le feature engineering



Feature engineering automatique

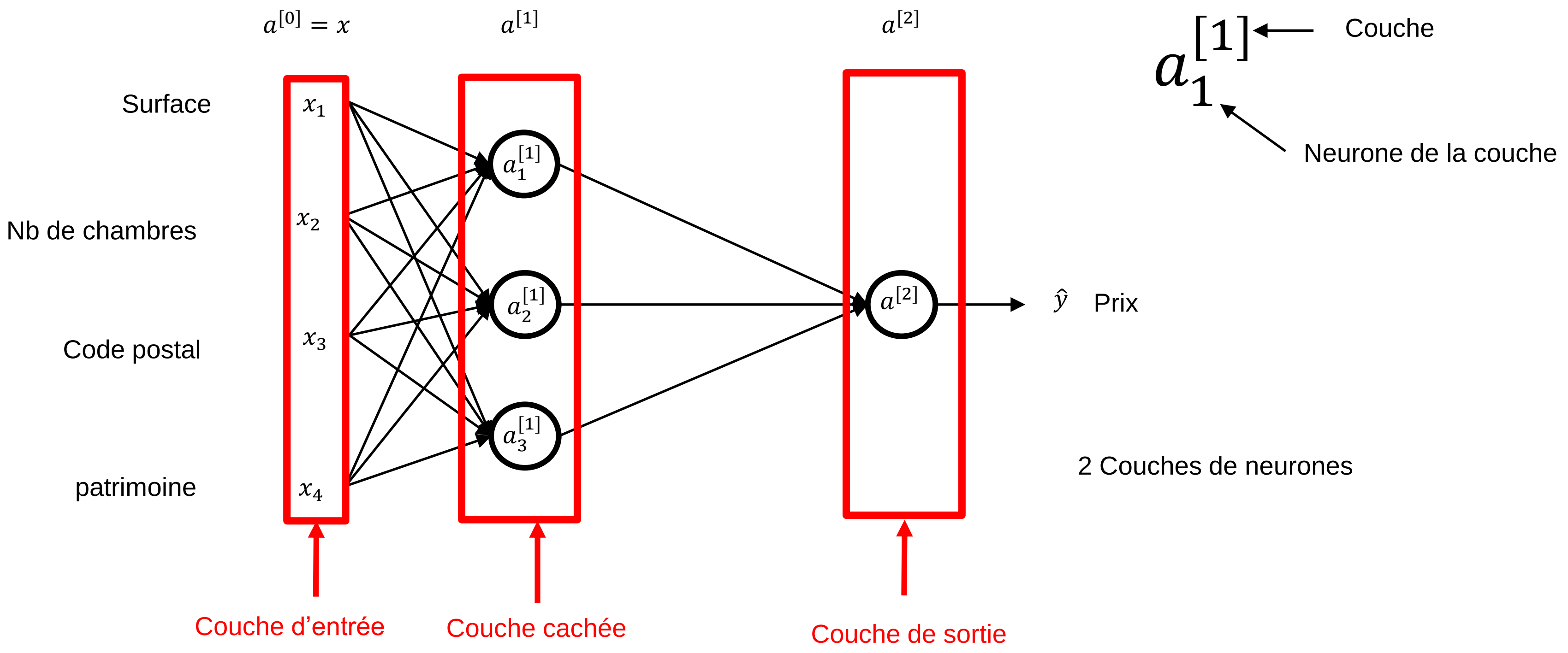


Deep learning par la pratique

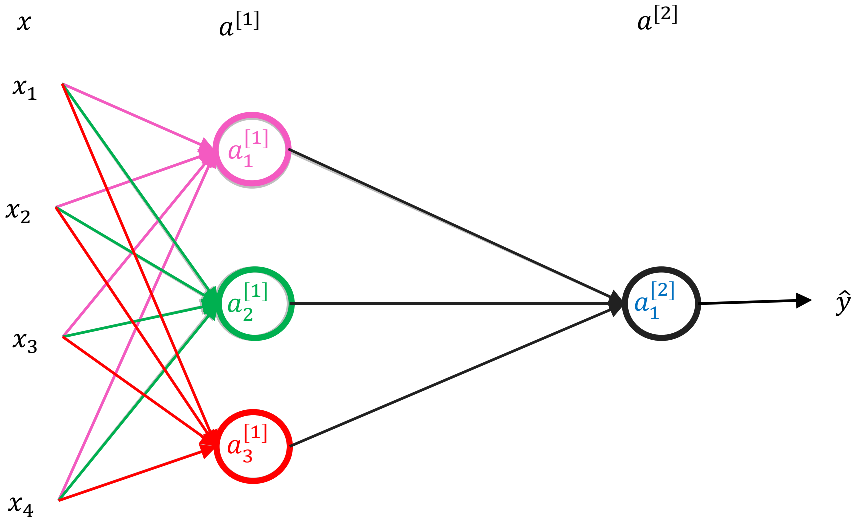
Leçon 2 : Représentation mathématique

Présenté par **Morgan Gautherot**

Un peu de notations



Et les mathématiques dans tout ça ?



$z_1^{[1]} = w_1^{[1]^T} x + b_1^{[1]}$; $a_1^{[1]} = g(z_1^{[1]})$

$z_2^{[1]} = w_2^{[1]^T} x + b_2^{[1]}$; $a_2^{[1]} = g(z_2^{[1]})$

$z_3^{[1]} = w_3^{[1]^T} x + b_3^{[1]}$; $a_3^{[1]} = g(z_3^{[1]})$

$z_1^{[2]} = w_1^{[2]^T} a^{[1]} + b_1^{[2]}$; $a_1^{[2]} = g(z_1^{[2]})$

Vectorisation sur un exemple

$$\begin{bmatrix} z_1^{[1]} = w_1^{[1]^T} x + b_1^{[1]} \\ z_2^{[1]} = w_2^{[1]^T} x + b_2^{[1]} \\ z_3^{[1]} = w_3^{[1]^T} x + b_3^{[1]} \end{bmatrix} \quad ; \quad \begin{bmatrix} a_1^{[1]} = g(z_1^{[1]}) \\ a_2^{[1]} = g(z_2^{[1]}) \\ a_3^{[1]} = g(z_3^{[1]}) \end{bmatrix}$$

$$\begin{bmatrix} z_1{}^{[1]} \\ z_2{}^{[1]} \\ z_3{}^{[1]} \end{bmatrix} = \begin{bmatrix} w_{1,1}{}^{[1]} & w_{1,2}{}^{[1]} & w_{1,3}{}^{[1]} & w_{1,4}{}^{[1]} \\ w_{2,1}{}^{[1]} & w_{2,2}{}^{[1]} & w_{2,3}{}^{[1]} & w_{2,4}{}^{[1]} \\ w_{3,1}{}^{[1]} & w_{3,2}{}^{[1]} & w_{3,3}{}^{[1]} & w_{3,4}{}^{[1]} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1{}^{[1]} \\ b_2{}^{[1]} \\ b_3{}^{[1]} \end{bmatrix} \quad ; \quad \begin{bmatrix} a_1{}^{[1]} \\ a_2{}^{[1]} \\ a_3{}^{[1]} \end{bmatrix} = \mathrm{g}\left(\begin{bmatrix} z_1{}^{[1]} \\ z_2{}^{[1]} \\ z_3{}^{[1]} \end{bmatrix}\right)$$

$$\underset{(3,1)}{z^{[1]}} = \underset{(3,4)}{W^{[1]^T}} \underset{(4,1)}{\mathrm{x}} + \underset{(3,1)}{b^{[1]}} \quad ; \quad a^{[1]} = g(z^{[1]})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,3)}{w^{[2]^T}} \underset{(3,1)}{a^{[1]}} + \underset{(1,1)}{b^{[2]}} \quad ; \quad a^{[2]} = g(z^{[2]})$$

Vectorisation sur plusieurs exemples

~~for i = 0 to m :~~

$$\overset{(n_1,1)}{a^{[1](i)}} = g(\overset{(n_1,\,n_x)}{w^{[1]^T}}\,\overset{(n_x,1)}{x^{(i)}} + \overset{(n_1,1)}{b})$$

$$\overset{(1,1)}{\hat{y}^{(i)}} = a^{[2](i)} = g(\overset{(1,\,n_1)}{w^{[2]^T}}\,\overset{(n_1,1)}{a^{[1](i)}} + \overset{(1,1)}{b})$$

$$\overset{(n_1,m)}{a^{[1]}} = g(\overset{(n_1,\,n_x)}{w^{[1]^T}}\,\overset{(n_x,m)}{X} + \overset{(n_1,1)}{b})$$

$$\overset{(1,m)}{\hat{y}} = a^{[2]} = g(\overset{(1,\,n_1)}{w^{[2]^T}}\,\overset{(n_1,m)}{a^{[1]}} + \overset{(1,1)}{b})$$

$x \in \mathbb{R}^{n_x}$

$\hat{y} \in \mathbb{R}$

$$x^{(i)} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_x} \end{bmatrix}$$

$X \in \mathbb{R}^{(n_x\,,\,m)}$

$\hat{y} \in \mathbb{R}^{m}$

$$X = \begin{bmatrix} {x_1}^{(1)} & {x_1}^{(2)} & {x_1}^{(3)} & \dots & {x_1}^{(m)} \\ {x_2}^{(1)} & {x_2}^{(2)} & {x_2}^{(3)} & \dots & {x_2}^{(m)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ {x_{n_x}}^{(1)} & {x_{n_x}}^{(2)} & {x_{n_x}}^{(3)} & \dots & {x_{n_x}}^{(m)} \end{bmatrix}$$

Deep learning par la pratique

Leçon 3 : Fonction d'activation

Présenté par **Morgan Gautherot**

Un modèle sans fonction d'activation

$$z^{[1]} = w^{[1]^T} x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = w^{[2]^T} a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]})$$

$$z^{[1]} = w^{[1]^T} \mathrm{x} + b^{[1]}$$

$$z^{[2]} = w^{[2]^T} z^{[1]} + b^{[2]}$$

$$z^{[2]} = w^{[2]^T} (w^{[1]^T} x + b^{[1]}) + b^{[2]}$$

$$z^{[2]} = \underbrace{w^{[2]^T} w^{[1]^T}}_{\mathrm{W}} x + \underbrace{w^{[2]^T} b^{[1]} + b^{[2]}}_{\mathrm{B}}$$

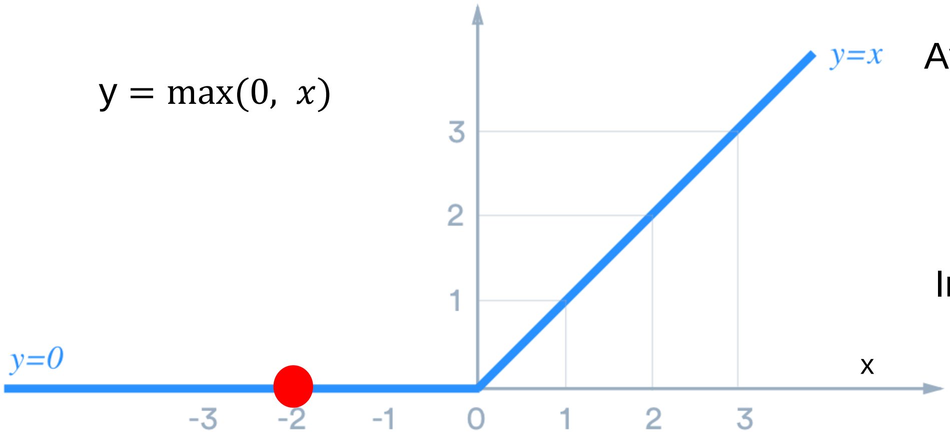
W= $w^{[2]^T} w^{[1]^T}$

B=$w^{[2]^T} b^{[1]} + b^{[2]}$

Fonction Linéaire $z^{[2]} = \mathrm{W}\, x + \mathrm{B}$

Rectified Linear Unit (ReLU)

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



Avantages :

ReLU ne sature pas pour les valeurs positives

ReLU est assez rapide à calculer

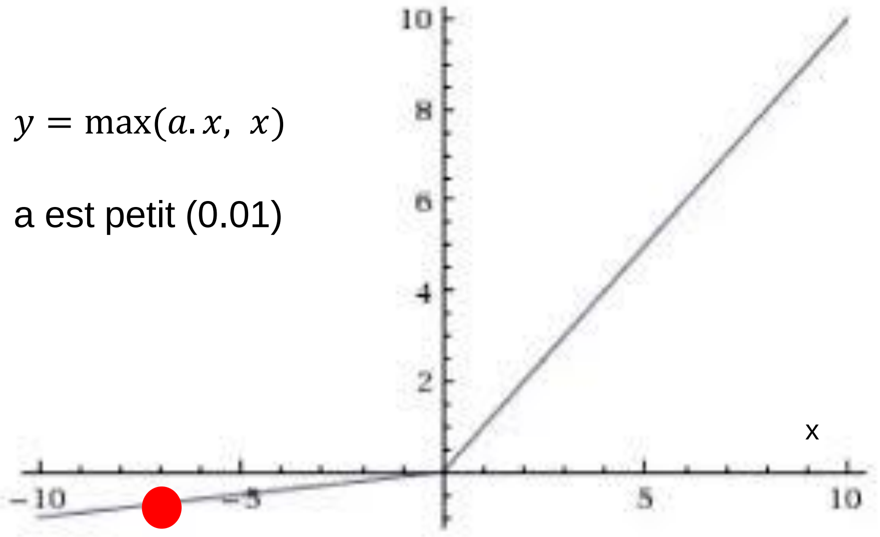
Inconvénients :

ReLU souffre d'un problème connu sous le nom de "dying ReLU".

Leaky ReLU

$y = \max(a.x,\ x)$

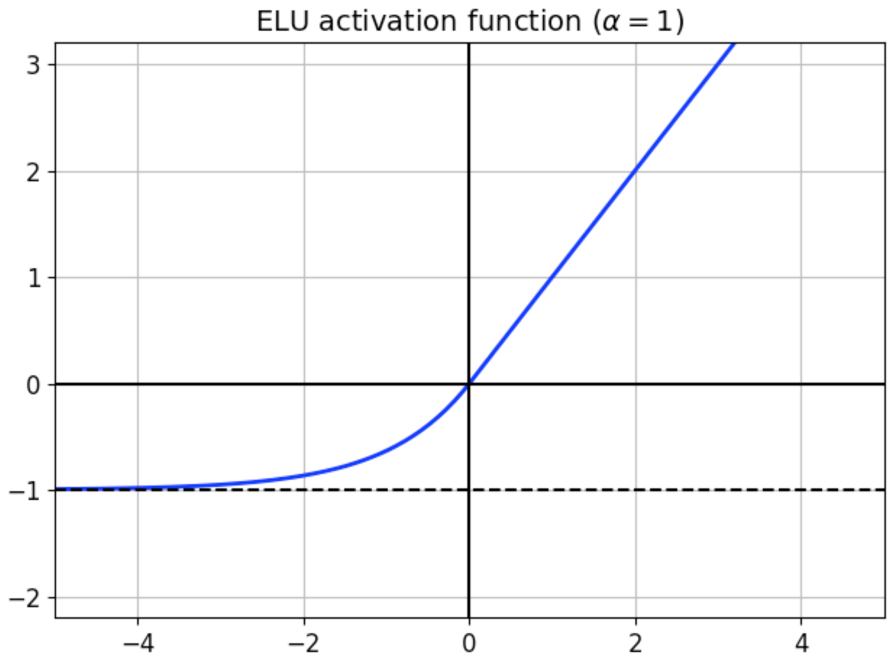
a est petit (0.01)



Variantes :

- Parametric leaky Relu (PReLU), si vous disposez de beaucoup de données.
- Randomized leaky Relu (RReLU), si votre réseau neuronal est surentraîné.

Exponential Linear Unit (ELU)



$$ELU_{\alpha}(z) = \begin{cases} \alpha(\exp(z) - 1) & if\ z < 0 \\ z & if\ z \geq 0 \end{cases}$$

Avantages :

- Il prend des valeurs négatives, ce qui permet au neurone d'avoir une moyenne plus proche de 0
- Il a un gradient non nul pour z < 0, ce qui évite le problème des ‘dying ReLU'.

Inconvénients :

- Il est plus lent à calculer que ReLU.

Résumé

- En général pour la performance :

  ELU > Leaky ReLU > ReLU

- Si vous vous souciez du temps d'exécution :

  Leaky ReLU > ELU

- Par défaut :

  ReLU

Deep learning par la pratique

Leçon 4 : La descente de gradient

Présenté par **Morgan Gautherot**

Descente de gradient pour les réseaux de neurones

$$n_x = n^{[0]}, \quad n^{[1]}, \quad n^{[2]} = 1$$

Paramètres : $\underset{(n^{[1]}, n^{[0]})}{w^{[1]}}, \quad \underset{(n^{[1]}, 1)}{b^{[1]}}, \quad \underset{(n^{[2]}, n^{[1]})}{w^{[2]}}, \quad \underset{(n^{[2]}, 1)}{b^{[2]}}$

$a^{[2]}$

Fonction de coût : $J\left(w^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}\right) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$

Descente de gradient :

Répéter {

Calculer la prédiction $(\hat{y}^{(i)}, i = (1, \dots, m))$

$$\boxed{dw^{[1]}} = \frac{dJ}{dw^{[1]}}, \boxed{db^{[1]}} = \frac{dJ}{db^{[1]}}, \boxed{dw^{[2]}} = \frac{dJ}{dw^{[2]}}, \boxed{db^{[2]}} = \frac{dJ}{db^{[2]}}$$

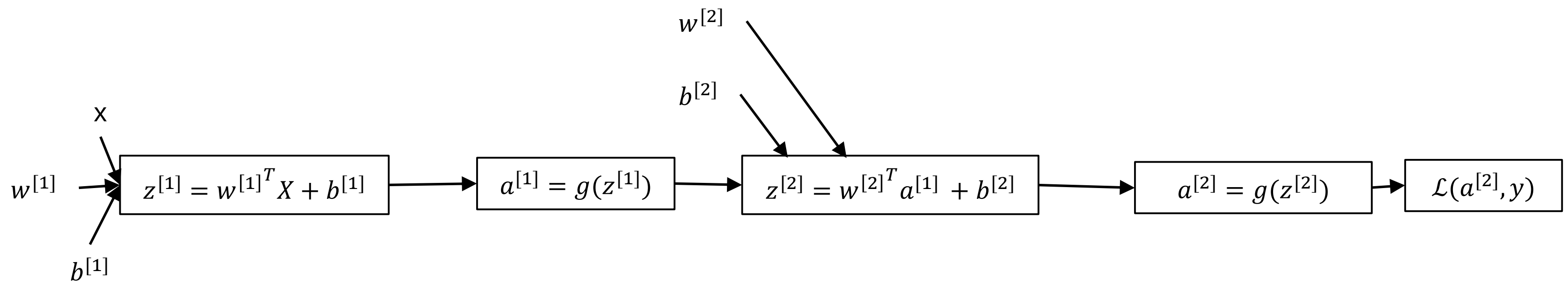
$$w^{[1]} := w^{[1]} - \alpha\, dw^{[1]}$$

$$b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$$

$$w^{[2]} := w^{[2]} - \alpha\, dw^{[2]}$$

$$b^{[2]} := b^{[2]} - \alpha\, db^{[2]}\}$$

Forward propagation



Back propagation

$$\frac{\partial J}{\partial w^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$\frac{\partial J}{\partial b^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial b^{[2]}}$$

$w^{[2]}$

$b^{[2]}$

X

$w^{[1]}$

$z^{[1]} = {w^{[1]}}^T X + b^{[1]}$

$a^{[1]} = g(z^{[1]})$

$z^{[2]} = {w^{[2]}}^T a^{[1]} + b^{[2]}$

$a^{[2]} = g(z^{[2]})$

$\mathcal{L}(a^{[2]}, y)$

$b^{[1]}$

$$\frac{\partial J}{\partial a^{[1]}} = \underbrace{\frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}}$$

$$\frac{\partial J}{\partial a^{[1]}} = \frac{\partial J}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}}$$

$$\frac{\partial J}{\partial z^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot g(z^{[2]})'$$

$$\frac{\partial J}{\partial a^{[2]}} = \left(-y\, log\left(a^{[2]}\right) - (1-y)\log\left(1-a^{[2]}\right)\right)' = -\frac{y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}$$

$$\frac{\partial J}{\partial w^{[1]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

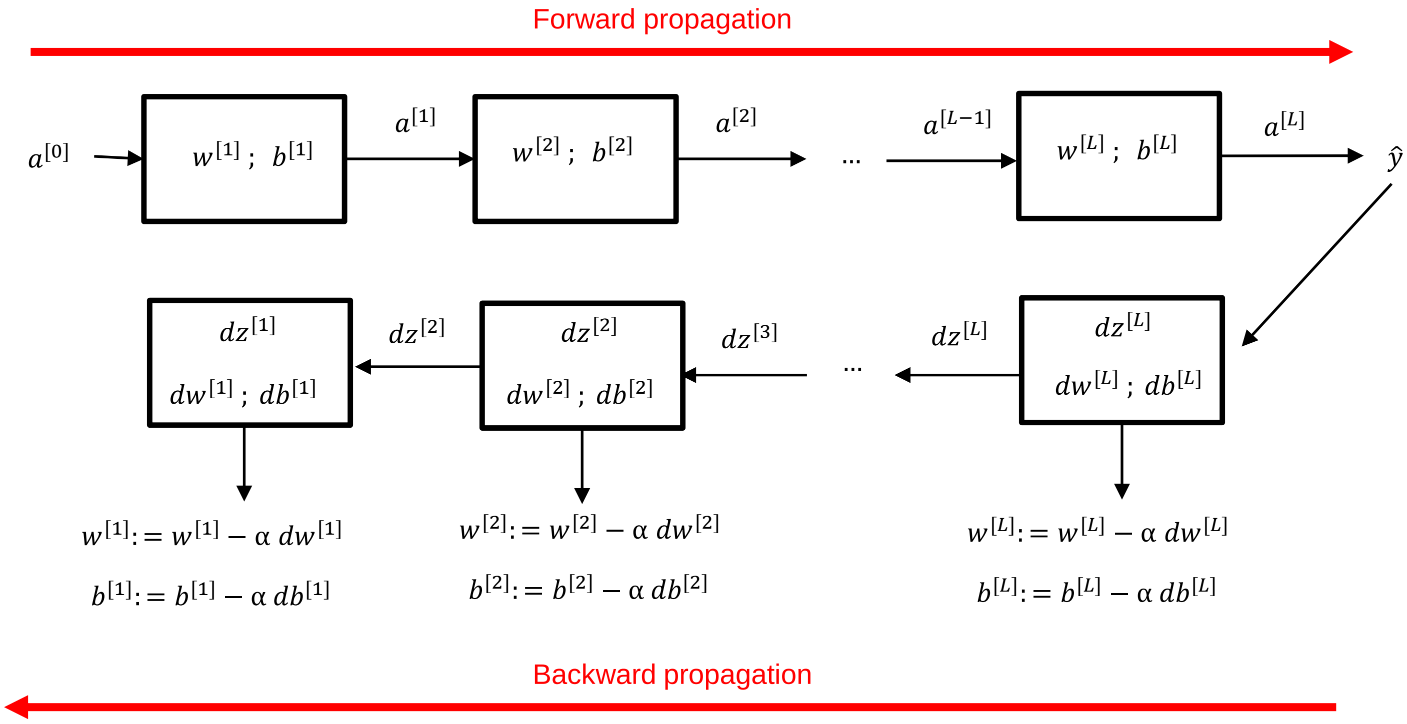
$$\frac{\partial J}{\partial b^{[1]}} = \underbrace{\frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}}} \cdot \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

$$\frac{\partial J}{\partial z^{[1]}} = \underbrace{\frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

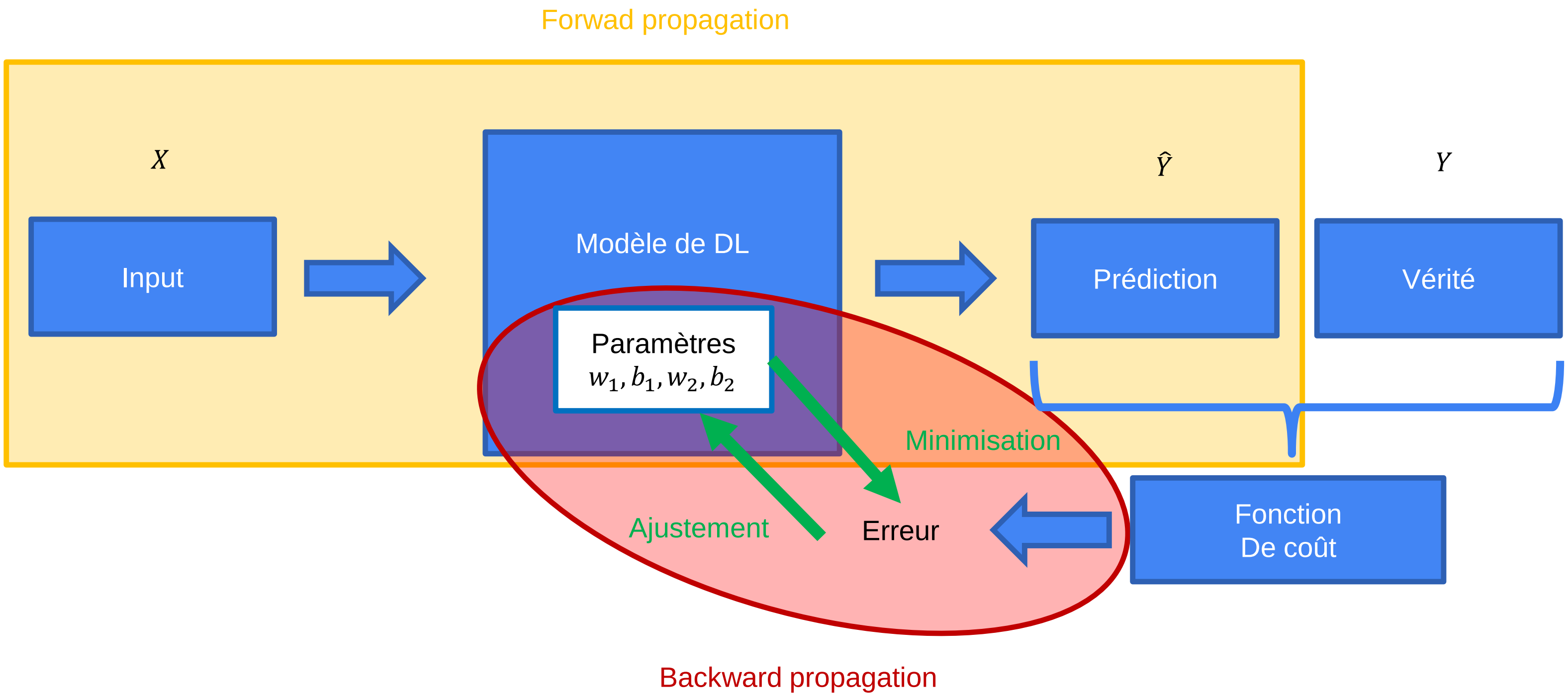
$$= \frac{\partial J}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

Résumé de l'apprentissage profond

$$dt = \frac{\partial J}{\partial t}$$



Entraînement d'un modèle de deep learning

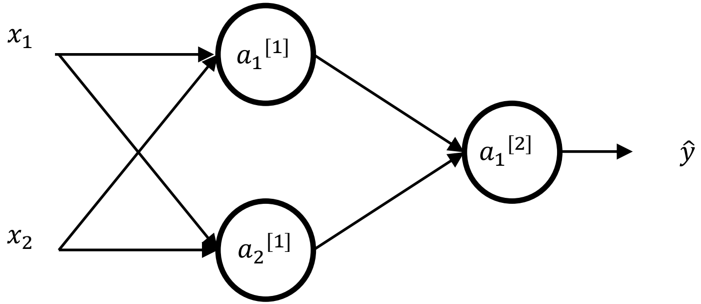


Deep learning par la pratique

Leçon 5 : La descente de gradient

Présenté par **Morgan Gautherot**

Initialiser les poids à zéro ?

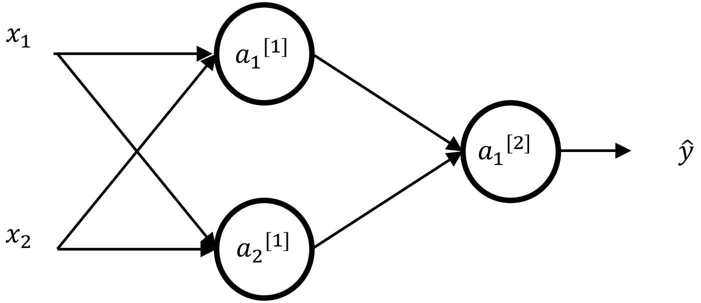


$$w^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w^{[1]} = \begin{bmatrix} p & l \\ p & l \end{bmatrix}$$

$$a_1^{[1]} = a_2^{[1]} \longrightarrow dz_1^{[1]} = dz_2^{[1]} \longrightarrow dw^{[1]} = \begin{bmatrix} u & v \\ u & v \end{bmatrix} \longrightarrow w^{[1]} := w^{[1]} - \alpha dw$$

Initialiser le poids de façon aléatoire



$$w^{[1]} = nombre\_aléatoire.\overbrace{0.01}^{\text{Petit nombre}}$$

$$b^{[1]} = zéro\_ou\_aléatoire$$

$$w^{[2]} = nombre\_aléatoire.\overbrace{0.01}^{\text{Petit nombre}}$$

$$b^{[2]} = zéro\_ou\_aléatoire$$

Deep learning par la pratique

Leçon 6 : Pourquoi apprentissage profond ?

Présenté par **Morgan Gautherot**

Pourquoi des représentations profondes ?

Pour approximer une fonction, une architecture de réseau neuronal plus profonde nécessite moins de paramètres qu'une architecture moins profonde.

$$y = x_1 XOR\ x_2\ XOR\ x_3\ XOR\ x_4\ XOR\ x_5\ XOR\ x_6\ XOR\ \ldots$$