# Machine Learning

## Réseaux de neurones artificiels

**Mustapha El Ossmani**

Ecole Nationale Supérieure des Arts et Métiers de Meknès
Université Moulay Ismail
Filière GI-IADS

2023-2024

# Sommaire

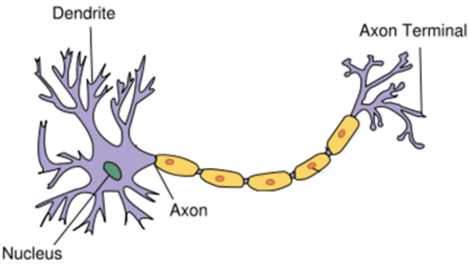
# Sommaire

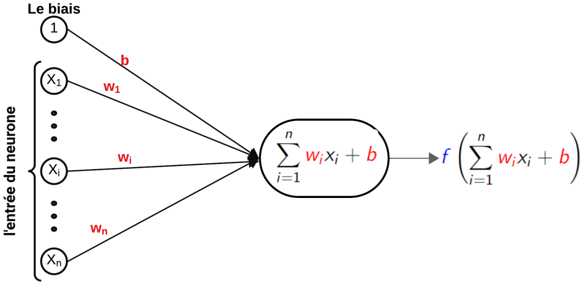


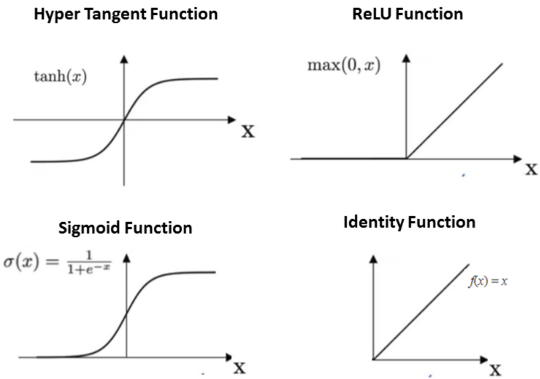
جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

# Perceptron

- Le neurone biologique est composé de dendrites, d'un corps cellulaire et d'axone.
- Le perceptron est une entité de calcul élémentaire inspirée du neurone biologique
- Chaque connexion entre le neurone et ses entrées est dotée d'un poids $w_i$.
- Le neurone fait la somme pondérée de ses entrées, puis passe cette somme à travers une fonction d'activation.






## La fonction Softmax:

$$\begin{array}{rcccl} S & : & \mathbb{R}^K & \longrightarrow & ]0,1[^K \\ & & z & \longmapsto & S(z) = \dfrac{1}{\displaystyle\sum_{k=1}^{K} e^{z_k}} \left(e^{z_1}, \cdots, e^{z_k}, \cdots, e^{z_K}\right)^T \end{array}$$

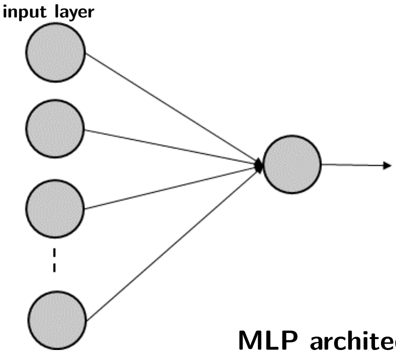
# Sommaire



جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

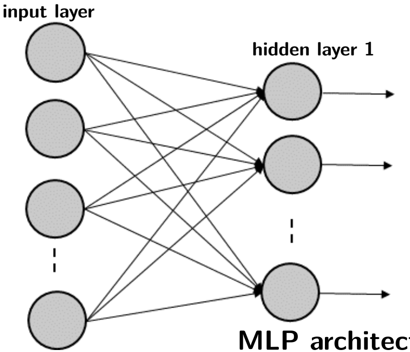
# Multilayer perceptron MLP




**MLP architecture (Feed Forward)**

- Chacun des neurones d'une couche est connecté à tous les neurones de la couche suivante.
- Chaque connexion a un poids $w_{ij}$.
- L'information se propage alors de couche en couche sans retour en arrière possible.
- Pour chaque couche du réseau, il y a un terme de biais.

# Multilayer perceptron MLP

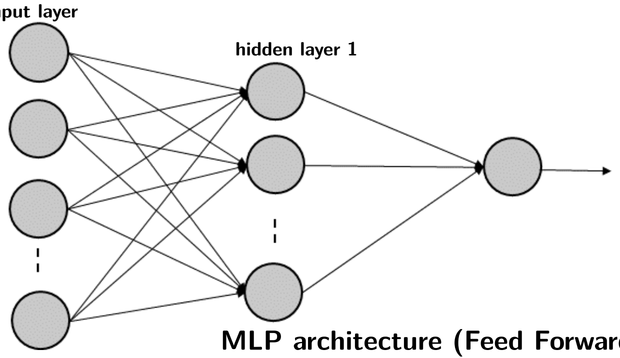



**MLP architecture (Feed Forward)**

- Chacun des neurones d'une couche est connecté à tous les neurones de la couche suivante.
- Chaque connexion a un poids $w_{ij}$.
- L'information se propage alors de couche en couche sans retour en arrière possible.
- Pour chaque couche du réseau, il y a un terme de biais.

جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

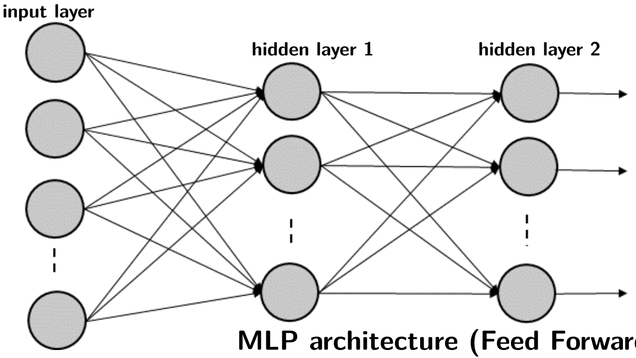
# Multilayer perceptron MLP




**MLP architecture (Feed Forward)**

- Chacun des neurones d'une couche est connecté à tous les neurones de la couche suivante.
- Chaque connexion a un poids $w_{ij}$.
- L'information se propage alors de couche en couche sans retour en arrière possible.
- Pour chaque couche du réseau, il y a un terme de biais.

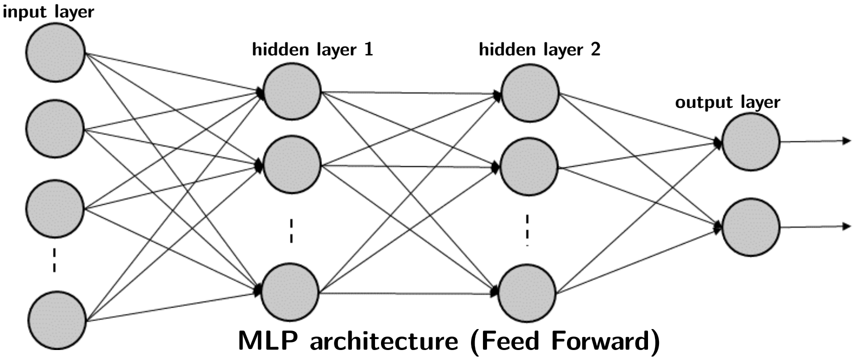
# Multilayer perceptron MLP




**MLP architecture (Feed Forward)**

- Chacun des neurones d'une couche est connecté à tous les neurones de la couche suivante.
- Chaque connexion a un poids $w_{ij}$.
- L'information se propage alors de couche en couche sans retour en arrière possible.
- Pour chaque couche du réseau, il y a un terme de biais.

# Multilayer perceptron MLP

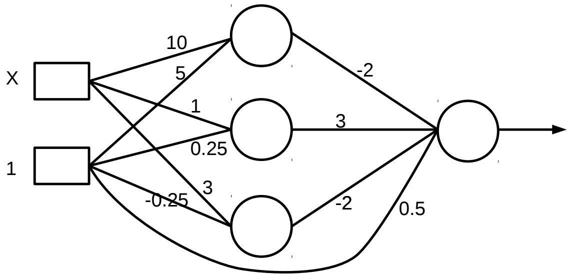



**MLP architecture (Feed Forward)**

- Chacun des neurones d'une couche est connecté à tous les neurones de la couche suivante.
- Chaque connexion a un poids $w_{ij}$.
- L'information se propage alors de couche en couche sans retour en arrière possible.
- Pour chaque couche du réseau, il y a un terme de biais.

# Multilayer perceptron MLP

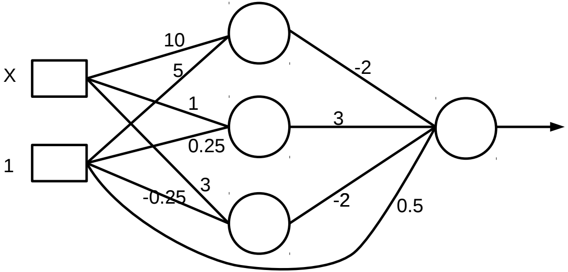
**Exercice 1 :**
Sachant que la fonction d'activation de la couche cachée est le tangent hyperbolique et celle de la couche de sortie est l'identité. Quelle est la sortie de ce réseau de neurones ?



# Multilayer perceptron MLP

**Exercice 1 :**

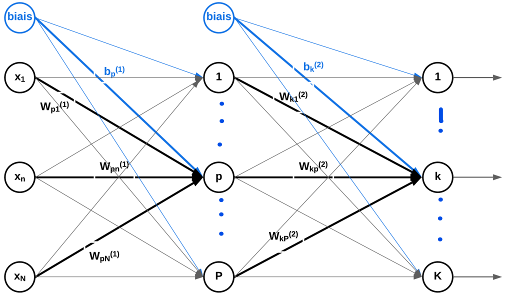
Sachant que la fonction d'activation de la couche cachée est le tangent hyperbolique et celle de la couche de sortie est l'identité. Quelle est la sortie de ce réseau de neurones ?




$$\boxed{\hat{y} = -2th(10x+5) + 3th(x+0.25) - 2th(3x-0.25) + 0.5}$$

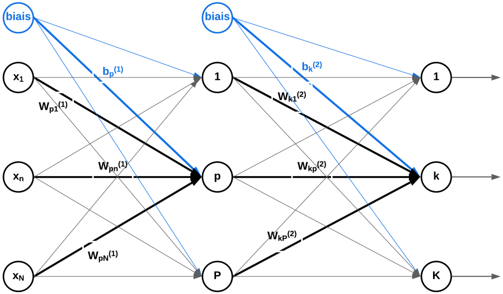
**Exercice 2 :**

Soit $f_1$ la fonction d'activation de la couche cachée et $f_2$ celle de la couche de sortie. Déterminons la fonction décrite par le réseau suivant.



**Exercice 2 :**
Soit $f_1$ la fonction d'activation de la couche cachée et $f_2$ celle de la couche de sortie. Déterminons la fonction décrite par le réseau suivant.

$$\begin{array}{rcl} F_W \; : \; \mathbb{R}^N & \longrightarrow & \mathbb{R}^K \\ x & \longmapsto & F_W(x) = f_2\Big(W^{(2)} f_1\big(W^{(1)} \cdot x + b^{(1)}\big) + b^{(2)}\Big) \end{array}$$



- Soit $D = \{(x^j, y^j)_{j=1}^{J}\} \subset \mathbb{R}^N \times \mathbb{R}^K$ la base d'apprentissage et $\Psi$ une solution inconnue du problème représenté par les données $D$ ($\Psi(x^j) = y^j$).
- L'objectif est de construire un tel réseau MLP $Fw$, qui approxime mieux la solution $\Psi$.

$$\begin{aligned} F_w : \mathbb{R}^N &\longrightarrow \mathbb{R}^K \\ x &\longmapsto F_w(x) = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)^T \end{aligned}$$

- Soit $D = \{(x^j, y^j)_{j=1}^{J}\} \subset \mathbb{R}^N \times \mathbb{R}^K$ la base d'apprentissage et $\Psi$ une solution inconnue du problème représenté par les données $D$ ($\Psi(x^j) = y^j$).
- L'objectif est de construire un tel réseau MLP $Fw$, qui approxime mieux la solution $\Psi$.

$$\begin{aligned} F_w : \mathbb{R}^N &\longrightarrow \mathbb{R}^K \\ x &\longmapsto F_w(x) = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)^T \end{aligned}$$

- La fonction objectif (Loss function)

$$E(W) = \frac{1}{J}\sum_{j=1}^{J} \mathcal{L}(F_W(x^j), y^j)$$

- Soit $D = \{(x^j, y^j)_{j=1}^{J}\} \subset \mathbb{R}^N \times \mathbb{R}^K$ la base d'apprentissage et $\Psi$ une solution inconnue du problème représenté par les données $D$ ($\Psi(x^j) = y^j$).
- L'objectif est de construire un tel réseau MLP $Fw$, qui approxime mieux la solution $\Psi$.

$$\begin{aligned} F_w : \mathbb{R}^N &\longrightarrow \mathbb{R}^K \\ x &\longmapsto F_w(x) = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)^T \end{aligned}$$

- La fonction objectif (Loss function)

$$E(W) = \frac{1}{J} \sum_{j=1}^{J} \mathcal{L}(F_W(x^j), y^j)$$

  - Mean Squared Error (MSE)

$$\mathcal{L}(F_W(x), y) = \frac{1}{2} \|F_W(x) - y\|_2^2$$

  - Cross-Entropy Loss

$$\mathcal{L}(F_W(x), y) = -\sum_{k=1}^{K} y_k \times \log(\hat{y}_k)$$

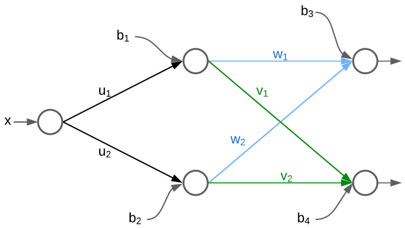
# Sommaire



جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

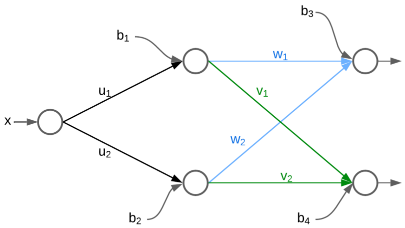
# Algorithme de Backpropagation

## Exemple



# Algorithme de Backpropagation

## Exemple

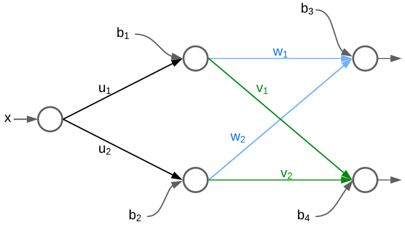



### Forward pass

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

# Algorithme de Backpropagation

## Exemple




**La fonction loss**

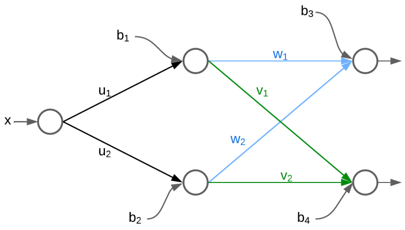
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

**Forward pass**

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

# Algorithme de Backpropagation

## Exemple




### Forward pass

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

### La fonction loss

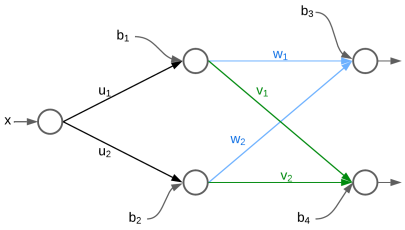
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

### Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_3}$ et $\frac{\partial \mathcal{L}}{\partial z_4}$

# Algorithme de Backpropagation

## Exemple




### La fonction loss

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

### Backward Pass

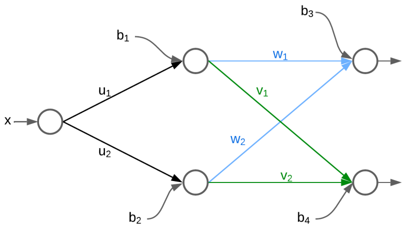
- Calculer $\frac{\partial \mathcal{L}}{\partial z_3}$ et $\frac{\partial \mathcal{L}}{\partial z_4}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, et $\frac{\partial \mathcal{L}}{\partial b_3}$

### Forward pass

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

# Algorithme de Backpropagation

## Exemple




**Forward pass**

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

**La fonction loss**

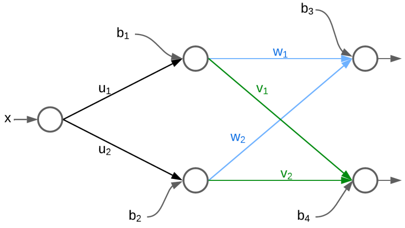
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

**Backward Pass**

- Calculer $\frac{\partial \mathcal{L}}{\partial z_3}$ et $\frac{\partial \mathcal{L}}{\partial z_4}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, et $\frac{\partial \mathcal{L}}{\partial b_3}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial v_1}$, $\frac{\partial \mathcal{L}}{\partial v_2}$, et $\frac{\partial \mathcal{L}}{\partial b_4}$

# Algorithme de Backpropagation

## Exemple




### Forward pass

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

### La fonction loss

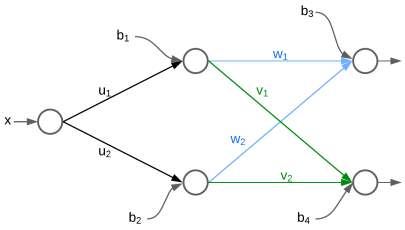
$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

### Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_3}$ et $\frac{\partial \mathcal{L}}{\partial z_4}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, et $\frac{\partial \mathcal{L}}{\partial b_3}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial v_1}$, $\frac{\partial \mathcal{L}}{\partial v_2}$, et $\frac{\partial \mathcal{L}}{\partial b_4}$
- $\frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_1} + \frac{\partial \mathcal{L}}{\partial z_4}\frac{\partial z_4}{\partial z_1}$ et $\frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_2} + \frac{\partial \mathcal{L}}{\partial z_4}\frac{\partial z_4}{\partial z_2}$

# Algorithme de Backpropagation

## Exemple




### La fonction loss

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}_1 - y_1)^2 + \frac{1}{2}(\hat{y}_2 - y_2)^2$$

### Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_3}$ et $\frac{\partial \mathcal{L}}{\partial z_4}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_1}$, $\frac{\partial \mathcal{L}}{\partial w_2}$, et $\frac{\partial \mathcal{L}}{\partial b_3}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial v_1}$, $\frac{\partial \mathcal{L}}{\partial v_2}$, et $\frac{\partial \mathcal{L}}{\partial b_4}$
- $\frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_1} + \frac{\partial \mathcal{L}}{\partial z_4}\frac{\partial z_4}{\partial z_1}$ et $\frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_2} + \frac{\partial \mathcal{L}}{\partial z_4}\frac{\partial z_4}{\partial z_2}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial u_1}$, $\frac{\partial \mathcal{L}}{\partial b_1}$, $\frac{\partial \mathcal{L}}{\partial u_2}$, et $\frac{\partial \mathcal{L}}{\partial b_2}$

### Forward pass

- La sortie de la couche cachée
  - $z_1 = u_1 x + b_1$ et $a_1 = f_1(z_1)$
  - $z_2 = u_2 x + b_2$ et $a_2 = f_1(z_2)$
- La sortie de la couche de sortie
  - $z_3 = w_1 a_1 + w_2 a_2 + b_3$ et $a_3 = f_2(z_3) = \hat{y}_1$
  - $z_4 = v_1 a_1 + v_2 a_2 + b_4$ et $a_4 = f_2(z_4) = \hat{y}_2$

# Algorithme de Backpropagation

## Exemple

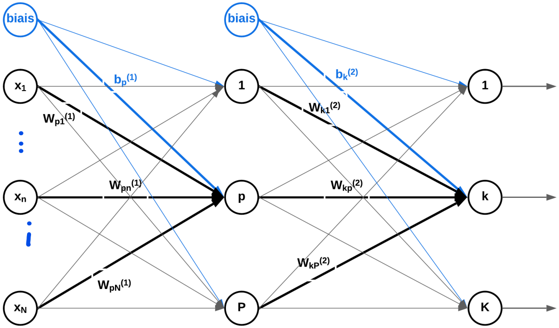
- $\delta_3 := \frac{\partial \mathcal{L}}{\partial z_3} = (a_3 - y_1) f_2'(z_3)$
- $\delta_4 := \frac{\partial \mathcal{L}}{\partial z_4} = (a_4 - y_2) f_2'(z_4)$
  - $\frac{\partial \mathcal{L}}{\partial w_1} = \delta_3 a_1$
  - $\frac{\partial \mathcal{L}}{\partial w_2} = \delta_3 a_2$
  - $\frac{\partial \mathcal{L}}{\partial b_3} = \delta_3$
  - $\frac{\partial \mathcal{L}}{\partial v_1} = \delta_4 a_1$
  - $\frac{\partial \mathcal{L}}{\partial v_2} = \delta_4 a_2$
  - $\frac{\partial \mathcal{L}}{\partial b_4} = \delta_4$

- $\delta_3 := \frac{\partial \mathcal{L}}{\partial z_3} = (a_3 - y_1) f_2'(z_3)$
- $\delta_4 := \frac{\partial \mathcal{L}}{\partial z_4} = (a_4 - y_2) f_2'(z_4)$
  - $\frac{\partial \mathcal{L}}{\partial w_1} = \delta_3 a_1$
  - $\frac{\partial \mathcal{L}}{\partial w_2} = \delta_3 a_2$
  - $\frac{\partial \mathcal{L}}{\partial b_3} = \delta_3$
  - $\frac{\partial \mathcal{L}}{\partial v_1} = \delta_4 a_1$
  - $\frac{\partial \mathcal{L}}{\partial v_2} = \delta_4 a_2$
  - $\frac{\partial \mathcal{L}}{\partial b_4} = \delta_4$

- $\delta_1 := \frac{\partial \mathcal{L}}{\partial z_1} = \delta_3 w_1 f_1'(z_1) + \delta_4 v_1 f_1'(z_1)$
- $\delta_2 := \frac{\partial \mathcal{L}}{\partial z_2} = \delta_3 w_2 f_1'(z_2) + \delta_4 v_2 f_1'(z_2)$
  - $\frac{\partial \mathcal{L}}{\partial u_1} = \delta_1 x$
  - $\frac{\partial \mathcal{L}}{\partial b_1} = \delta_1$
  - $\frac{\partial \mathcal{L}}{\partial u_2} = \delta_2 x$
  - $\frac{\partial \mathcal{L}}{\partial b_2} = \delta_2$

جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

# Algorithme de Backpropagation




- $f_1$: la fonction d'activation de la couche cachée.
- $f_2$: la fonction d'activation de la couche de sortie.

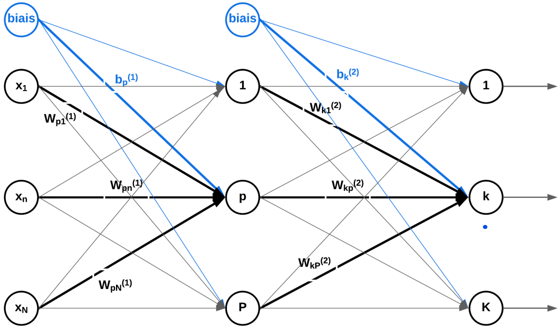
La fonction objectif (Loss function)

$$E(W) = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}(F_W(x^j), y^j)$$

## Étape 1 : Forward pass

- $z_p^{(1)} = \sum_{n=1}^{N} w_{pn}^{(1)} x_n + b_p^{(1)}$ et $a_p^{(1)} = f_1\left(z_p^{(1)}\right)$ $\forall p \in \{1, \cdots, p\}$

# Algorithme de Backpropagation




- $f_1$: la fonction d'activation de la couche cachée.
- $f_2$: la fonction d'activation de la couche de sortie.

La fonction objectif (Loss function)

$$E(W) = \frac{1}{J}\sum_{j=1}^{J} \mathcal{L}(F_W(x^j), y^j)$$

## Étape 1 : Forward pass

- $z_p^{(1)} = \sum_{n=1}^{N} w_{pn}^{(1)} x_n + b_p^{(1)}$ et $a_p^{(1)} = f_1\left(z_p^{(1)}\right)$
  $\forall p \in \{1, \cdots, p\}$
- $z_k^{(2)} = \sum_{p=1}^{P} w_{kp}^{(2)} a_p^{(1)} + b_k^{(2)}$ et $a_k^{(2)} = f_2\left(z_k^{(2)}\right)$
  $\forall k \in \{1, \cdots, K\}$

# Algorithme de Backpropagation

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$

# Algorithme de Backpropagation

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{kp}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{kp}^{(2)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial b_k^{(2)}}$

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{kp}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{kp}^{(2)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial b_k^{(2)}}$
- Calculer $\frac{\partial \mathcal{L}}{\partial z_p^{(1)}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial z_p^{(1)}}$

# Algorithme de Backpropagation

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{kp}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{kp}^{(2)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial b_k^{(2)}}$
- Calculer $\frac{\partial \mathcal{L}}{\partial z_p^{(1)}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial z_p^{(1)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{pn}^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial w_{pn}^{(1)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_p^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial b_p^{(1)}}$

# Algorithme de Backpropagation

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{kp}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{kp}^{(2)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial b_k^{(2)}}$
- Calculer $\frac{\partial \mathcal{L}}{\partial z_p^{(1)}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial z_p^{(1)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{pn}^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial w_{pn}^{(1)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_p^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial b_p^{(1)}}$

$$E(W) = \frac{1}{J} \sum_{j=1}^{J} \mathcal{L}(\hat{y}^j, y^j)$$

# Algorithme de Backpropagation

## Étape 2 : Backward Pass

- Calculer $\frac{\partial \mathcal{L}}{\partial z_k^{(2)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{kp}^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{kp}^{(2)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_k^{(2)}} = \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial b_k^{(2)}}$
- Calculer $\frac{\partial \mathcal{L}}{\partial z_p^{(1)}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial z_p^{(1)}}$
  - Calculer $\frac{\partial \mathcal{L}}{\partial w_{pn}^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial w_{pn}^{(1)}}$
  - $\frac{\partial \mathcal{L}}{\partial b_p^{(1)}} = \frac{\partial \mathcal{L}}{\partial z_p^{(1)}} \frac{\partial z_p^{(1)}}{\partial b_p^{(1)}}$

$$E(W) = \frac{1}{J} \sum_{j=1}^{J} \mathcal{L}(\hat{y}^j, y^j)$$

- $\frac{\partial E}{\partial w_{pn}^{(1)}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\partial \mathcal{L}(\hat{y}^j, y^j)}{\partial w_{pn}^{(1)}}$ et $\frac{\partial E}{\partial b_p^{(1)}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\partial \mathcal{L}(\hat{y}^j, y^j)}{\partial b_p^{(1)}}$
- $\frac{\partial E}{\partial w_{kp}^{(2)}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\partial \mathcal{L}(\hat{y}^j, y^j)}{w_{kp}^{(2)}}$ et $\frac{\partial E}{\partial b_k^{(2)}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\partial \mathcal{L}(\hat{y}^j, y^j)}{b_k^{(2)}}$

# Algorithme de Backpropagation

## Étape 3 : Mise à jour des paramètres

- $w_{pn}^{(1)} = w_{pn}^{(1)} - \eta \dfrac{\partial E}{\partial w_{pn}^{(1)}}$ et $b_p^{(1)} = b_p^{(1)} - \eta \dfrac{\partial E}{\partial b_p^{(1)}}$
- $w_{kp}^{(2)} = w_{kp}^{(2)} - \eta \dfrac{\partial E}{\partial w_{kp}^{(2)}}$ et $b_k^{(2)} = b_k^{(2)} - \eta \dfrac{\partial E}{\partial b_k^{(2)}}$

Où $\eta$ est le pas d'apprentissage prédéterminé (Learning rate).

جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

# Algorithme de Backpropagation

## Exercice 3 :

On considère un réseau de neurones MLP similaire à celui de **l'exercice 2**, où la fonction d'activation de la couche cachée est **sigmoïde** $\sigma$ et celle de la couche de sortie est **softmax** $S$, avec l'utilisation de cross entropy loss comme fonction objectif.

- Soit $a^{(2)} = S(z^{(2)})$, vérifier que:

$$\boxed{\frac{\partial a_k^{(2)}}{\partial z_{k'}^{(2)}} = \begin{cases} a_k^{(2)}(1 - a_k^{(2)}) & \text{if } k = k' \\ -a_k^{(2)} a_k^{(2)\prime} & \text{if } k \neq k' \end{cases}}$$

- Vérifier que:

$$\boxed{\frac{\partial \mathcal{L}}{\partial z_k^{(2)}} = a_k^{(2)} - y_k}$$

- Déterminer $\frac{\partial E}{\partial w_{kp}^{(2)}}$ et $\frac{\partial E}{\partial b_k^{(2)}}$

- Vérifier que:

$$\boxed{\frac{\partial \mathcal{L}}{\partial z_p^{(1)}} = \sum_{k=1}^{K} (a_k^{(2)} - y_k) w_{kp}^{(2)} \sigma'(z_p^{(1)})}$$

- Déterminer $\frac{\partial E}{\partial w_{pn}^{(1)}}$, et $\frac{\partial E}{\partial b_p^{(1)}}$

# Multilayer perceptron en sklearn (Classification)

Pour la classification on utilise la classe MLPClassifier du module sklearn.neural_network

- Output activation : softmax (multiclass) or logistic (binary)
- Fonction coût : the Cross-Entropy loss function

```
from sklearn.neural_network import MLPClassifier
network =  MLPClassifier(hidden_layer_sizes=(100,),
    activation='relu', learning_rate_init=0.001, max_iter=200,
    random_state = None)
network.fit(x_train, y_train)
```

# Multilayer perceptron en sklearn (Regression)

Pour la regression on utilise la classe MLPRegressor du module sklearn.neural_network

- Output activation : identity
- Fonction coût : the Squared-error loss function

```
from sklearn.neural_network import MLPRegressor
network =  MLPRegressor(hidden_layer_sizes=(100,),
    activation='relu', learning_rate_init=0.001, max_iter=200,
    random_state = None)
network.fit(x_train, y_train)
```

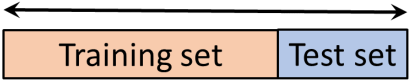
# Sommaire

# Validation croisée

## Validation croisée

La validation croisée est une procédure de rééchantillonnage permettant d'évaluer un modèle d'apprentissage automatique et de tester sa capacité de faire des prédictions sur des données de test indépendant. Elle permet ainsi de choisir lequel entre deux modèles représente mieux le vrai modèle de nos données.

La performance dépend du "data splitting"

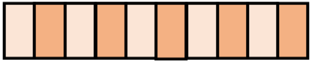



L'évaluation doit être faite au cours du développement de modèle indépendamment du test.

**Leave-P-Out Cross Validation**

On utilise des échantillons de taille p pour la validation et les n-p échantillons qui restent comme ensemble d'apprentissage. On considère toutes les combinaisons possibles. Le nombre d'itérations est alors $C_n^p$.

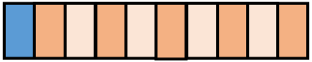
**Leave-One-Out**



# Validation croisée

**Leave-P-Out Cross Validation**
On utilise des échantillons de taille p pour la validation et les n-p échantillons qui restent comme ensemble d'apprentissage. On considère toutes les combinaisons possibles. Le nombre d'itérations est alors $C_n^p$.

**Leave-One-Out**



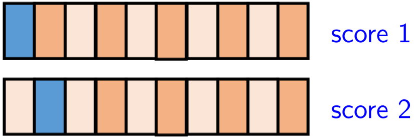
score 1

# Validation croisée

**Leave-P-Out Cross Validation**

On utilise des échantillons de taille p pour la validation et les n-p échantillons qui restent comme ensemble d'apprentissage. On considère toutes les combinaisons possibles. Le nombre d'itérations est alors $C_n^p$.

**Leave-One-Out**

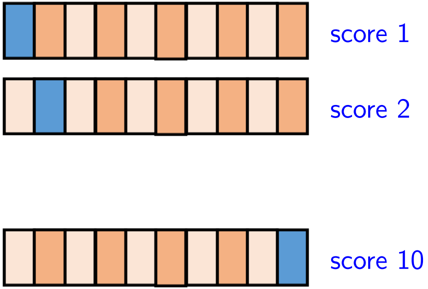


# Validation croisée

**Leave-P-Out Cross Validation**

On utilise des échantillons de taille p pour la validation et les n-p échantillons qui restent comme ensemble d'apprentissage. On considère toutes les combinaisons possibles. Le nombre d'itérations est alors $C_n^p$.

**Leave-One-Out**



# Validation croisée

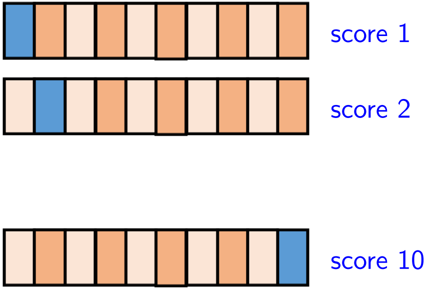
**Leave-P-Out Cross Validation**

Create nCp combainaison from n example, Train with n-p and test with the p selected.

On utilise des échantillons de taille p pour la validation et les n-p échantillons qui restent comme ensemble d'apprentissage. On considère toutes les combinaisons possibles. Le nombre d'itérations est alors $C_n^p$.

**Leave-One-Out**

One for test ,and others examples
for training
- if data countain 1 milion line, so
we have 1 milion itteration!!

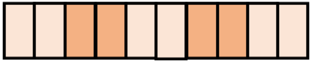



Cette méthode est couteuse lorsqu'il s'agit d'un nombre très grand d'échatillons.

# Validation croisée

**K-Fold Cross Validation**

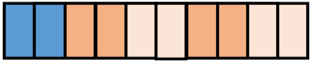
Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).



# Validation croisée

**K-Fold Cross Validation**

Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).

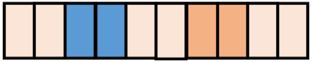


{score1, }

# Validation croisée

**K-Fold Cross Validation**

Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).



$\{$score1, score2, $\}$

# Validation croisée

**K-Fold Cross Validation**

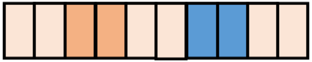
Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).

$\{\text{score1}, \text{score2}, \text{score3}, \qquad\qquad \}$

# Validation croisée

**K-Fold Cross Validation**

Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).

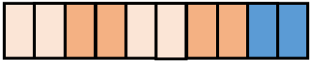


{score1, score2, score3, score4, }

# Validation croisée

**K-Fold Cross Validation**

Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).

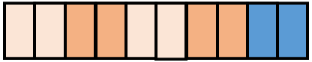


$\{$score1, score2, score3, score4, score5$\}$

# Validation croisée

**K-Fold Cross Validation**

Les données sont divisées en K sous-ensembles. Pour chaque itération, un sous-ensemble est utilisé pour la validation et les autres pour l'entrainement. Le nombre d'itérations est alors égale à K (souvent on choisit K=5 à 10).



{score1, score2, score3, score4, score5}

**Stratified K-Fold Cross Validation**

Cette méthode de validation croisée est une légère variation de K-Fold, de sorte que chaque 'fold' contient approximativement le même pourcentage d'échantillons de chaque classe cible, ou en cas de problèmes de prédiction, la valeur de réponse moyenne est approximativement égale dans tous les 'folds'.

# Sommaire

# Tuning des hyperparamètres

Les hyper-paramètres sont des paramètres qui ne sont pas directement appris dans le modèle. Les exemples typiques incluent le paramètre de régularisation, l'architecture du réseau, ...

De nombreuses hyperparamètres doivent être prises lors de la construction du réseau MLP, notamment :

- Le nombre de couches cachées (la profondeur du réseau).
- Le nombre de neurones par couche cachée (la largeur du réseau).
- Le choix de la fonction d'activation pour chaque couche.
- Le pas d'apprentissage $\eta$

La commande suivante permet de récupérer les paramètres d'un modèle (de la librairie sklearn):

```
model.get_params()
```

| Hyperparamètre 1 \ Hyperparamètre 2 | val 1 | val 2 | val 3 | val 4 | val 5 | val 6 |
|---|---|---|---|---|---|---|
| val 1 | Score(1,1) | Score(1,2) | Score(1,3) | Score(1,4) | Score(1,5) | Score(1,6) |
| val 2 | Score(2,1) | Score(2,2) | Score(2,3) | Score(2,4) | Score(2,5) | Score(2,6) |
| val 3 | Score(3,1) | Score(3,2) | Score(3,3) | Score(3,4) | Score(3,5) | Score(3,6) |
| val 4 | Score(4,1) | Score(4,2) | Score(4,3) | Score(4,4) | Score(4,5) | Score(4,6) |

La fonction `GridSearchCV` de `sklearn.model_selection` considère de manière exhaustive toutes les combinaisons de paramètres de la grille.

# Tuning des hyperparamètres

## Voici des exemples de grilles :

```
grid={"C":[0.0001, 0.001, 0.01, 0.1, 1.0], "penalty":["l2","l1"]}

grid={
'learning_rate': ["constant", "invscaling", "adaptive"],
'hidden_layer_sizes': [(100,1), (100,2), (100,3)],
'alpha': [10.0 ** -np.arange(1, 7)],
'activation': ["logistic", "relu", "Tanh"] }

grid={'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}
```

## Utilisation :

```
model_cv=GridSearchCV(model,grid,cv=10)
model_cv.fit(X,y)

print("tuned hpyerparameters :(best parameters) ",model_cv.best_params_)
print("accuracy :",model_cv.best_score_)
```

# End Lecture N° 3