

# **Deep Learning par la Pratique**

Apprentissage d'un Réseau de Neurones

---

Les données en entrée doivent être normalisées :

$$X_{norm} = \frac{X - \text{mean}(X_{train})}{\text{std}(X_{train})}$$

- 1 - initialisation **aléatoire** du modèle
- 2 - Tant que(critère arrêt == 0)
  - Selection aléatoire d'un **batch** de données
  - **Forward** : Passe avant du **batch** dans le modèle
  - Calcul de l'erreur par rapport aux sorties attendues
  - **Backward** : Rétropropagation du gradient de l'erreur en fonction des paramètres dans le modèle (mise à jour du modèle)
  - Calcul critère arrêt
- 3 - Calcul de l'erreur sur un échantillon de données **qui n'ont JAMAIS été vues par le modèle pendant l'apprentissage !**

Quelle fonction de perte (Loss)  
choisir ?

# Fonction de perte (Loss function)

Exemple facile : “Prédire la température demain”

$$\text{Erreur} = T_{pred} - T_{real}$$

# Fonction de perte (Loss function)

Exemple difficile : “Prédire la catégorie d’une image”

≈ Distance entre la sortie et la cible

**Sortie :**

0.0	0.1	0.4	0.0	0.0	0.2	0.1	0.0	0.2	0.0
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

**Cible :**

0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

# Fonction de perte (Loss function)

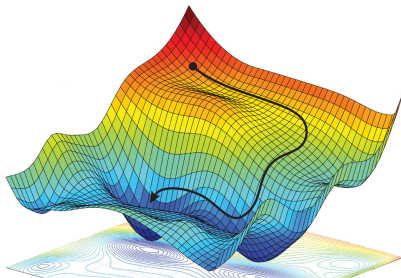
Calcul du gradient de l'erreur par rapport aux paramètres :

$$grad_i = \frac{\partial Err}{\partial w_i}$$

Mise à jour :

$$w'_i = w_i - \gamma * grad_i$$

où :  $0 < \gamma < 1$  (learning rate)



# Fonction de perte (Loss function)

## Regression Loss Functions :

- Erreur **absolue**
- Erreur **lissée**
- ...

## Classification Loss Functions :

- Entropies-croisées
- Log vraisemblance
- Loss à marge
- ...

## Embedding Loss Functions :

- $L1, L2, \dots$
- Distance cosinus
- ...



Exemple de descente de gradient sur un cas simple : la regression

- $y = a.X + b$
- $L = \frac{1}{2n} \sum_{i=[1..n]} (y_i^* - y_i)^2$
- $L = \frac{1}{2n} \sum_{i=[1..n]} (y_i^* - (a.x_i + b))^2$
- ...
- $\frac{\partial L}{\partial a} = \frac{1}{n} \sum_{i=[1..n]} (a.X + b - y^*).X$
- $\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=[1..n]} (a.X + b - y^*)$

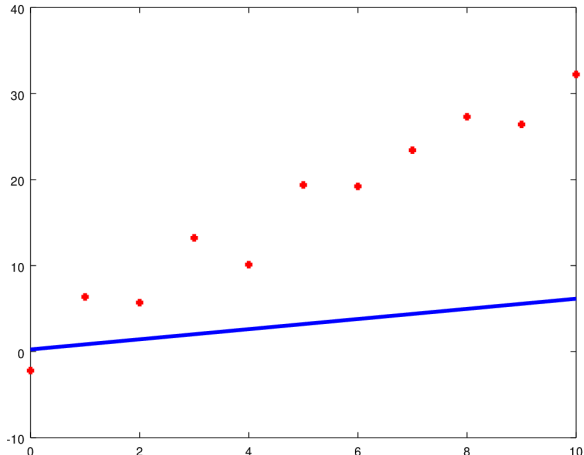
**M.A.J :**

- $a = a - \gamma \frac{\partial L}{\partial a}$
- $b = b - \gamma \frac{\partial L}{\partial b}$
- où  $1 > \gamma > 0$  (learning rate)

# Gradient et mise à jour

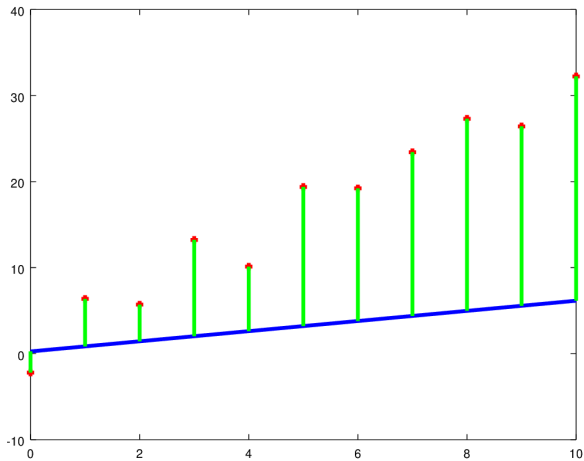
initialisation au hasard (  $\gamma = 0.01$  )

- $a = 0.58$  ( $\hat{a} = 3.0$ )
- $b = 0.25$  ( $\hat{b} = 0.5$ )



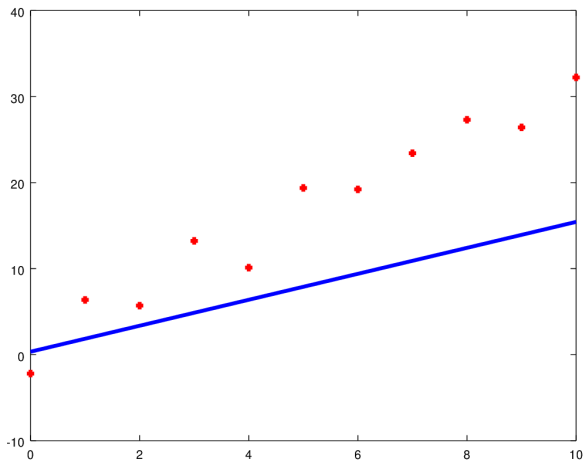
# Gradient et mise à jour

- $a = 0.58$  ( $\hat{a} = 3.0$ )
- $b = 0.25$  ( $\hat{b} = 0.5$ )



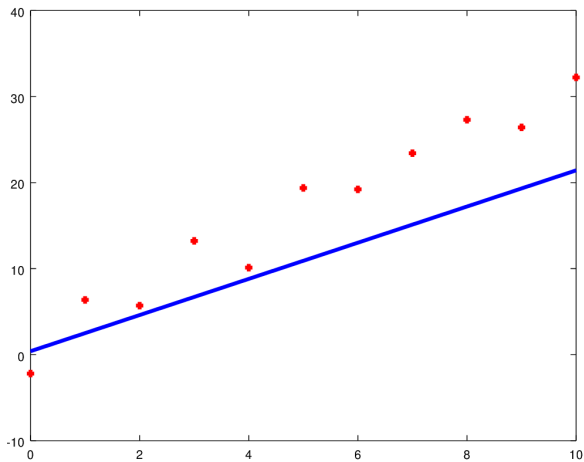
# Gradient et mise à jour

- $a = 1.50$  ( $\hat{a} = 3.0$ )
- $b = 0.35$  ( $\hat{b} = 0.5$ )



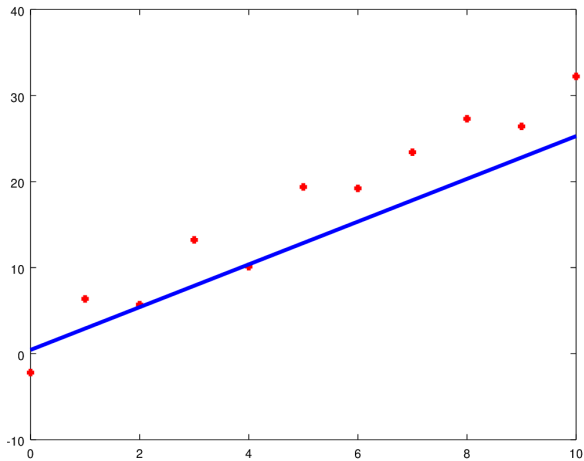
# Gradient et mise à jour

- $a = 2.10$  ( $\hat{a} = 3.0$ )
- $b = 0.40$  ( $\hat{b} = 0.5$ )



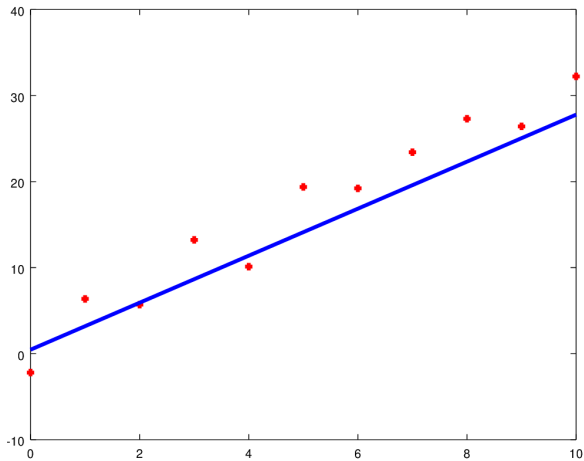
# Gradient et mise à jour

- $a = 2.48$  ( $\hat{a} = 3.0$ )
- $b = 0.43$  ( $\hat{b} = 0.5$ )



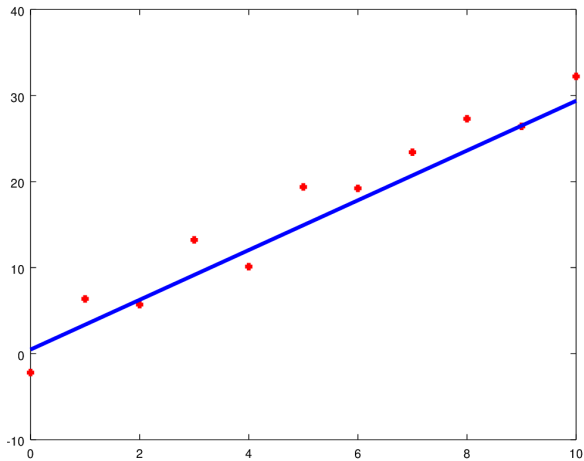
# Gradient et mise à jour

- $a = 2.73$  ( $\hat{a} = 3.0$ )
- $b = 0.46$  ( $\hat{b} = 0.5$ )



# Gradient et mise à jour

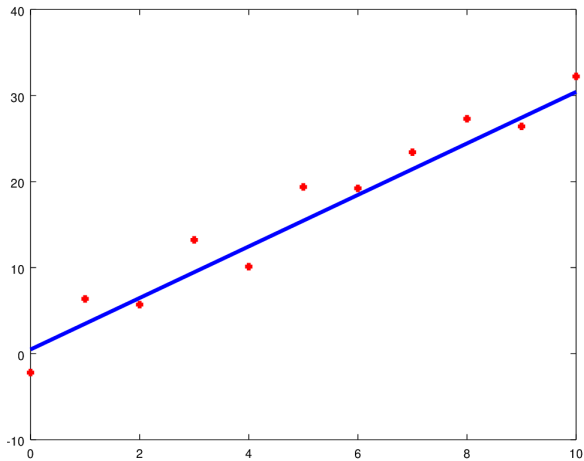
- $a = 2.89$  ( $\hat{a} = 3.0$ )
- $b = 0.47$  ( $\hat{b} = 0.5$ )





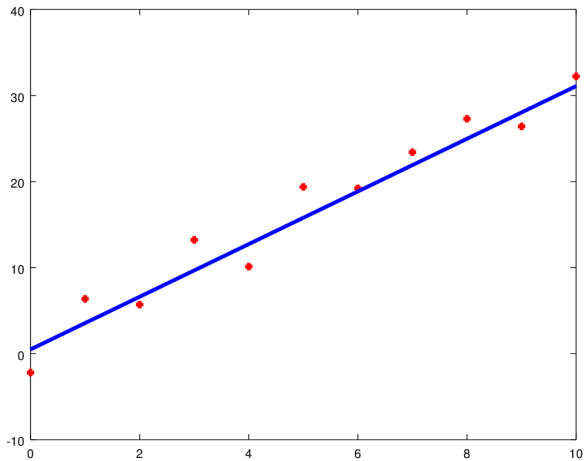
# Gradient et mise à jour

- $a = 2.99$  ( $\hat{a} = 3.0$ )
- $b = 0.48$  ( $\hat{b} = 0.5$ )



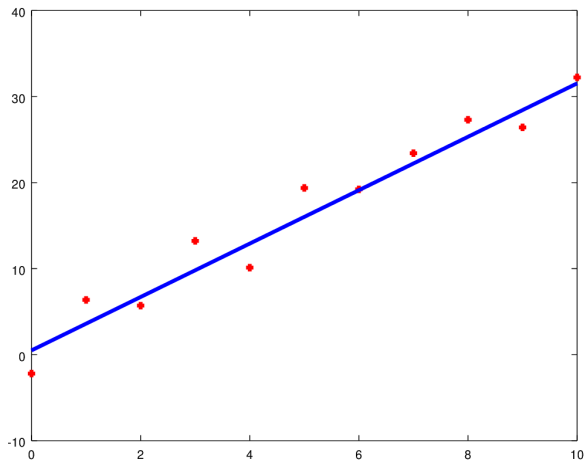
# Gradient et mise à jour

- $a = 3.06$  ( $\hat{a} = 3.0$ )
- $b = 0.49$  ( $\hat{b} = 0.5$ )



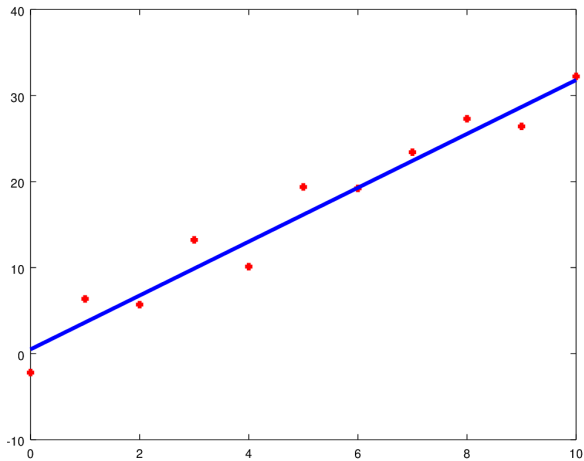
# Gradient et mise à jour

- $a = 3.10$  ( $\hat{a} = 3.0$ )
- $b = 0.49$  ( $\hat{b} = 0.5$ )



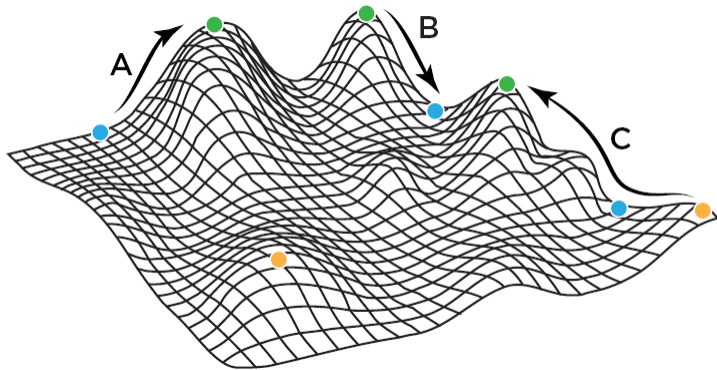
# Gradient et mise à jour

- $a = 3.13$  ( $\hat{a} = 3.0$ )
- $b = 0.50$  ( $\hat{b} = 0.5$ )



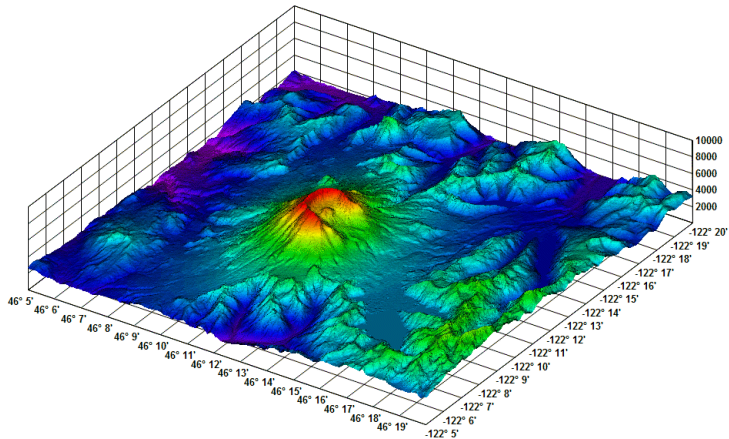
Convergence du modèle vers l'optimal ?

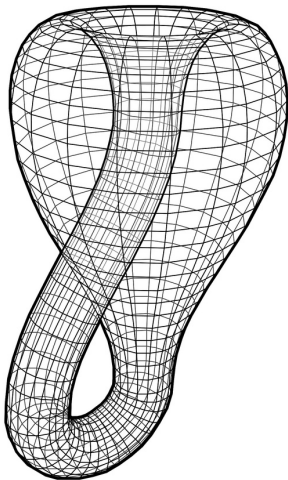
# Convergence



By Max Olson for FutureBlind

# Convergence







# Evaluation des performances

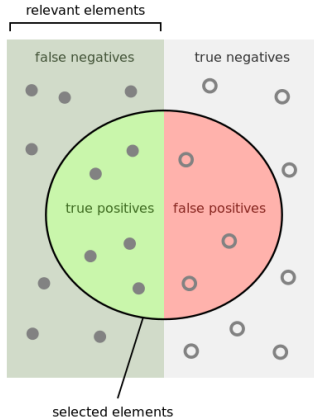
$$Précision_i = \frac{\text{vrai positifs}_i}{\text{vrai positifs}_i + \text{faux positif}_i}$$

$$Précision = \frac{\sum_{i=1}^n précision_i}{n}$$

$$Rappel_i = \frac{\text{vrai positifs}_i}{\text{vrai positifs}_i + \text{faux négatifs}_i}$$

$$Rappel = \frac{\sum_{i=1}^n rappel_i}{n}$$

# Evaluation des performances



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# Évaluation — outils — matrice de confusion

