

# Descente de gradient

# Optimisation

- En machine learning il y a 2 critères différents que l'on cherche à optimiser :

# Optimisation

- En machine learning il y a 2 critères différents que l'on cherche à optimiser :

## **La MSE sur les données d'entraînement**

- D'autres fonctions que la MSE sont possibles
- En fait c'est un détail d'implémentation : Si on pouvait trouver des bons paramètres sans avoir à regarder la MSE sur un jeu d'entraînement, on le ferait.

# Optimisation

- En machine learning il y a 2 critères différents que l'on cherche à optimiser :

## La MSE sur les données d'entraînement

- D'autres fonctions que la MSE sont possibles
- En fait c'est un détail d'implémentation : Si on pouvait trouver des bons paramètres sans avoir à regarder la MSE sur un jeu d'entraînement, on le ferait.

## L'erreur de prédiction

- Mesurée par la MSE sur les données de test (pour les régressions) ou par la précision, l'accuracy... pour les classification
- Il s'agit de notre vrai objectif : minimiser cette erreur est exactement équivalent à résoudre le problème posé.

# Optimisation

- En machine learning il y a 2 critères différents que l'on cherche à optimiser :

## La MSE sur les données d'entraînement

- D'autres fonctions que la MSE sont possibles
- En fait c'est un détail d'implémentation : Si on pouvait trouver des bons paramètres sans avoir à regarder la MSE sur un jeu d'entraînement, on le ferait.

## L'erreur de prédiction

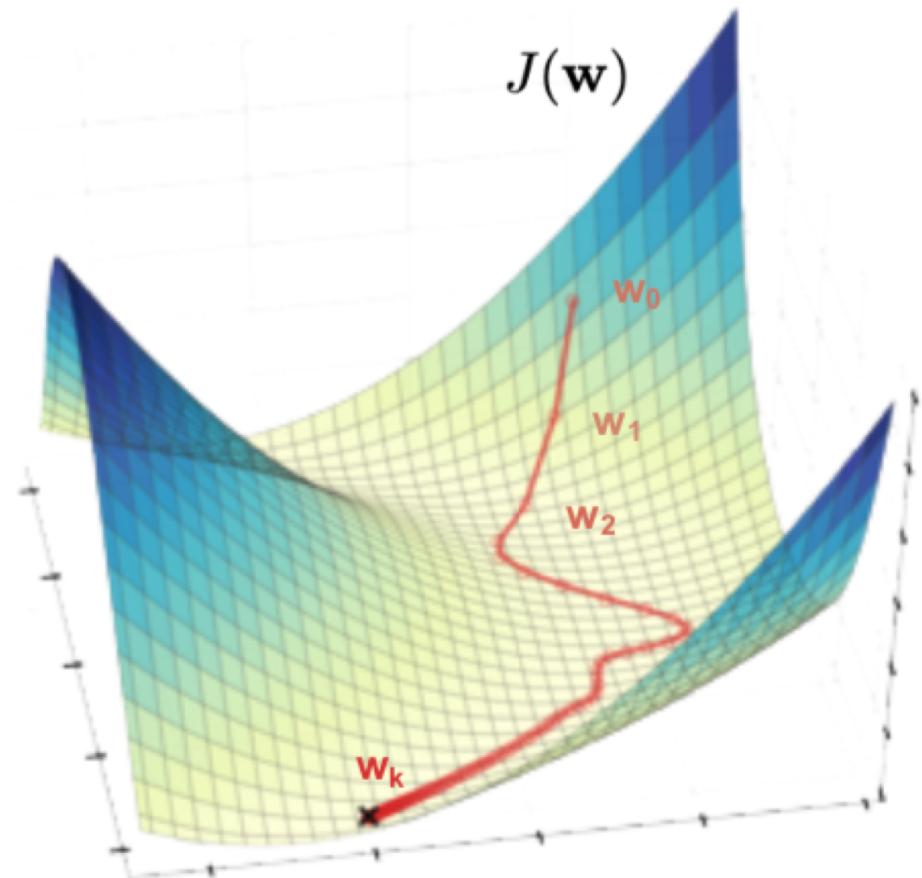
- Mesurée par la MSE sur les données de test (pour les régressions) ou par la précision, l'accuracy... pour les classification
- Il s'agit de notre vrai objectif : minimiser cette erreur est exactement équivalent à résoudre le problème posé.

- Toute la théorie de machine learning repose sur notre confiance qu'un modèle qui minimise la première va aussi minimiser la deuxième.

# Principe

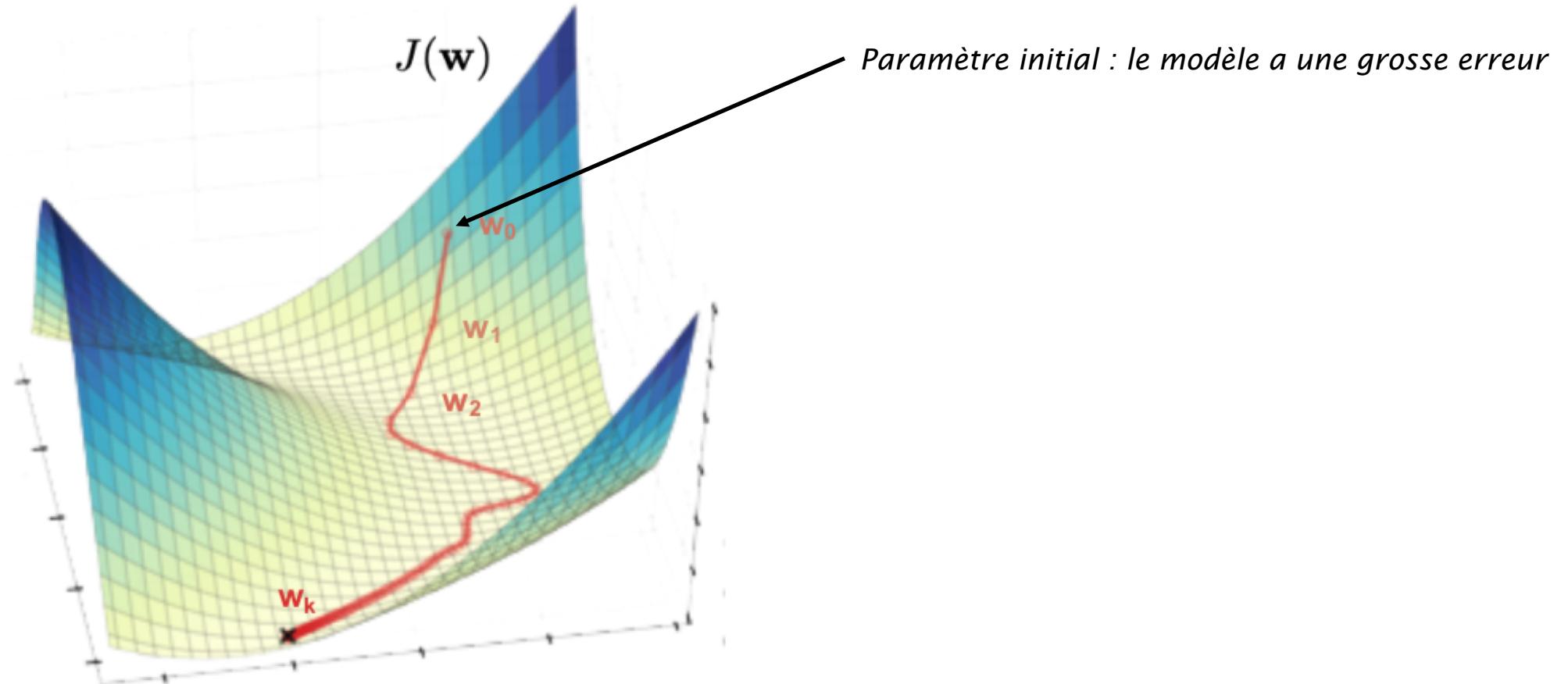
# Descente de gradient : le principe

- On voit bien que pour trouver le point le plus bas d'une fonction, il suffit de partir de n'importe où et de descendre.



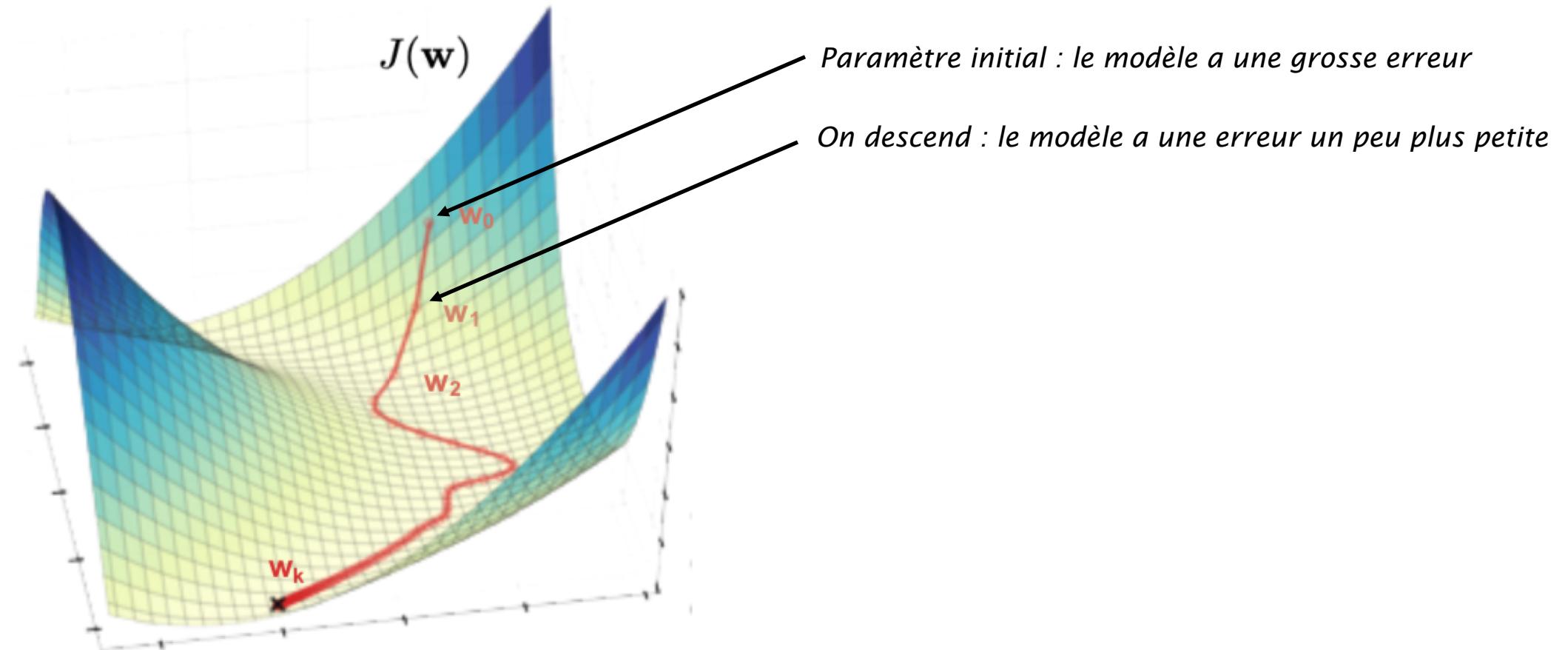
# Descente de gradient : le principe

- On voit bien que pour trouver le point le plus bas d'une fonction, il suffit de partir de n'importe où et de descendre.



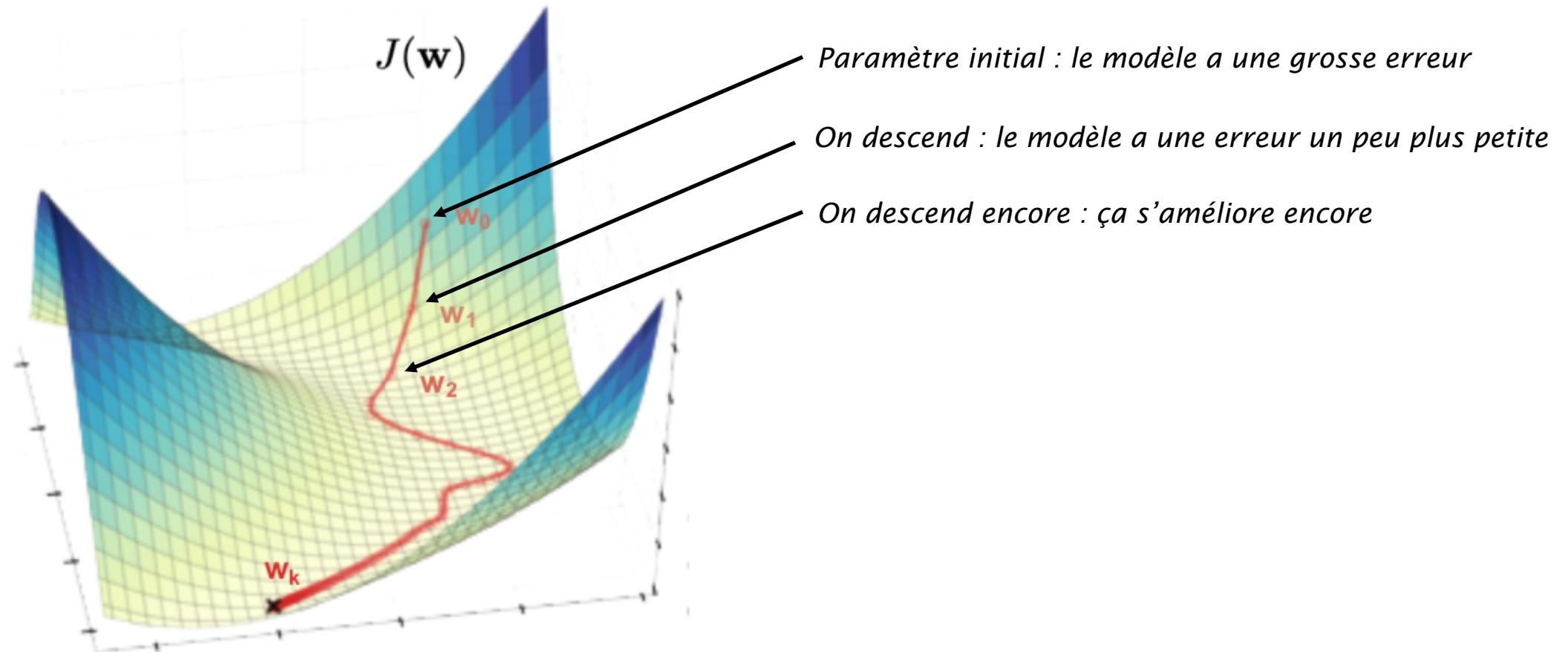
# Descente de gradient : le principe

- On voit bien que pour trouver le point le plus bas d'une fonction, il suffit de partir de n'importe où et de descendre.



# Descente de gradient : le principe

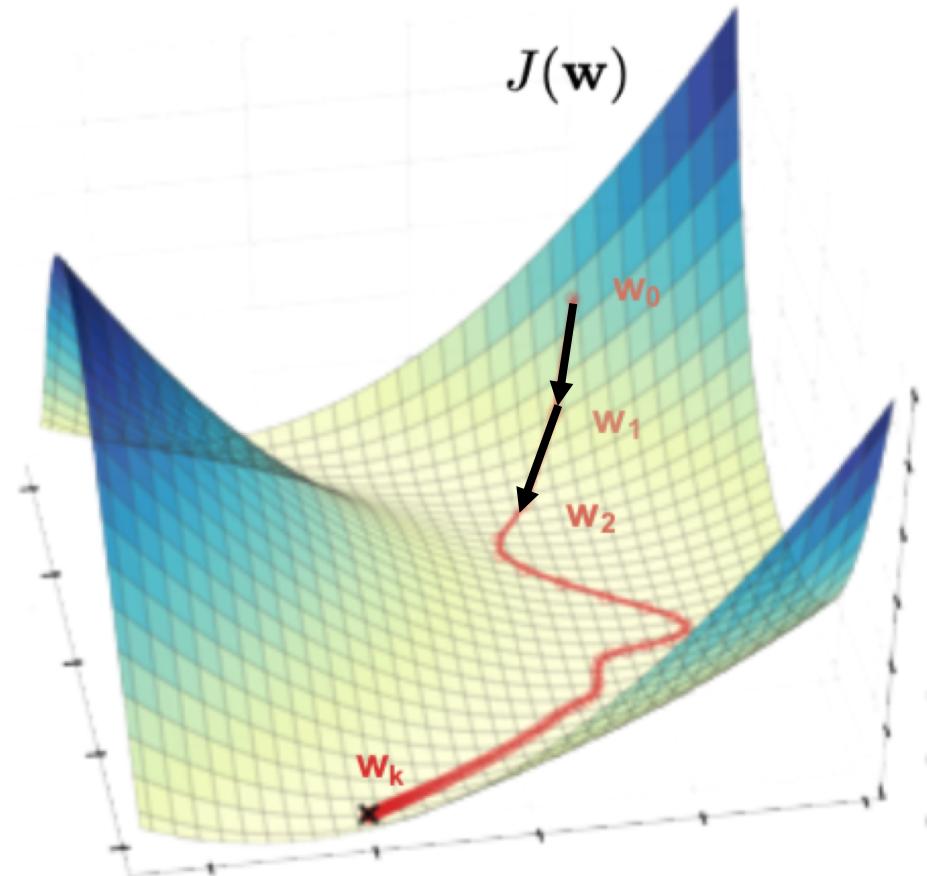
- On voit bien que pour trouver le point le plus bas d'une fonction, il suffit de partir de n'importe où et de descendre.



# Le gradient

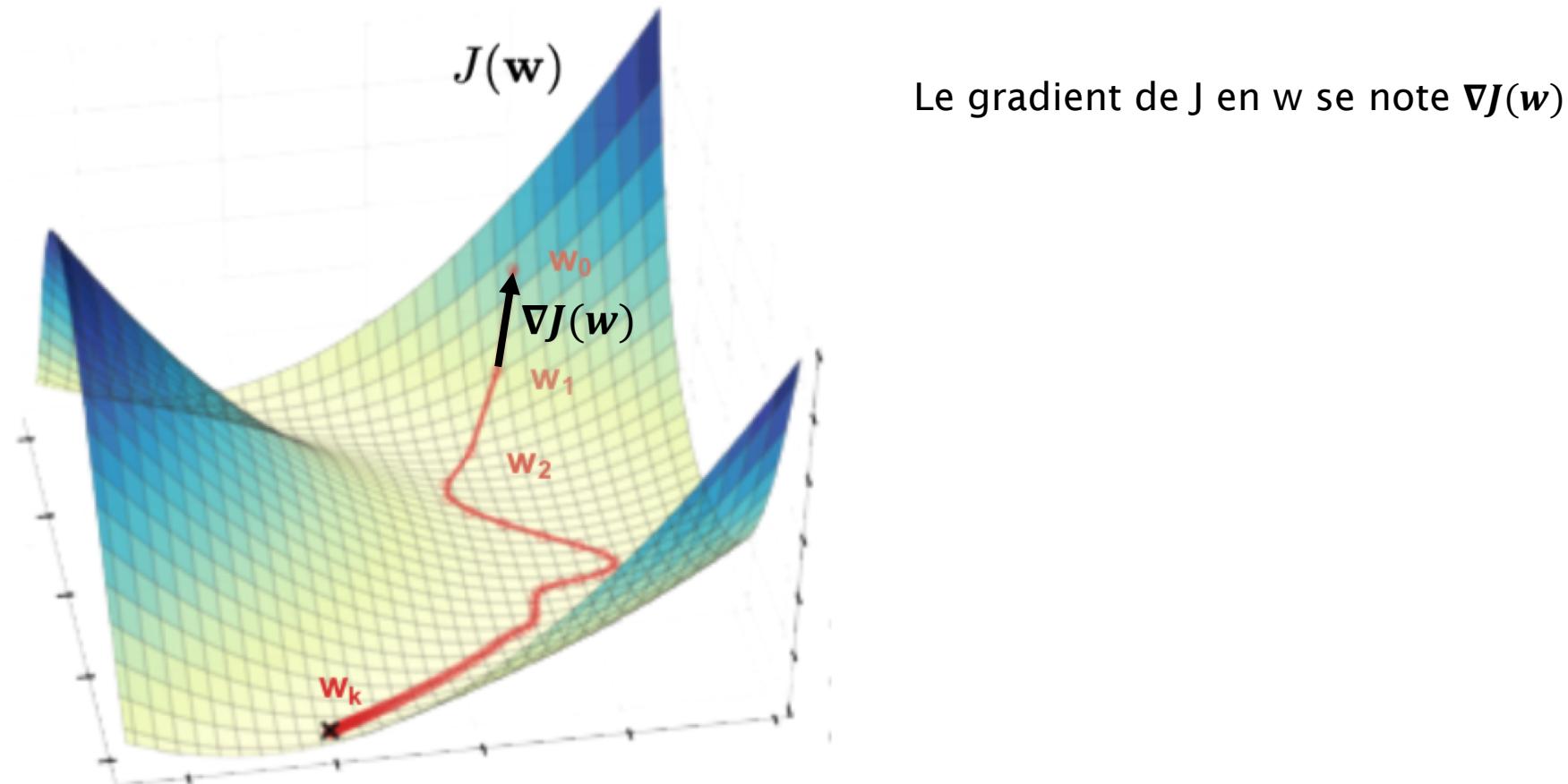
# Le gradient

- Comment savoir quelle direction prendre à chaque étape ?



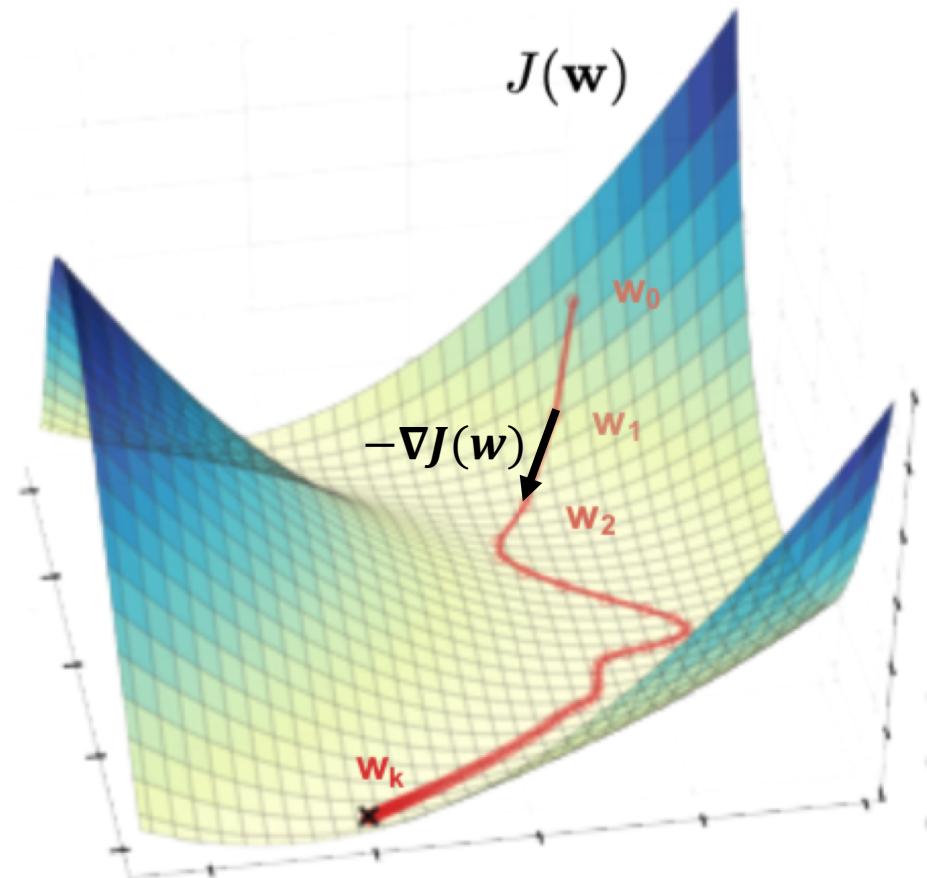
# Le gradient

- Pour chaque point  $w$ , il y a **une** direction où  $J$  monte le plus. On l'appelle le **gradient de  $J$  en  $w$** .



# Le gradient

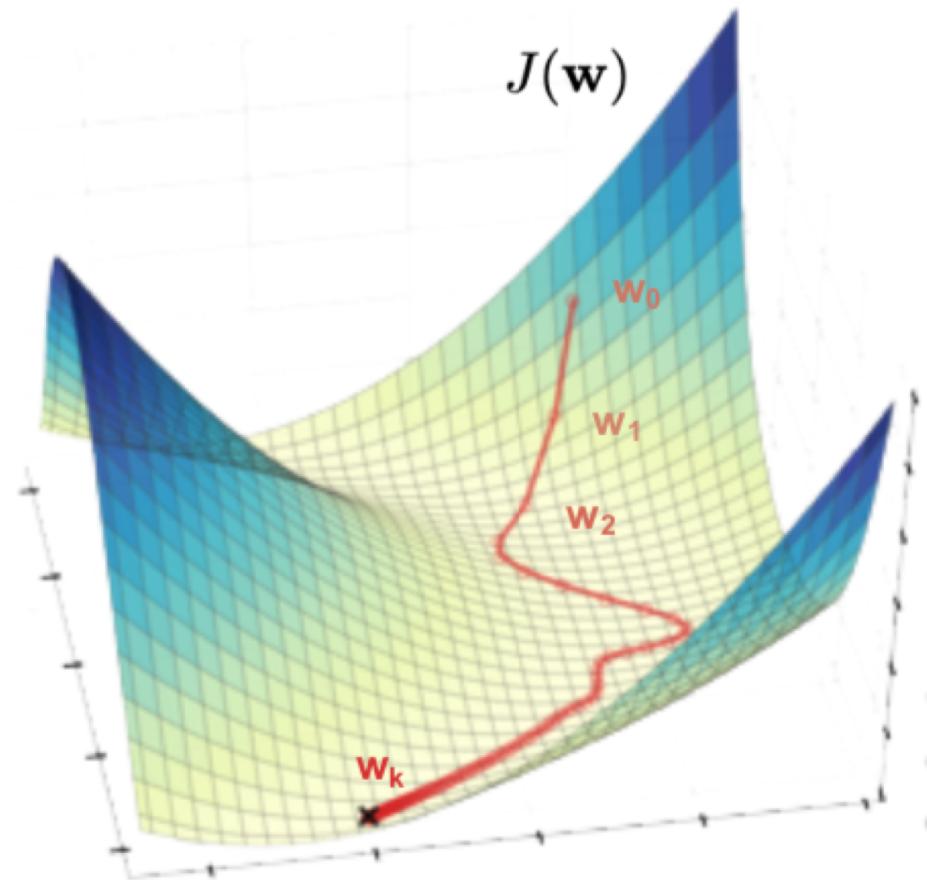
- L'opposée du gradient, c'est la direction où la fonction **descend** le plus.



# L'algorithme

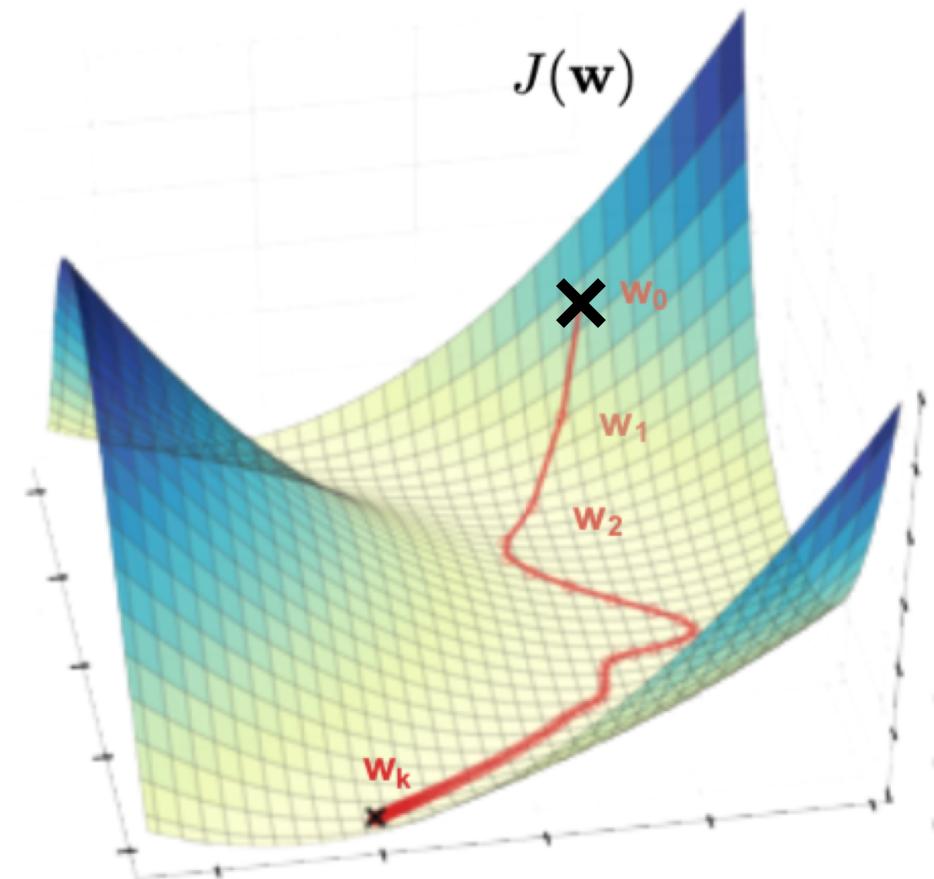
# Descente de gradient : l'algorithme

Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



# Descente de gradient : l'algorithme

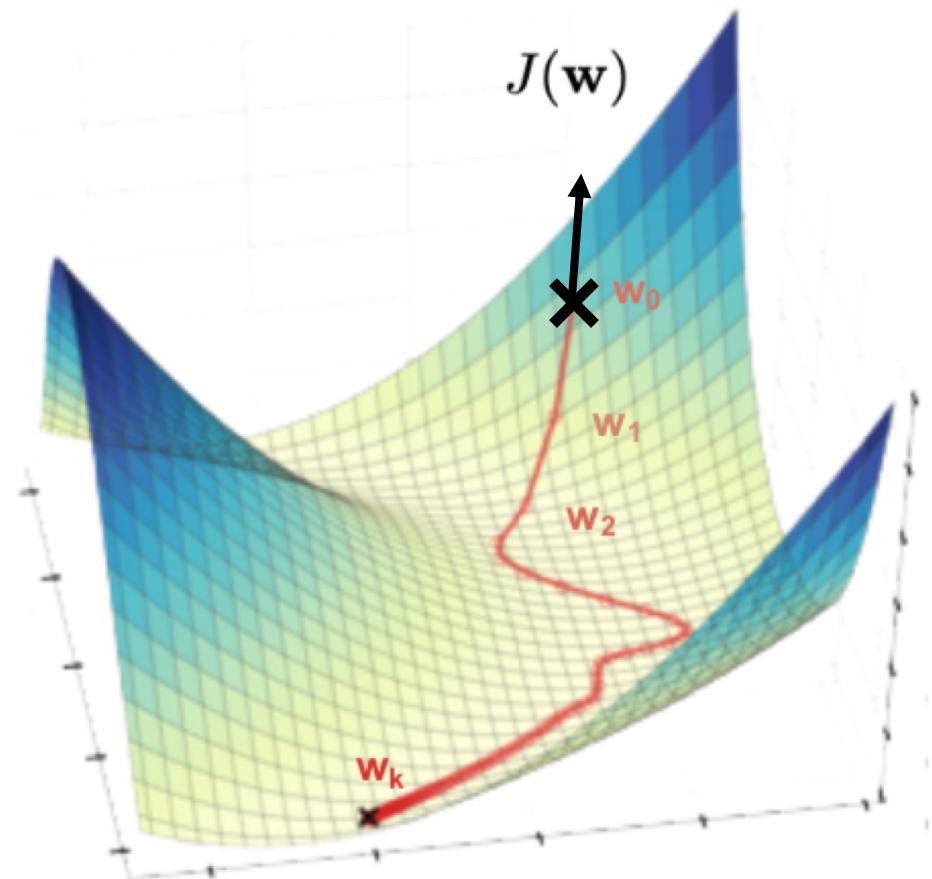
Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



1. Définir un point  $\theta$  quelconque

# Descente de gradient : l'algorithme

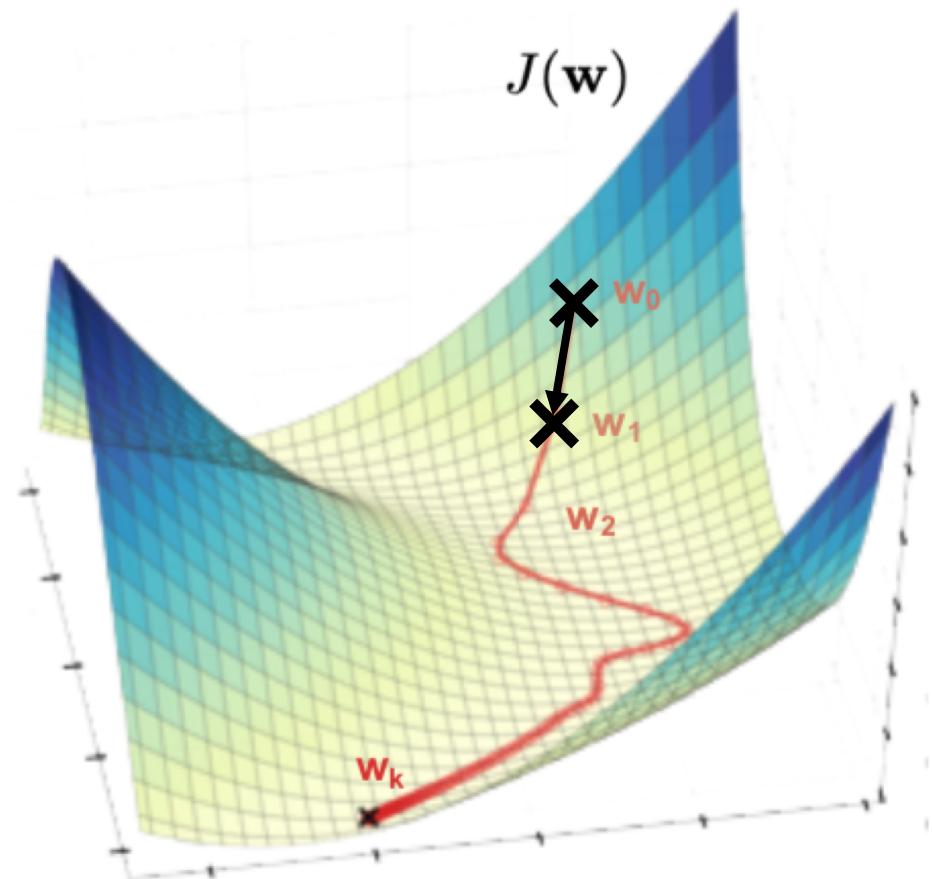
Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



1. Définir un point  $\theta$  quelconque
2. Calculer le gradient  $\nabla E(\theta)$

# Descente de gradient : l'algorithme

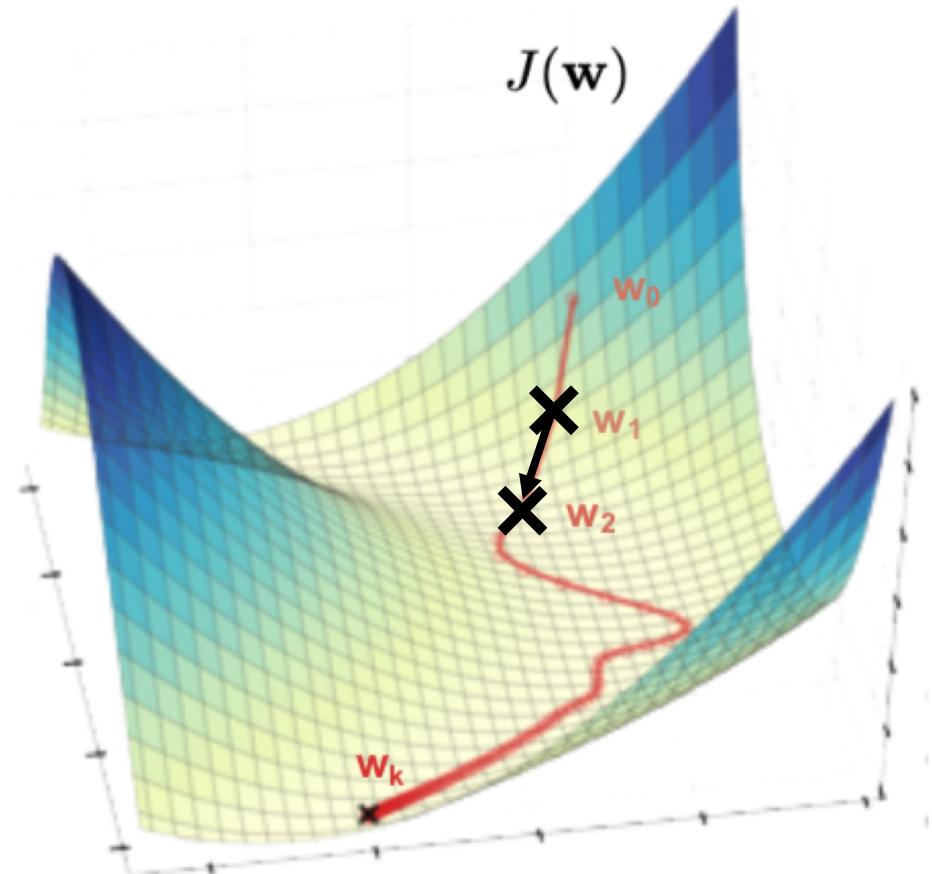
Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



1. Définir un point  $\theta$  quelconque
2. Calculer le gradient  $\nabla E(\theta)$
3.  $\theta$  prend la valeur  $\theta - \nabla E(\theta)$

# Descente de gradient : l'algorithme

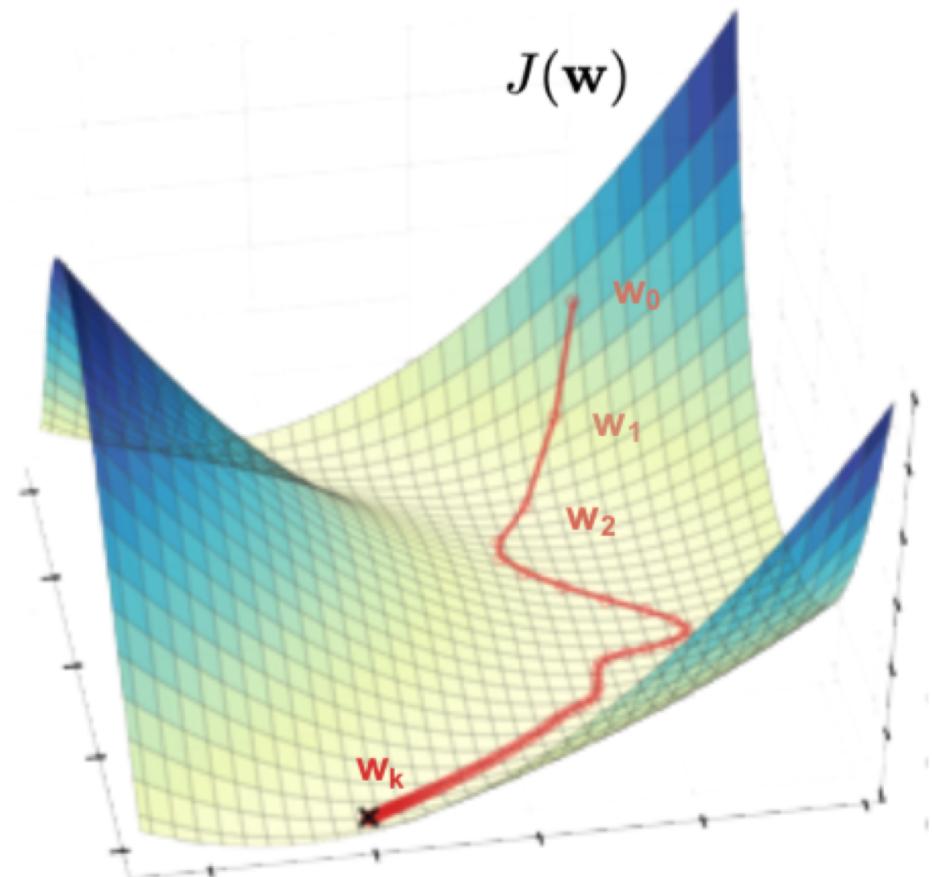
Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



1. Définir un point  $\theta$  quelconque
2. Calculer le gradient  $\nabla E(\theta)$
3.  $\theta$  prend la valeur  $\theta - \nabla E(\theta)$
4. Recommencer à partir de l'étape 2

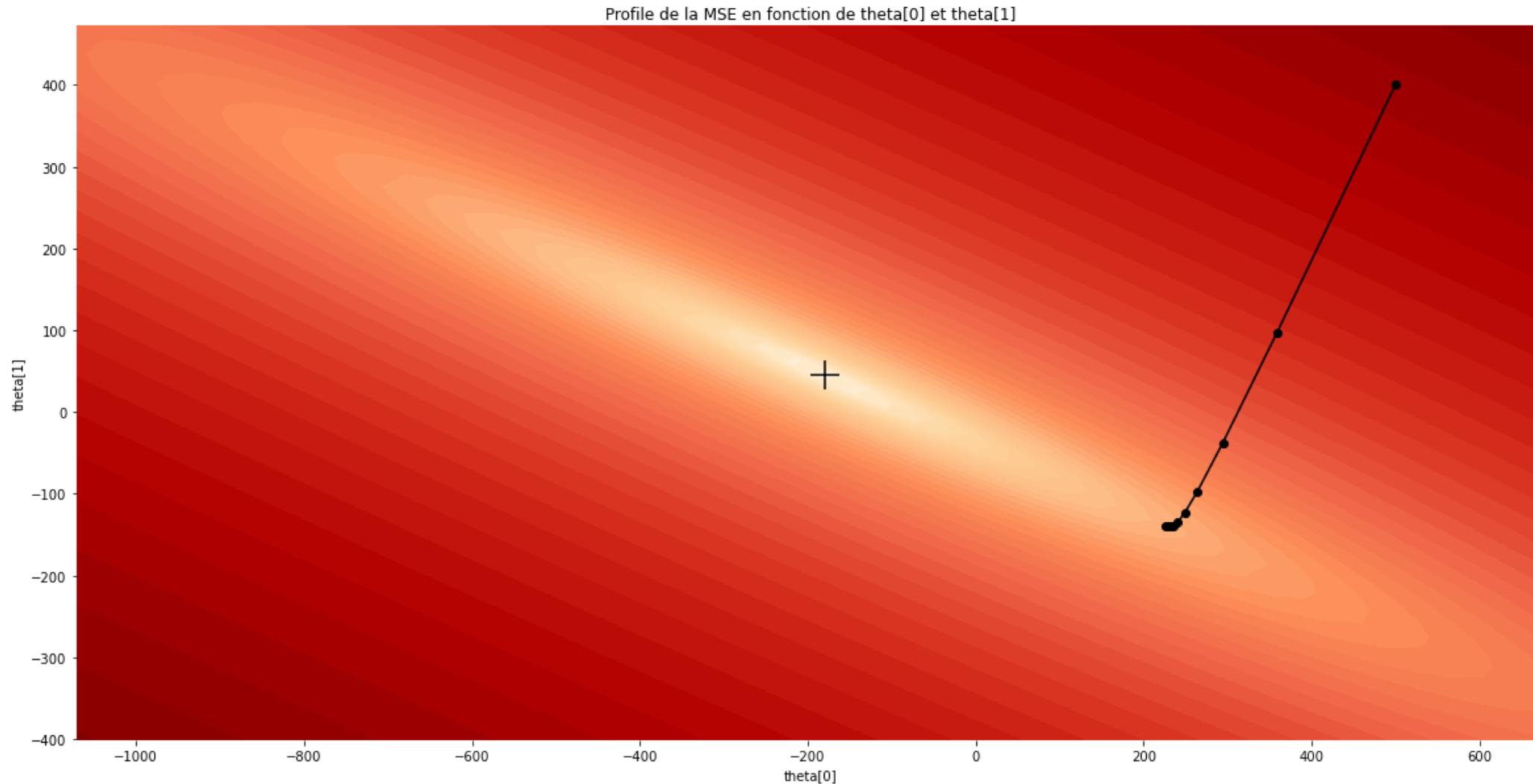
# Descente de gradient : l'algorithme

Appliquons l'algorithme du gradient à une fonction  $E(\theta)$



1. Définir un point  $\theta$  quelconque
2. Calculer le gradient  $\nabla E(\theta)$
3.  $\theta$  prend la valeur  $\theta - \nabla E(\theta)$
4. Recommencer à partir de l'étape 2
5. S'arrêter au bout de 100 étapes

# La descente de gradient en action



# Le code

# Calculer le gradient

A chaque étape on doit calculer  $\nabla E(\theta)$ , le gradient de la fonction  $E$

$$E : \theta \mapsto \sum_i (f(\theta, x_i) - y_i)^2$$

Diagram illustrating the components of the function  $E$ :

- Curved arrow pointing to  $f(\theta, x_i)$ : "la prédiction du modèle..."
- Curved arrow pointing to  $y_i$ : "qui dépend de ses paramètres..."
- Curved arrow pointing to the squared difference term: "et des données d'entrée"
- Curved arrow pointing to the right: "comparée à la vraie valeur qu'il fallait trouver"

Ca peut se faire à la main... Mais c'est pas très rigolo. Il faut calculer des dérivées.

On va utiliser un module python qui nous donne la fonction  $\nabla E$ .

$\nabla E$  est donc une fonction qui prend un paramètre  $\theta$  et qui retourne un vecteur qui a autant de dimensions que  $\theta$

# Calculer le gradient

Le module qu'on va utiliser s'appelle `autograd`

`autograd.grad(E)` retourne la fonction gradient  $\nabla E$ , qu'on utilise ensuite à chaque étape de l'algorithme du gradient

Pour que ça marche, il faut que  $E$  soit une fonction qui ne prend que un `np.array` en entrée.

On devra donc recoder notre fonction `get_mse`

# À vous de jouer

- Ce qu'on a vu :
  - Ce que c'est qu'un gradient et comment on le calcule
  - Le principe de l'algorithme de descente de gradient pour trouver des bons paramètres
- Ce qu'on va voir en TD :
  - Réimplémenter l'algorithme du gradient
  - L'utiliser pour entraîner une régression linéaire sur des données simples