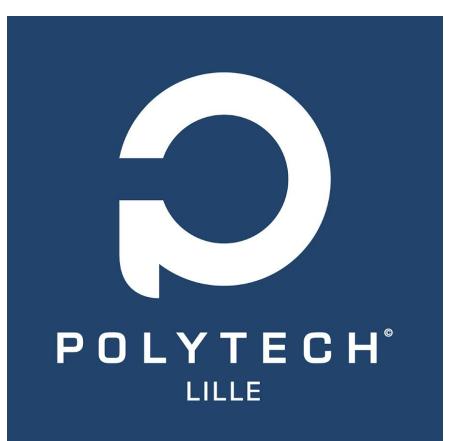


Introduction au Deep Learning

par Thomas Besnier, d'après le cours de Kevin
Feghouli

CRISyAL - Université de Lille

2023-2024



Contact

Mail: thomas.besnier@univ-lille.fr

Support cours/TPs: <https://tbesnier.github.io/teaching/>

Objectifs du cours (22h)

- Être capable d'expliquer comment fonctionne le deep learning dans les grandes lignes
- Identifier des cas d'utilisation: quand faire du deep ?
- Etre capable d'identifier les blocs principaux d'un algo + coder des modèles de base

Objectifs du cours (22h)

- Être capable d'expliquer comment fonctionne le deep learning dans les grandes lignes
- Identifier des cas d'utilisation: quand faire du deep ?
- Etre capable d'identifier les blocs principaux d'un algo + coder des modèles de base

Évaluation: **TPs (40%) + Exam (60%)**

Programme du cours

- Introduction et contexte
- Réseaux de neurones et optimisation
- Deep Computer Vision
- Deep Generative model
- Deep Sequence model

Programme du cours

- Introduction et contexte
- Réseaux de neurones et optimisation
- Deep Computer Vision
- Deep Generative model
- Deep Sequence model

Qu'est ce que “le deep” ?

Qu'est ce que “le deep” ?

- Une représentation complexe des données + grande quantité**

Qu'est ce que “le deep” ?

- Une représentation complexe des données + grande quantité
- **Un (très) gros modèle statistique: réseaux de neurones**

Qu'est ce que “le deep” ?

- Une représentation complexe des données + grande quantité
- Un (très) gros modèle statistique: réseaux de neurones
- **Differentiation automatique**

Qu'est ce que “le deep” ?

Intelligence Artificielle

Any techniques that imitate human Intelligence

Machine Learning

Statistical algorithm that learns patterns from data

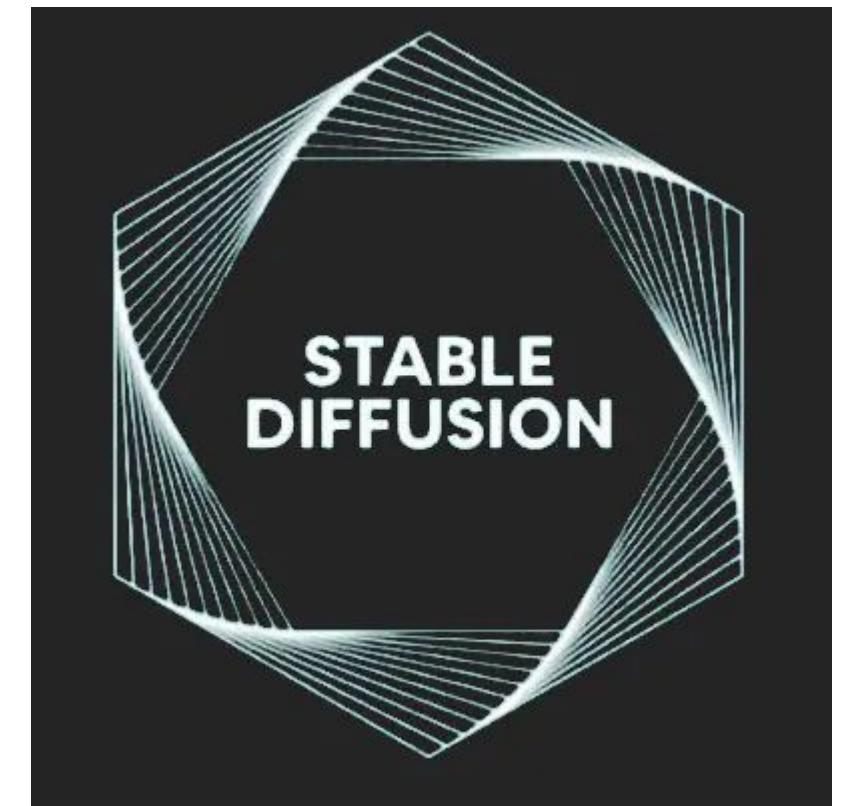
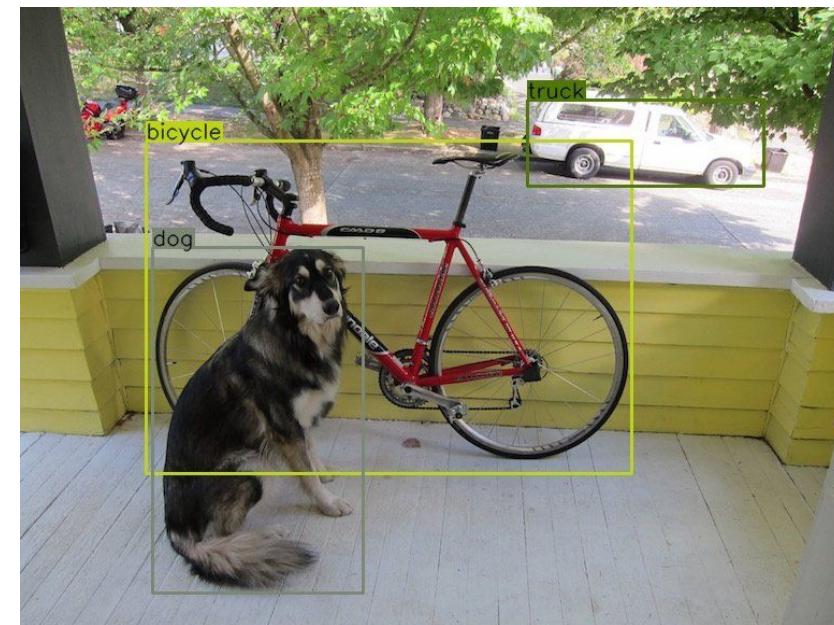
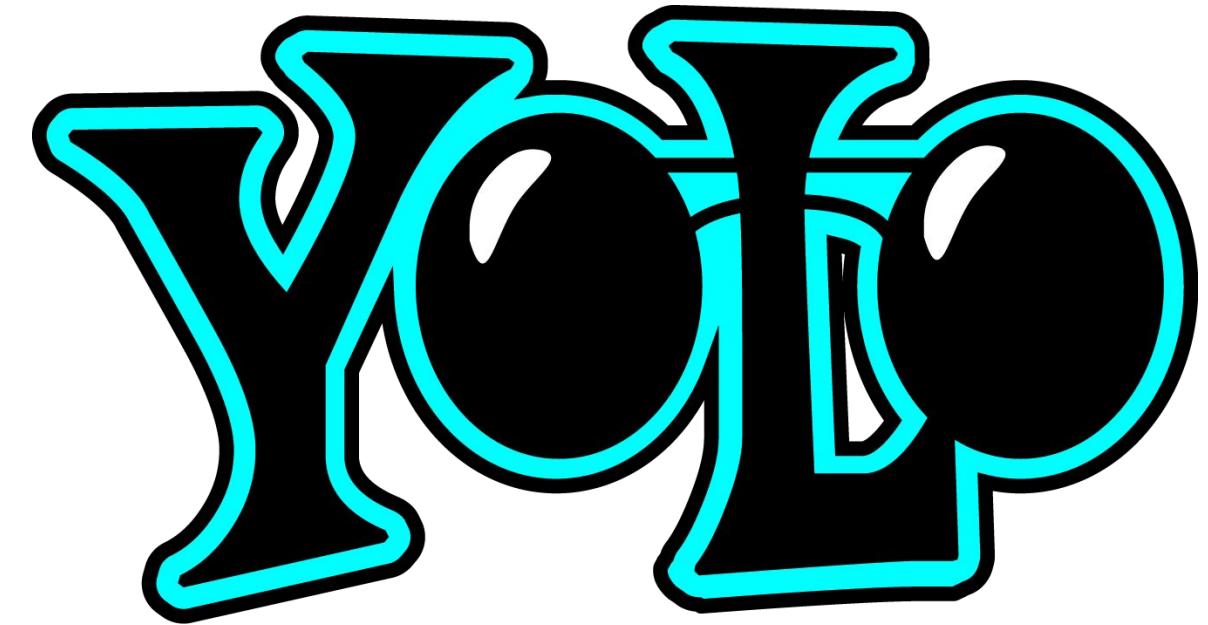
Deep Learning

Learns from data using neural network

Pourquoi maintenant ?

Pourquoi maintenant ?

- Meilleurs algo



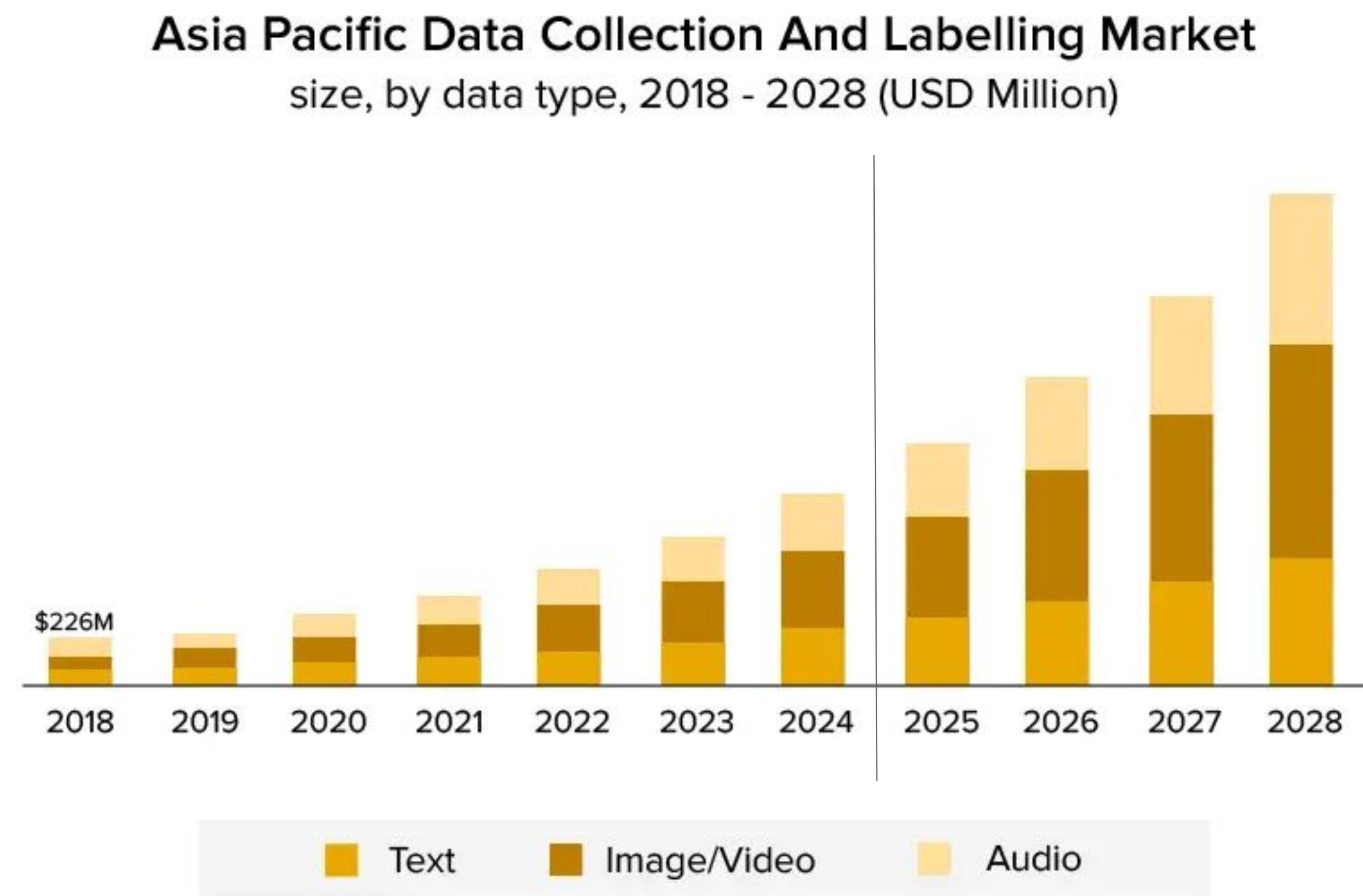
Pourquoi maintenant ?

- Meilleurs algo
- Meilleurs hardware



Pourquoi maintenant ?

- Meilleurs algo
- Meilleurs hardware
- Plus de données labellisées



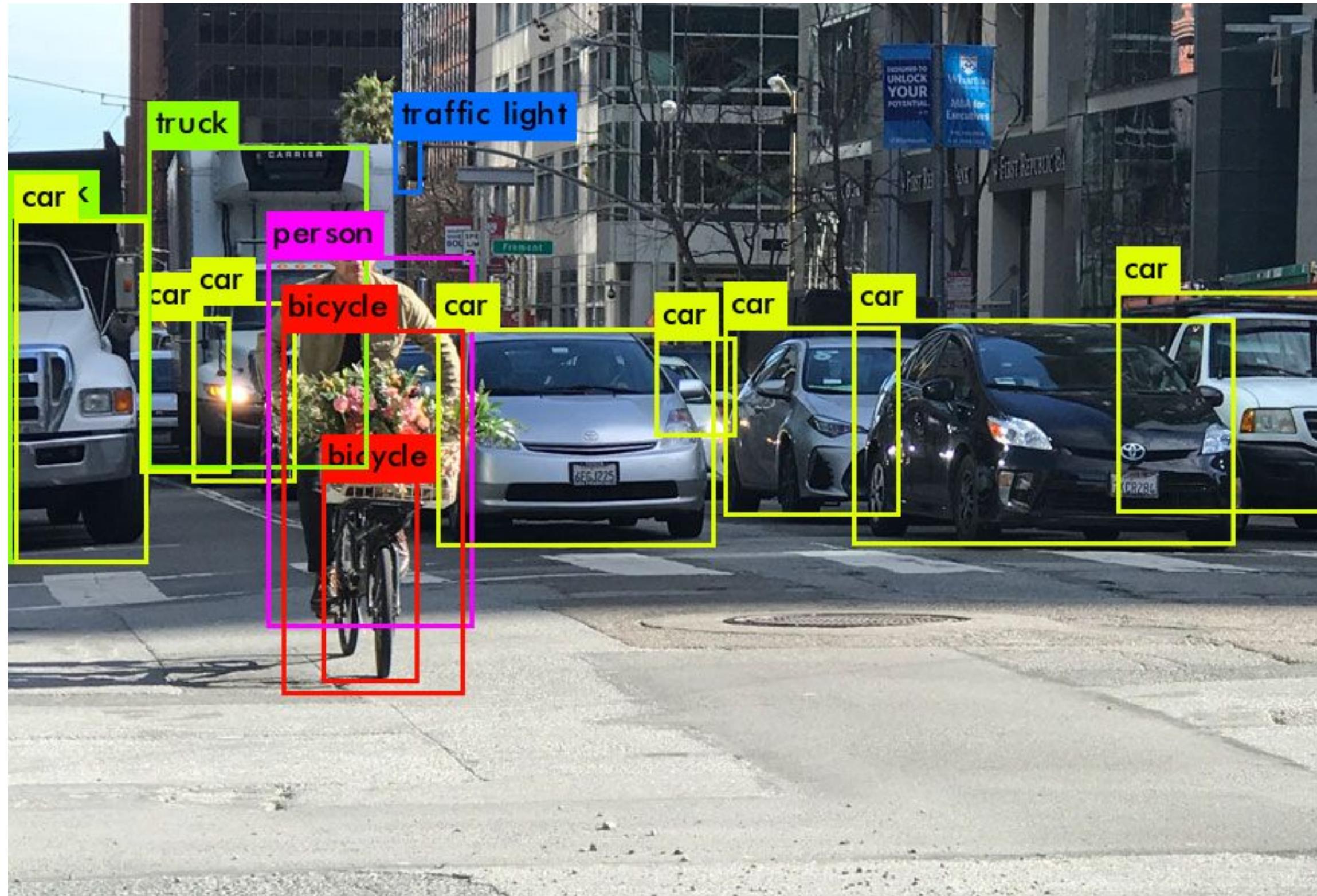
Pourquoi maintenant ?

- Meilleurs algo
- Meilleurs hardware
- Plus de données labellisées
- Outils open source et accessibles**



Progrès récents

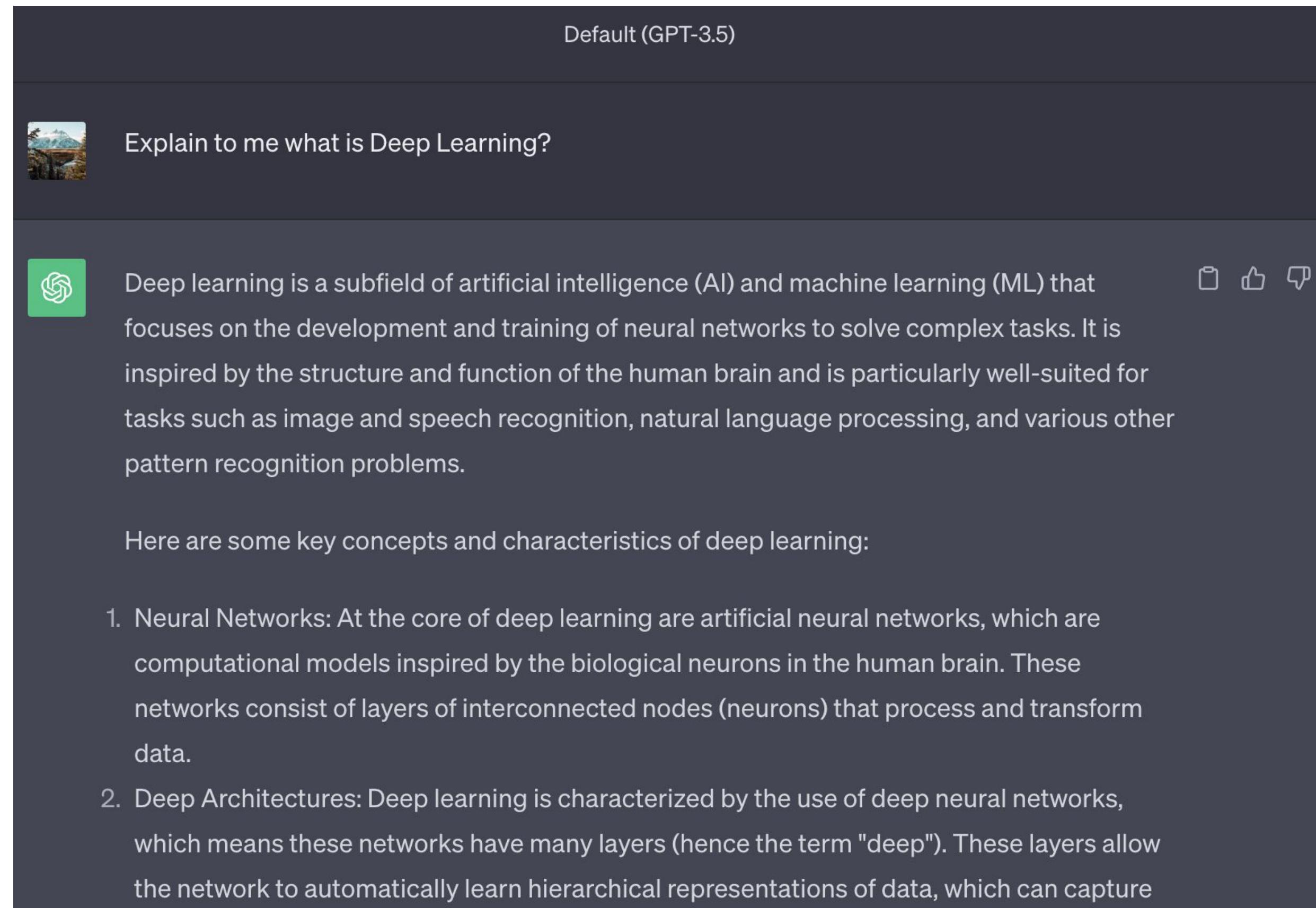
Compréhension d'images



Modèles génératifs: ChatGPT

Prompt to text generation

Default (GPT-3.5)



Explain to me what is Deep Learning?

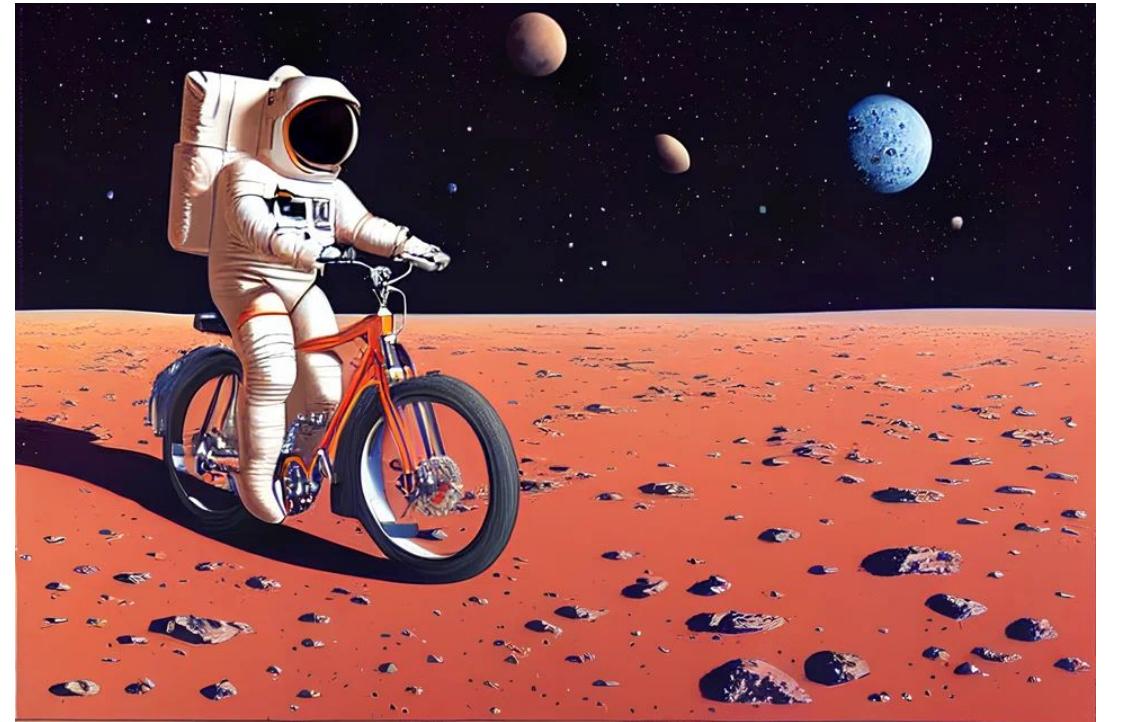
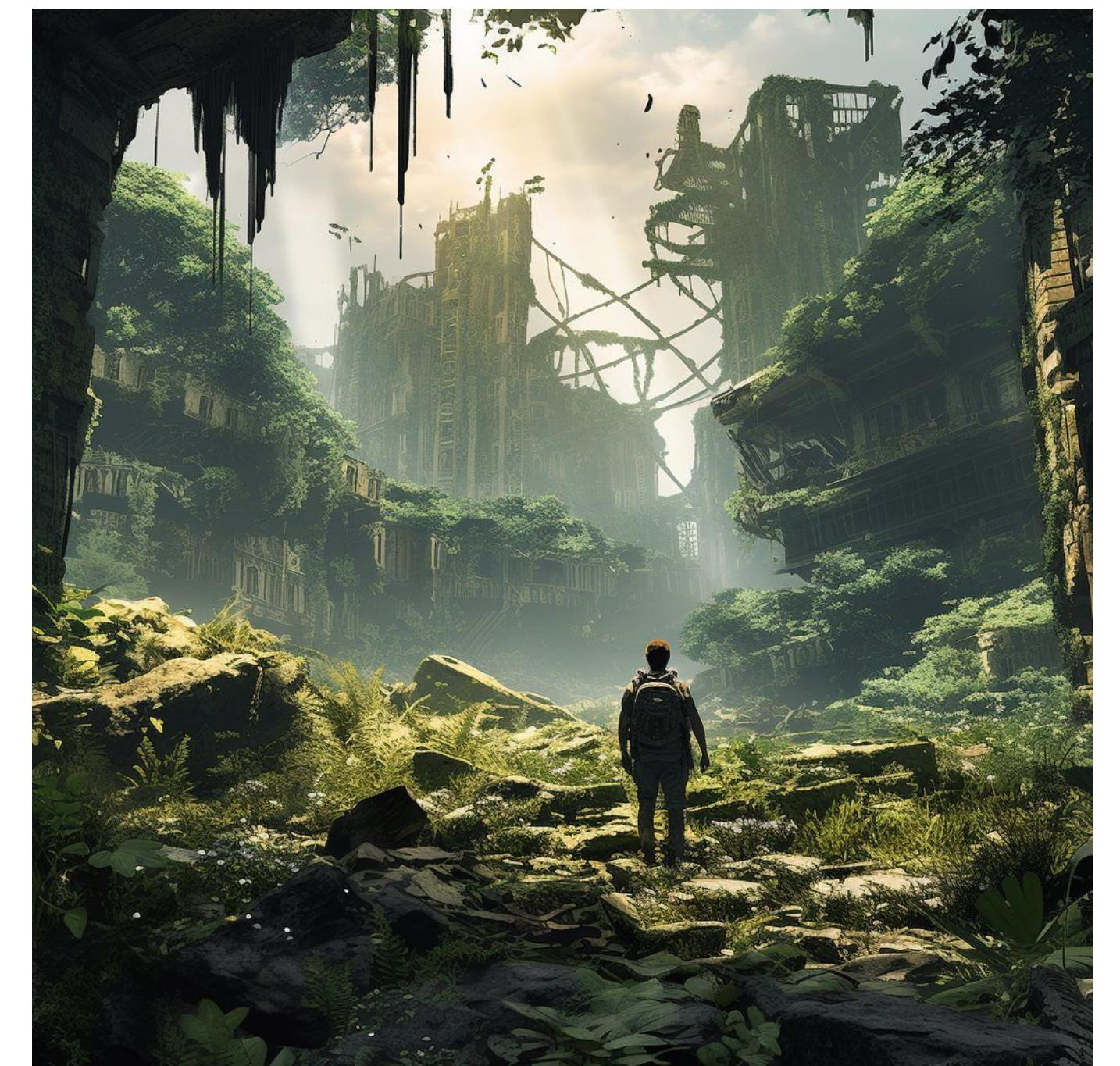
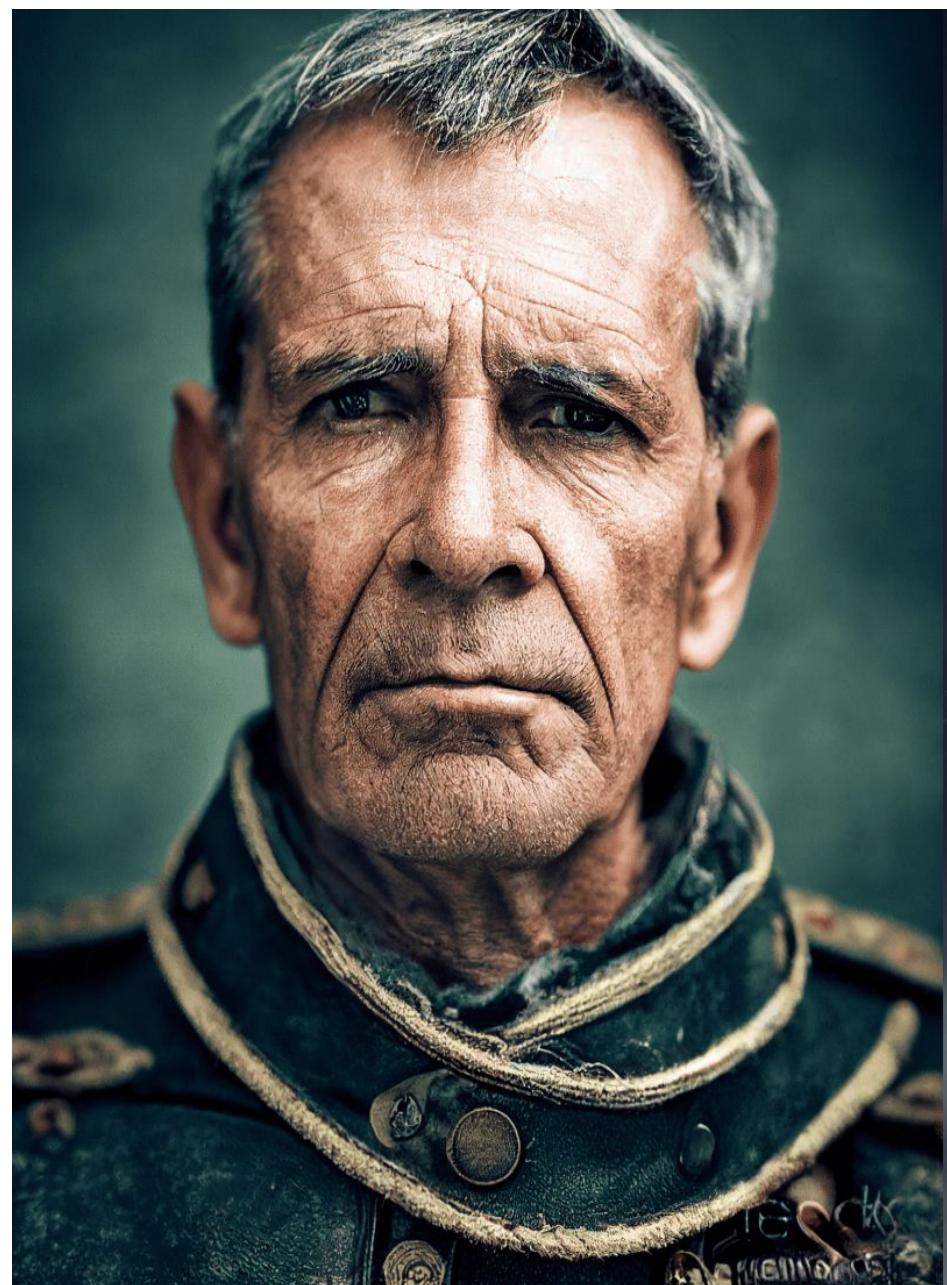
Deep learning is a subfield of artificial intelligence (AI) and machine learning (ML) that focuses on the development and training of neural networks to solve complex tasks. It is inspired by the structure and function of the human brain and is particularly well-suited for tasks such as image and speech recognition, natural language processing, and various other pattern recognition problems.

Here are some key concepts and characteristics of deep learning:

1. Neural Networks: At the core of deep learning are artificial neural networks, which are computational models inspired by the biological neurons in the human brain. These networks consist of layers of interconnected nodes (neurons) that process and transform data.
2. Deep Architectures: Deep learning is characterized by the use of deep neural networks, which means these networks have many layers (hence the term "deep"). These layers allow the network to automatically learn hierarchical representations of data, which can capture

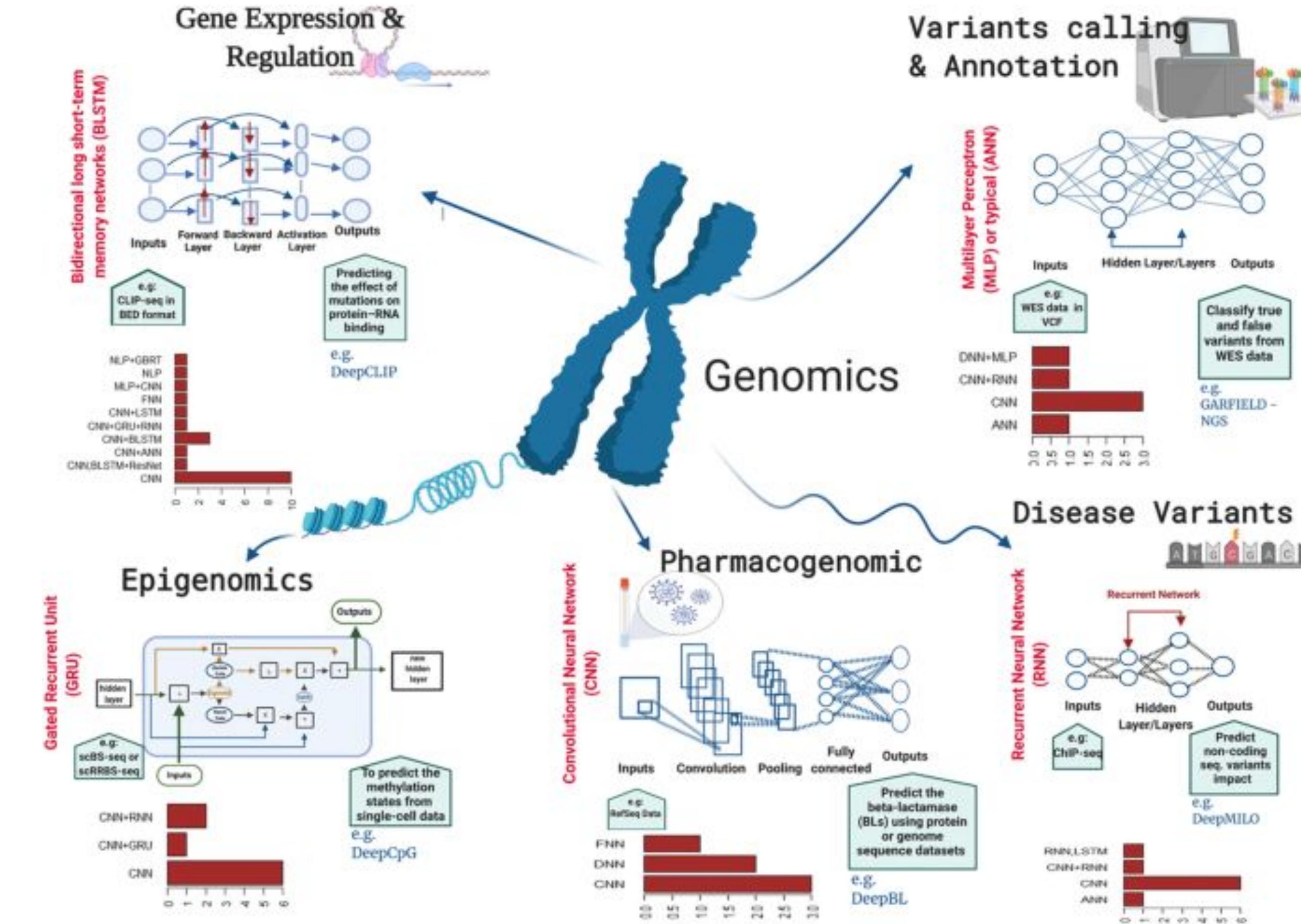
Modèle génératif: Stable Diffusion

Text-to-image generation



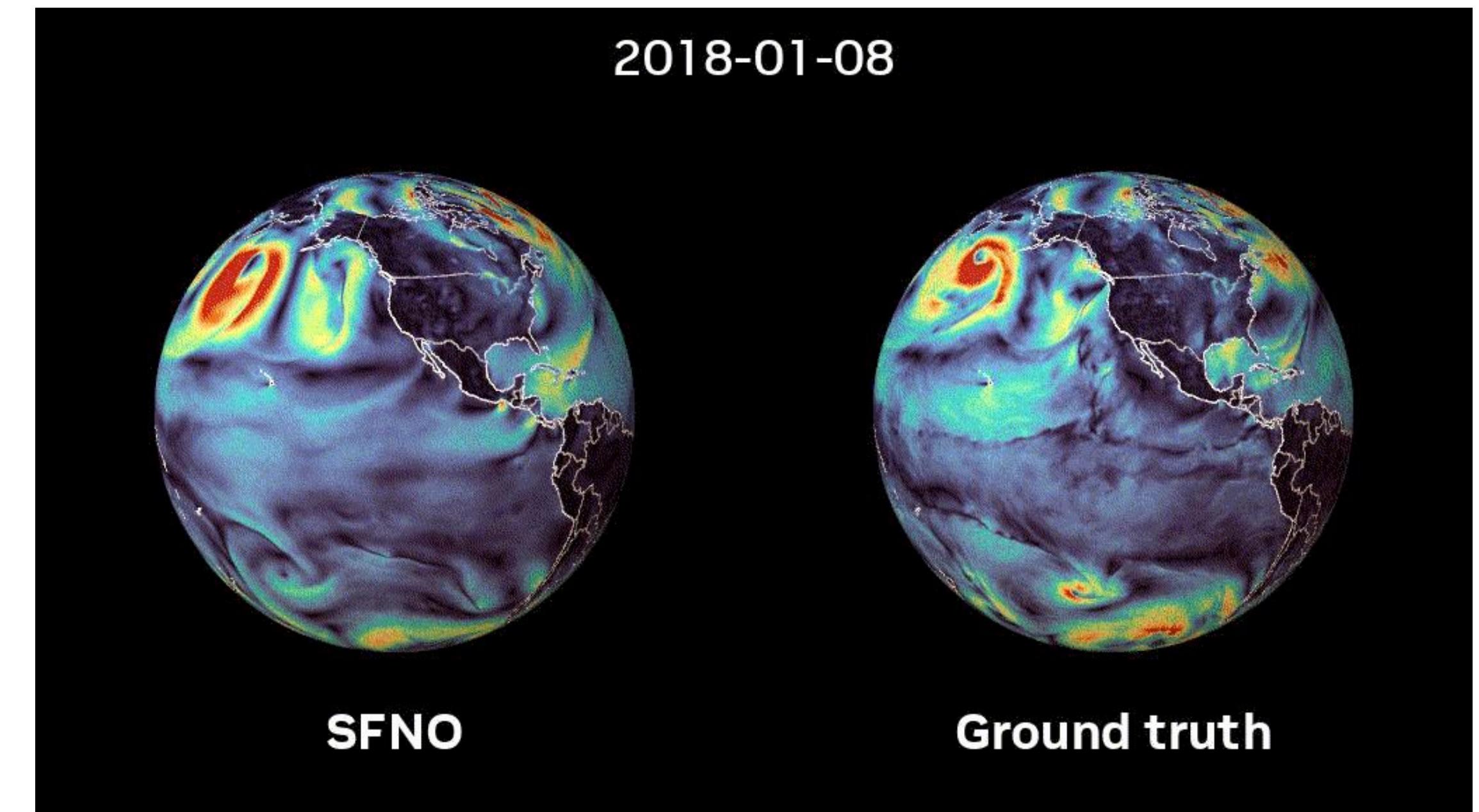
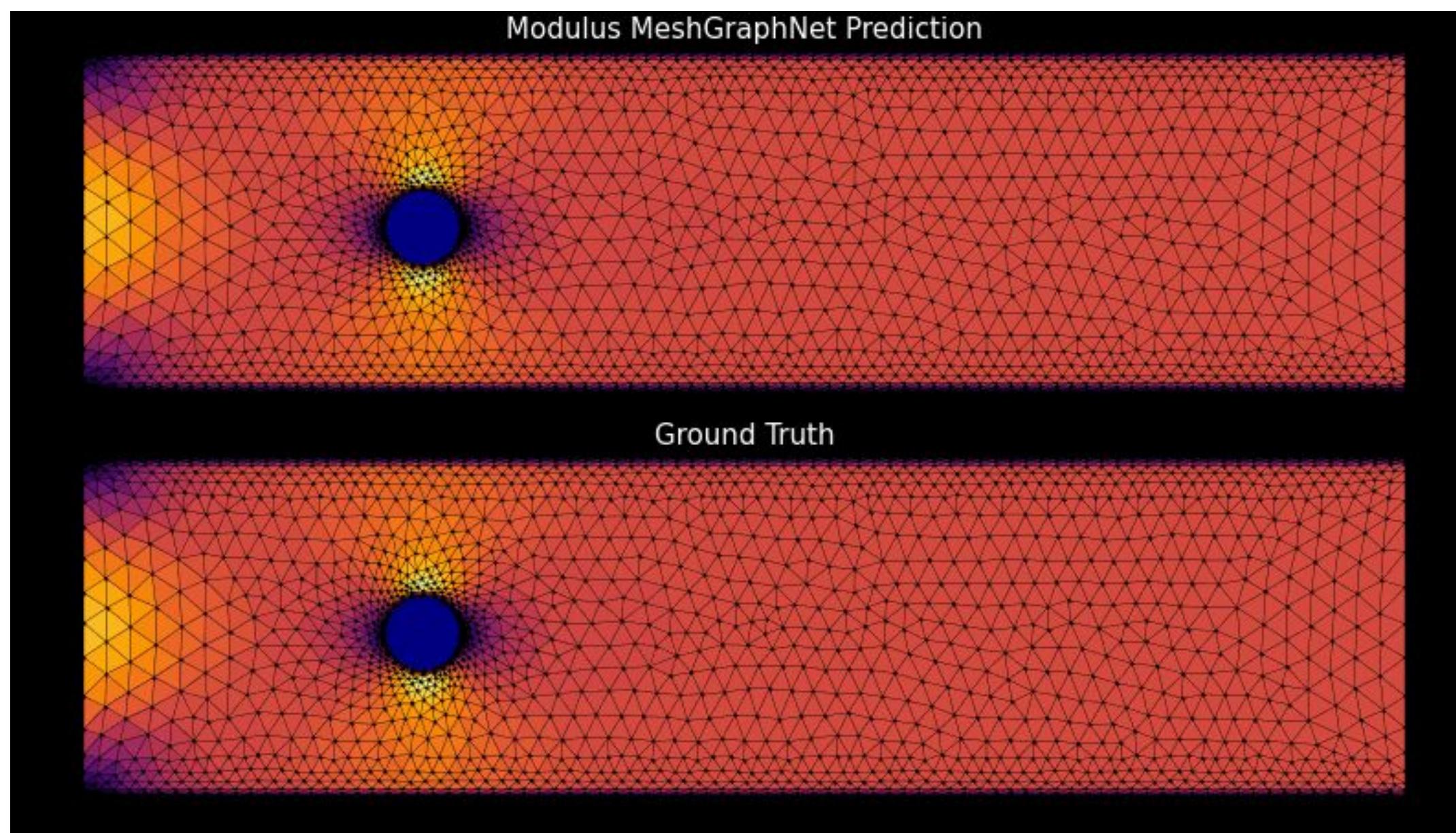
Utilisation en sciences

Génomique



Utilisation en sciences

Simulation en chimie / Physique



Comment ça marche ?

Le but de l'apprentissage supervisé est de construire une fonction de décision f qui fonctionne bien, vite et garantie de bonne performance :

Comment ça marche ?

Le but de l'apprentissage supervisé est de construire une fonction de décision f qui fonctionne bien, vite et garantie de bonne performance :

$$\begin{aligned} f & : \mathcal{X} \rightarrow \mathcal{Y} \\ x & \mapsto y = f(x) \end{aligned}$$

$$\mathcal{Y} = \mathbb{R}$$

Problème de régression

$$\mathcal{Y} = \{0, 1\}$$

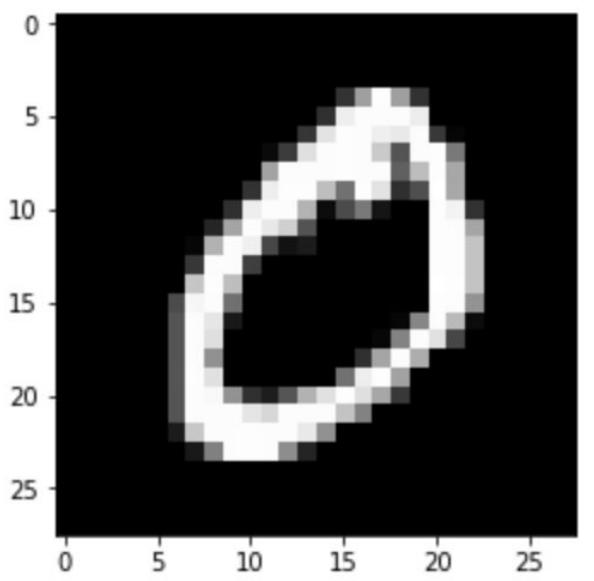
Problème de classification binaire

$$\mathcal{Y} = \{1, 2, \dots, C\}, C > 2$$

Problème de classification multi-classe

Comment ça marche ?

0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 0
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 1
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9

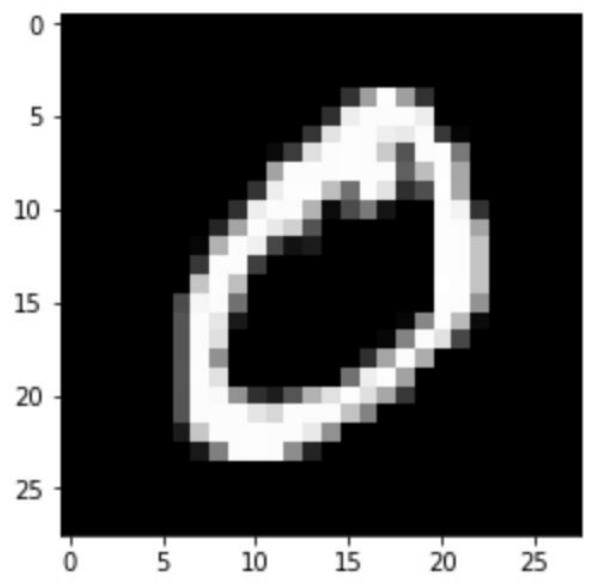


$x \in$

$y \in$

Comment ça marche ?

0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 0
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 1
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9



$$x \in \{0, \dots, 1, \dots, 0\}^{28 \times 28}$$

$$y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Comment ça marche ?

Cat



Dog



$x \in$

$y \in$

f



$$y = f(x)$$

Comment ça marche ?

Cat



Dog



f



$$y = f(x)$$

$$x \in \{0, \dots, 1, \dots, 0\}^{28 \times 28 \times 3}$$

$$y \in \{0, 1\}$$

Comment ça marche ?



$$f \longrightarrow$$

$$y = f(x) = 254,500 \text{ €}$$

$x \in$

$y \in$

Comment ça marche ?



f



$$y = f(x) = 254,500 \text{ €}$$

$$x \in \{0, 1, \dots, 255\}^{4 \times 64 \times 64 \times 3}$$

$$y \in \mathbb{R}_+$$

Comment ça marche ?



f



$x \in$

$y \in$

Comment ça marche ?



f



$$x \in \{0, \dots, 255\}^{n \times m}$$

$$y \in \{0, 1\}^{n \times m}$$

Comment ça marche ?

Recette de l'entraînement d'un modèle

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle

1. Définir un objectif

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle

1. Définir un objectif

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle

1. Définir un objectif

1. Optimiser les paramètres

Comment ça marche ?

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle

1. Définir un objectif

1. Optimiser les paramètres

1. Evaluer

Comment ça marche ?

Recette de l'entraînement d'un modèle

- 
1. Préparer la donnée → Dataloader
 1. Définir / construire le modèle
 1. Définir un objectif
 1. Optimiser les paramètres
 1. Evaluer

Programme du cours

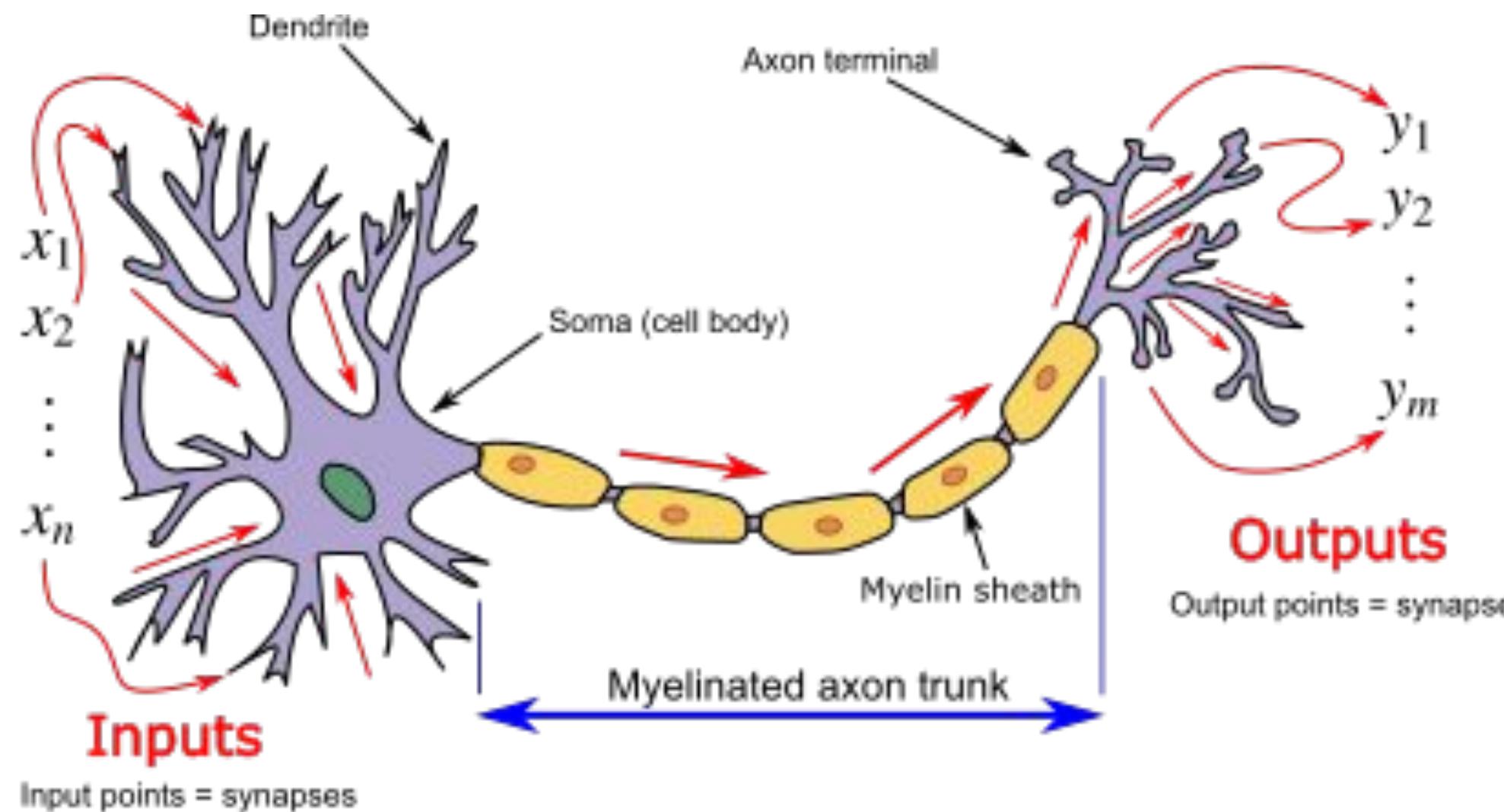
- Introduction et contexte
- Réseaux de neurones et optimisation
- Deep Computer Vision
- Deep Generative model
- Deep Sequence model

Comment ça marche ?

Neurone artificiel / Perceptron

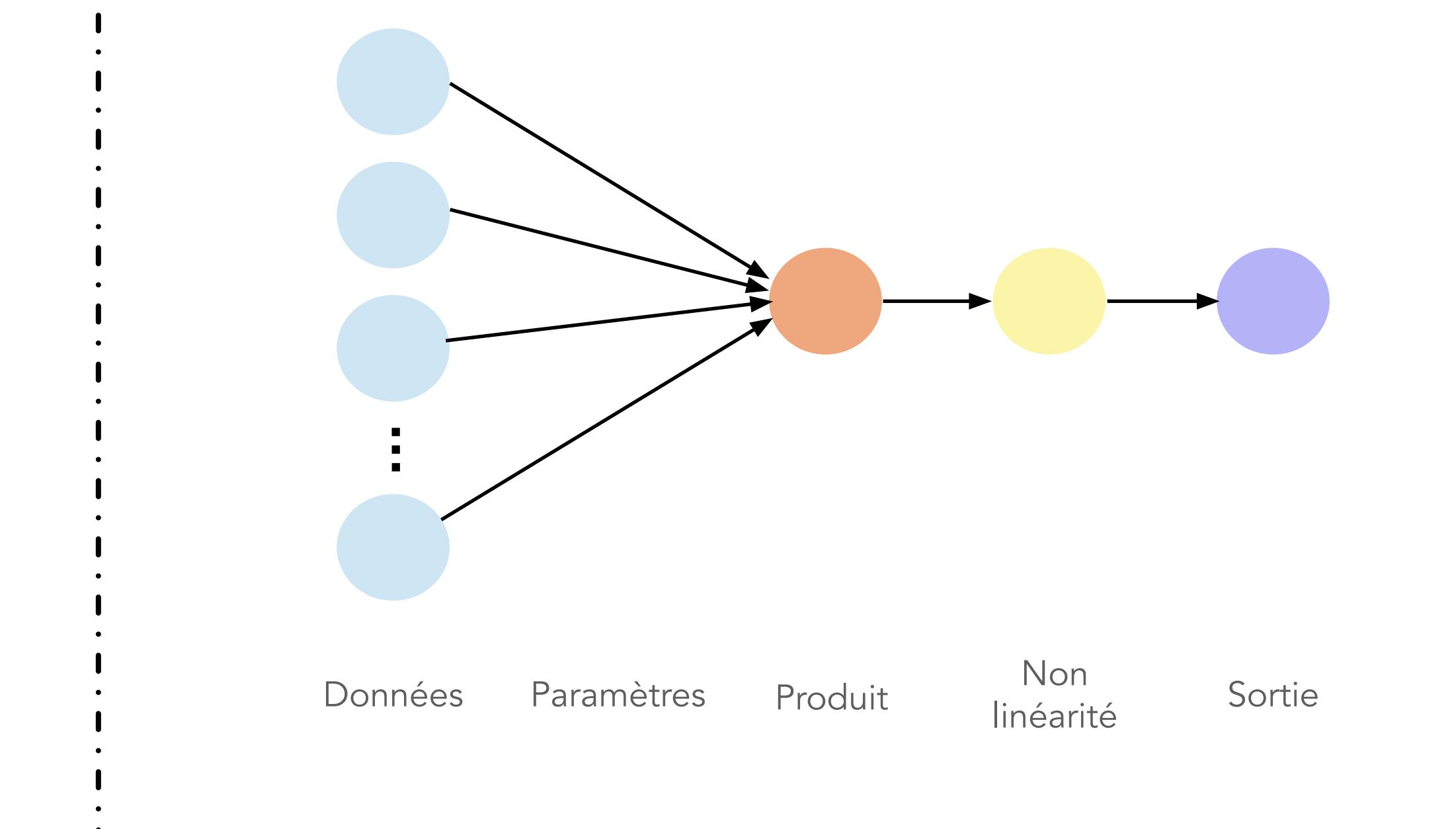
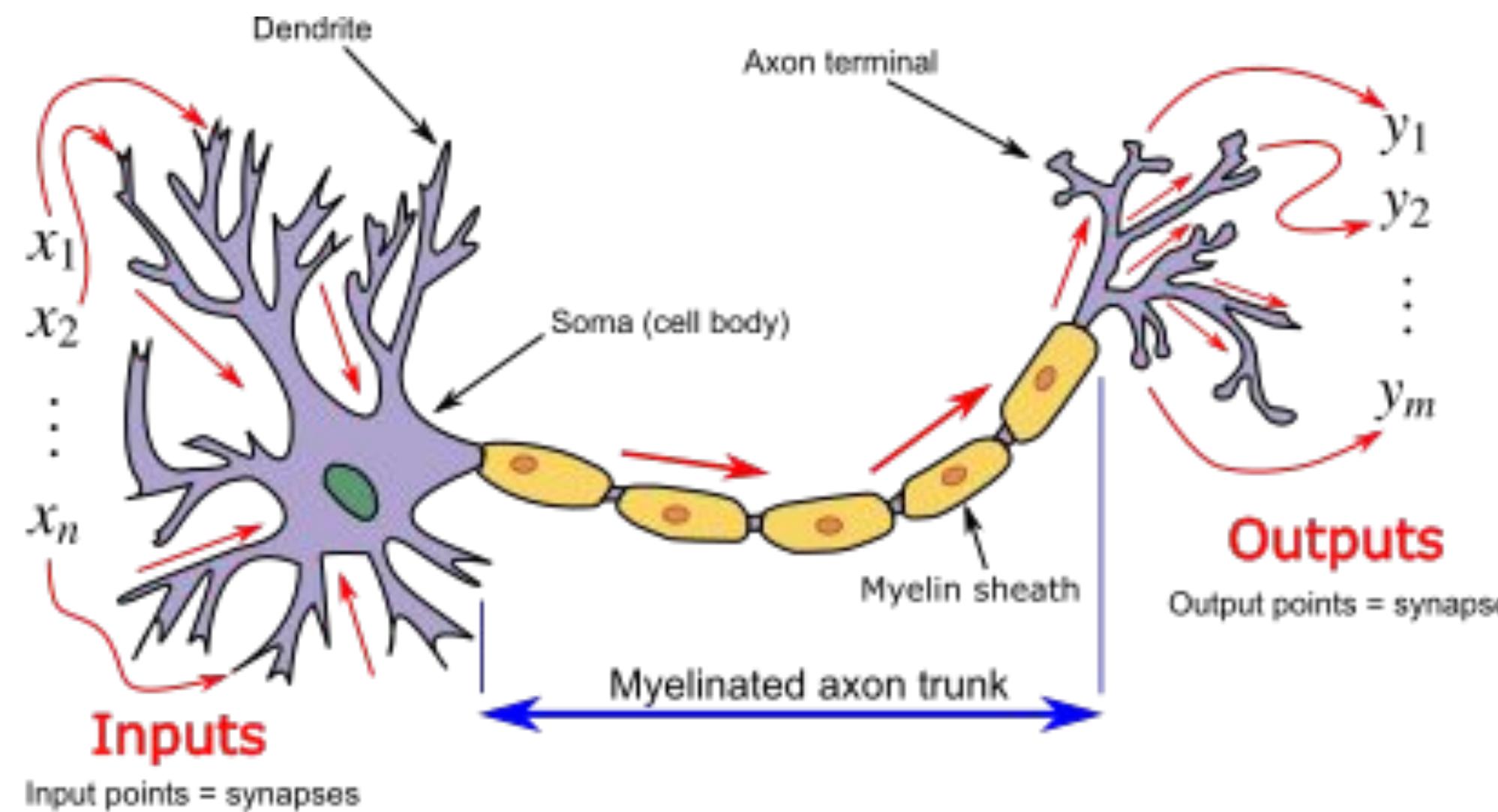
Comment ça marche ?

Neurone artificiel / Perceptron



- Soma: intègre les signaux provenant d'autres neurones.
- Dendrites : reçoivent les signaux d'autres neurones.
- Axone : transporte le signal vers d'autres neurones.
- Potentiel d'Action : signal électrique qui détermine si le neurone transmet de l'information à d'autres neurones ou non.
- Synapses : sont les connexions entre les neurones où les neurotransmetteurs sont libérés. Ces connexions déterminent la force du signal transmis entre les neurones.

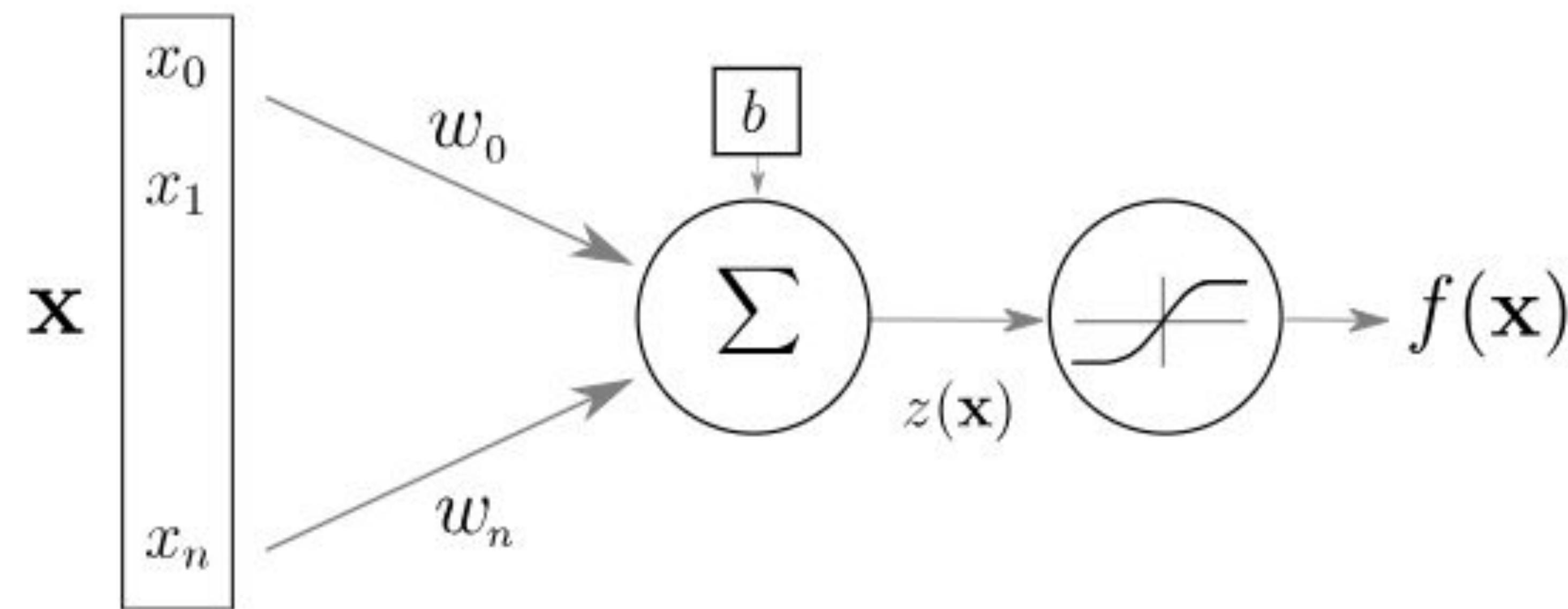
Analogie entre neurone biologique et formel



Le Perceptron

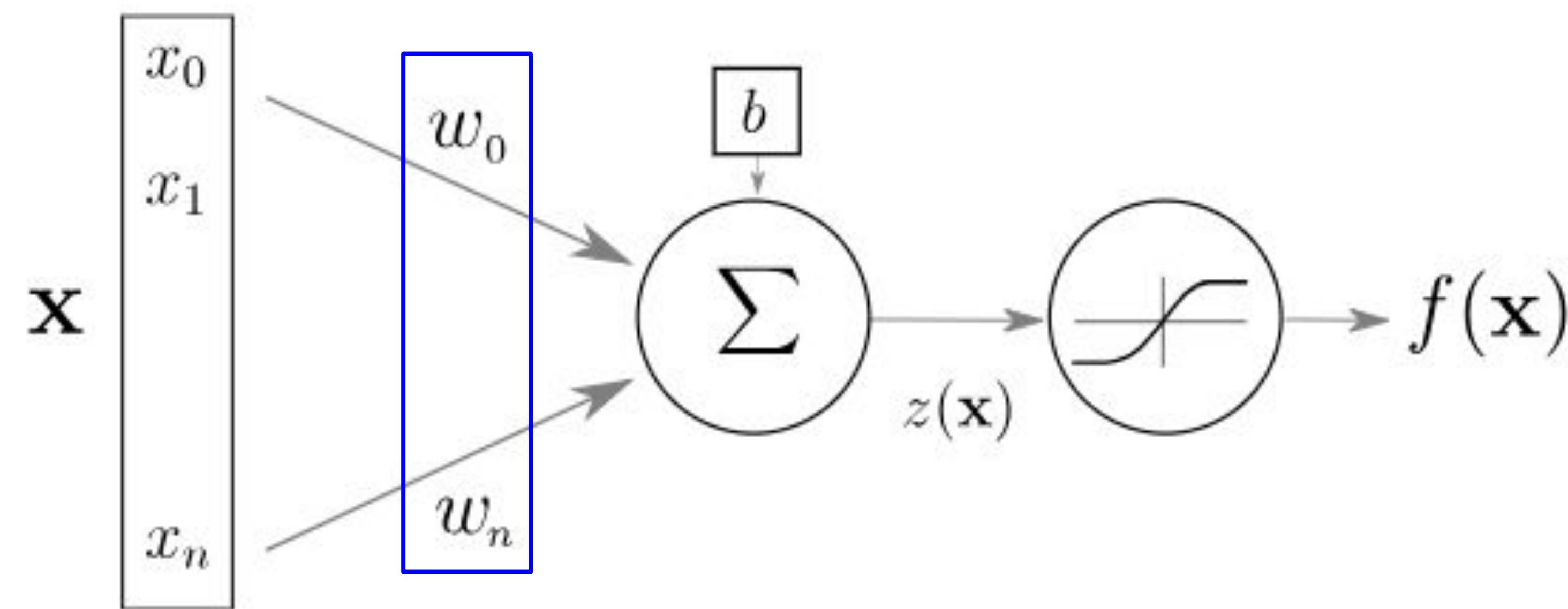
Comment ça marche ?

Neurone artificiel / Perceptron



Comment ça marche ?

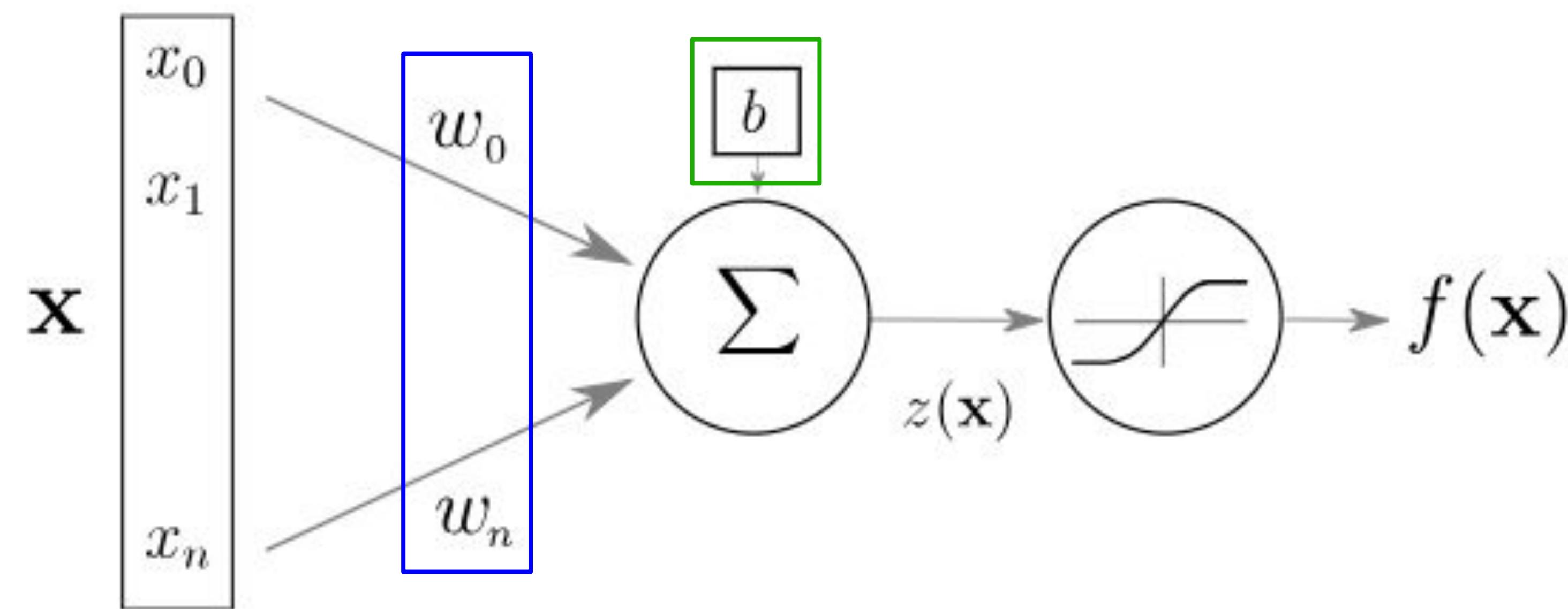
Neurone artificiel / Perceptron



$$f(\mathbf{x}) = \boxed{w^T} \mathbf{x}$$

Comment ça marche ?

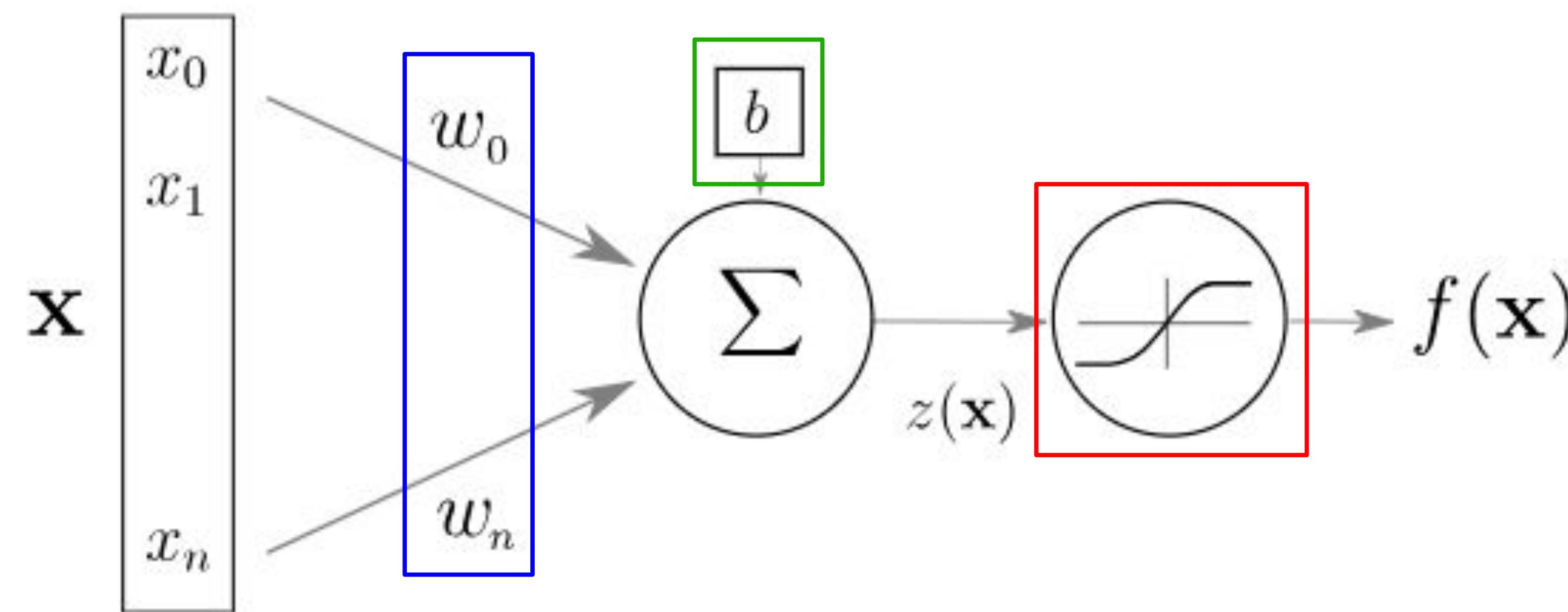
Neurone artificiel / Perceptron



$$f(\mathbf{x}) = \boxed{w^T} \mathbf{x} + \boxed{b}$$

Comment ça marche ?

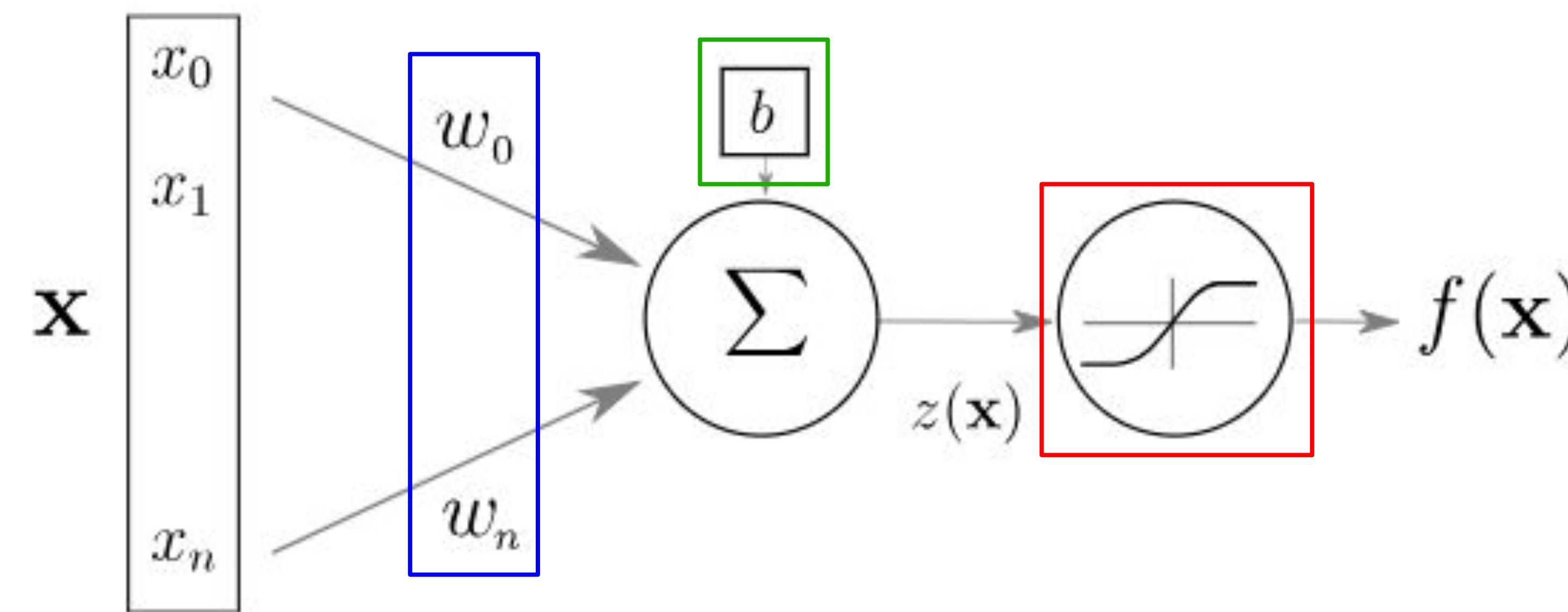
Neurone artificiel / Perceptron



$$f(\mathbf{x}) = g(w^T \mathbf{x} + b)$$

Comment ça marche ?

Neurone artificiel / Perceptron



$$f(\mathbf{x}) = g(w^T \mathbf{x} + b)$$

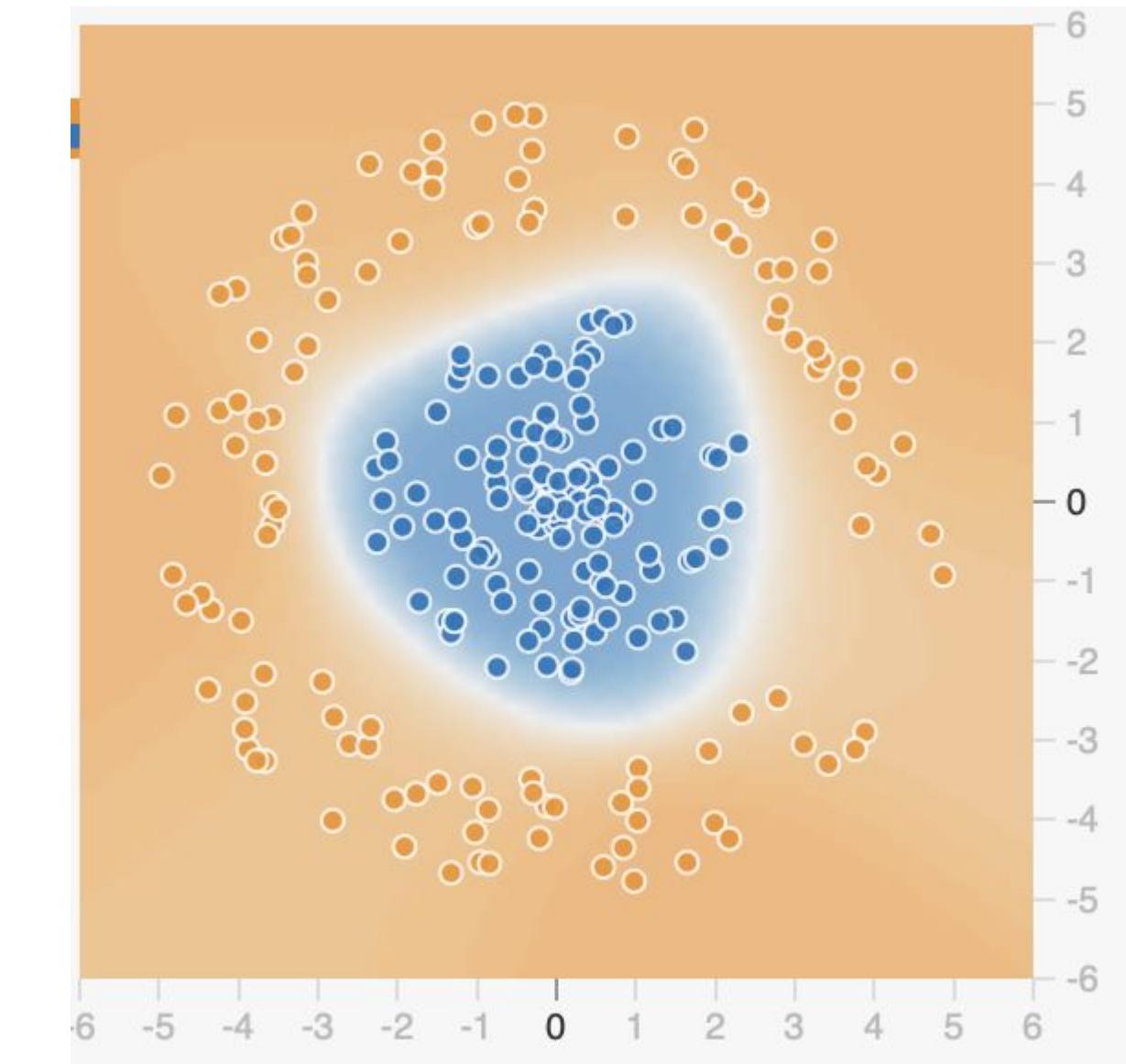
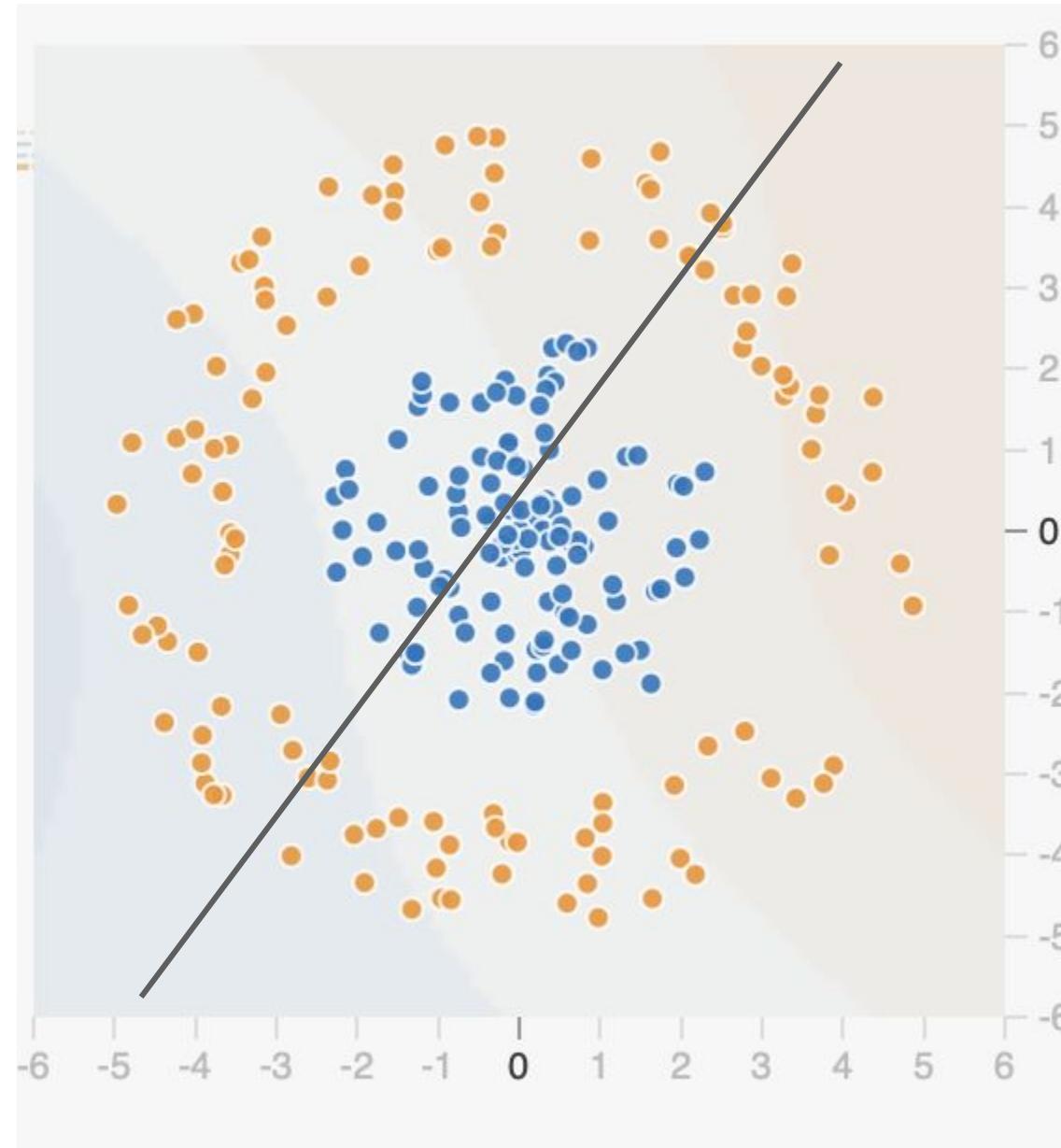
→ Régression logistique

Importance des fonctions d'activation

- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.

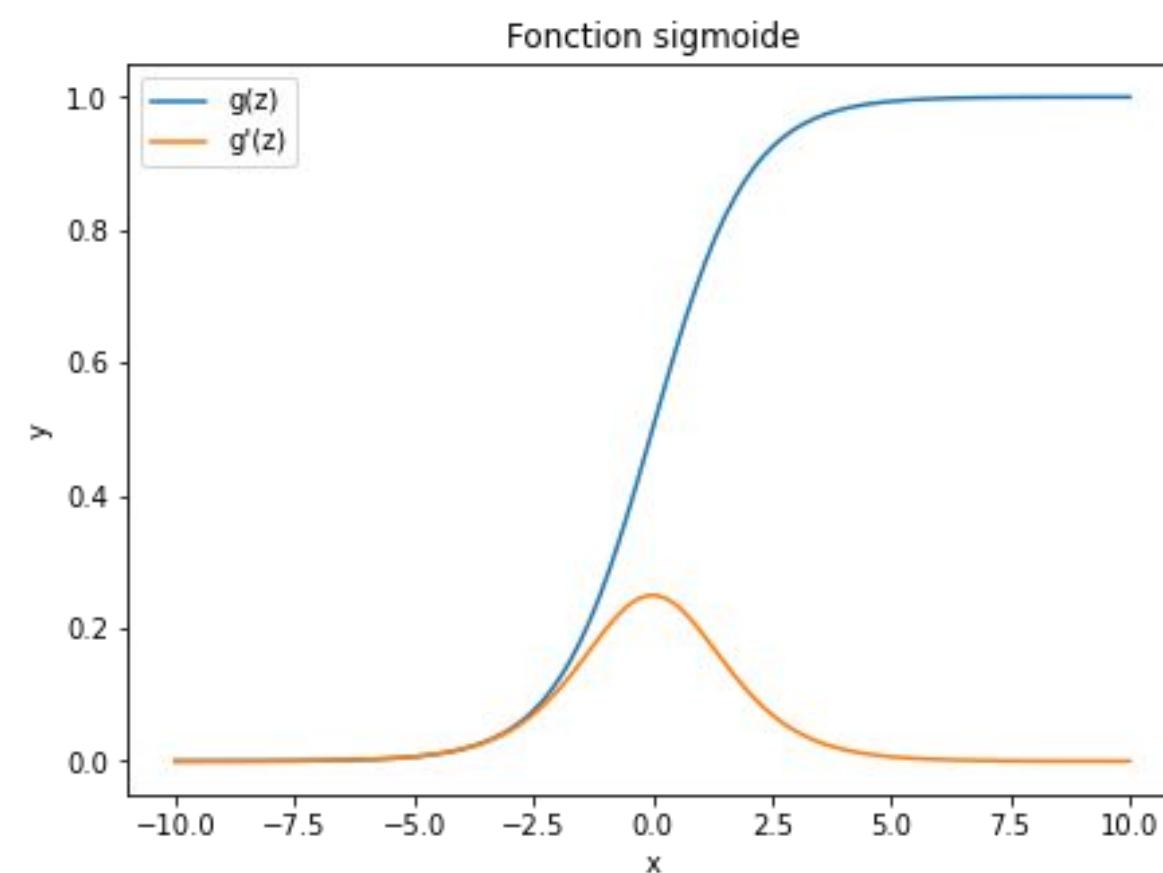
Importance des fonctions d'activation

- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.



Importance des fonctions d'activation

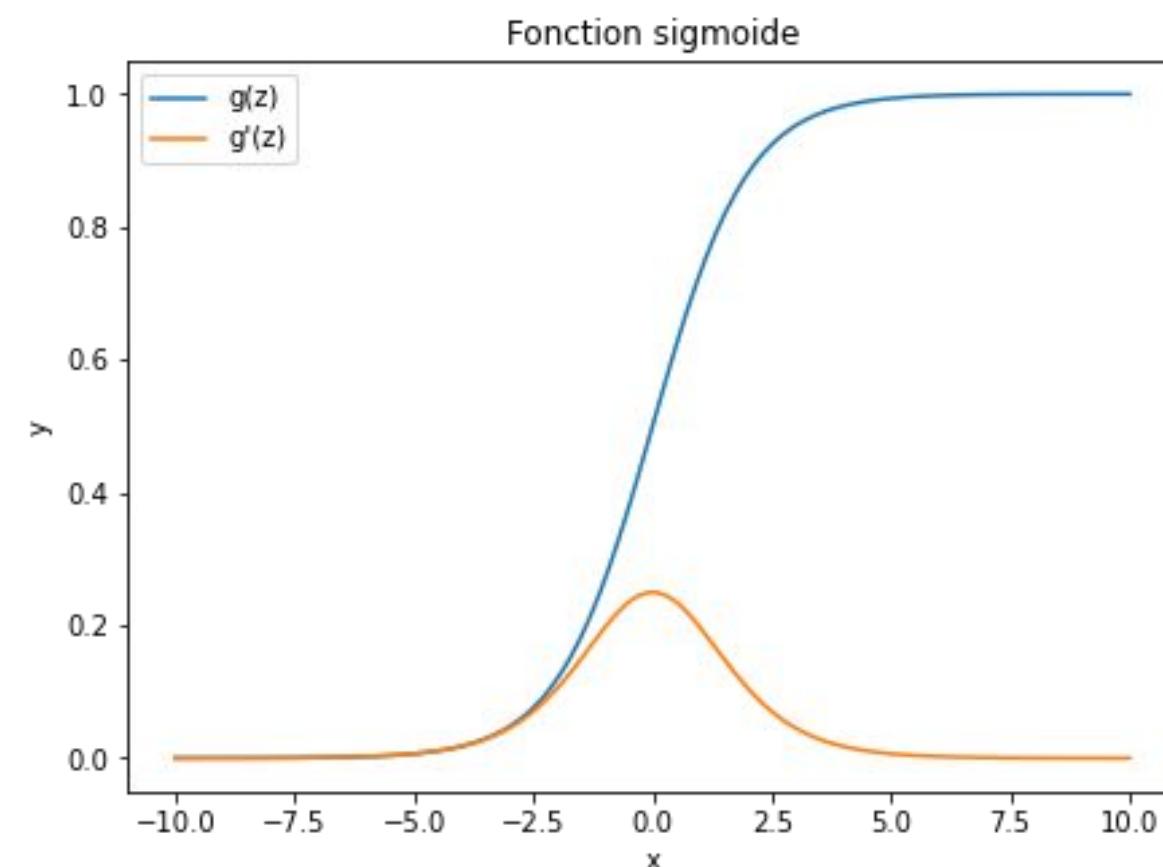
- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.



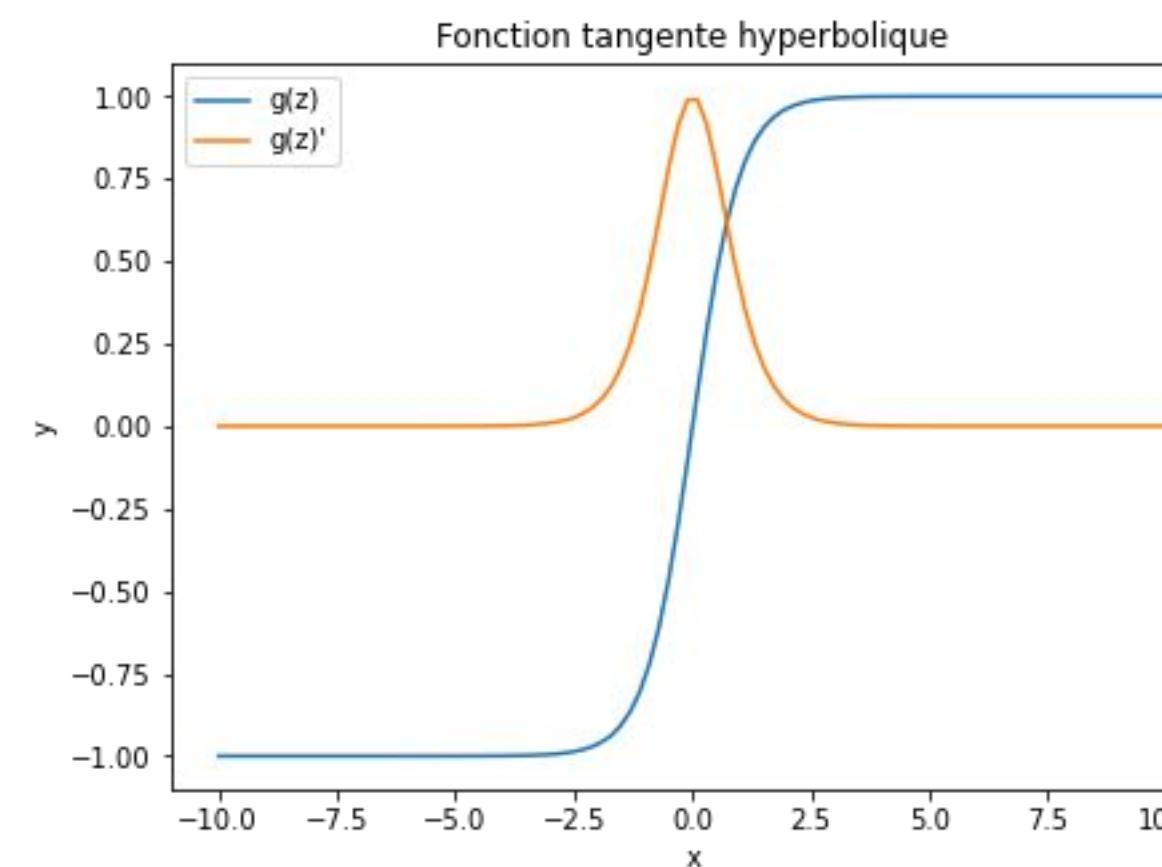
```
torch.nn.functional.sigmoid(z)
```

Importance des fonctions d'activation

- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.



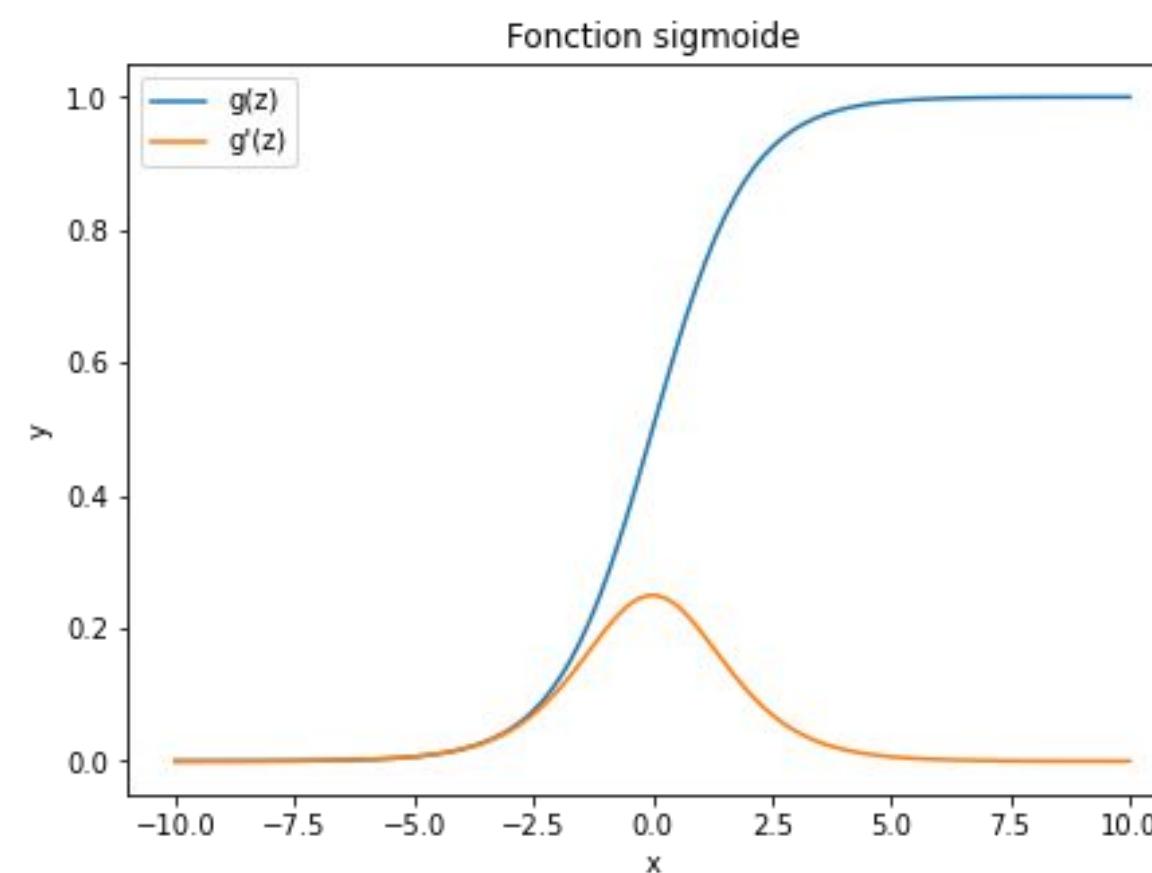
```
torch.nn.functional.sigmoid(z)
```



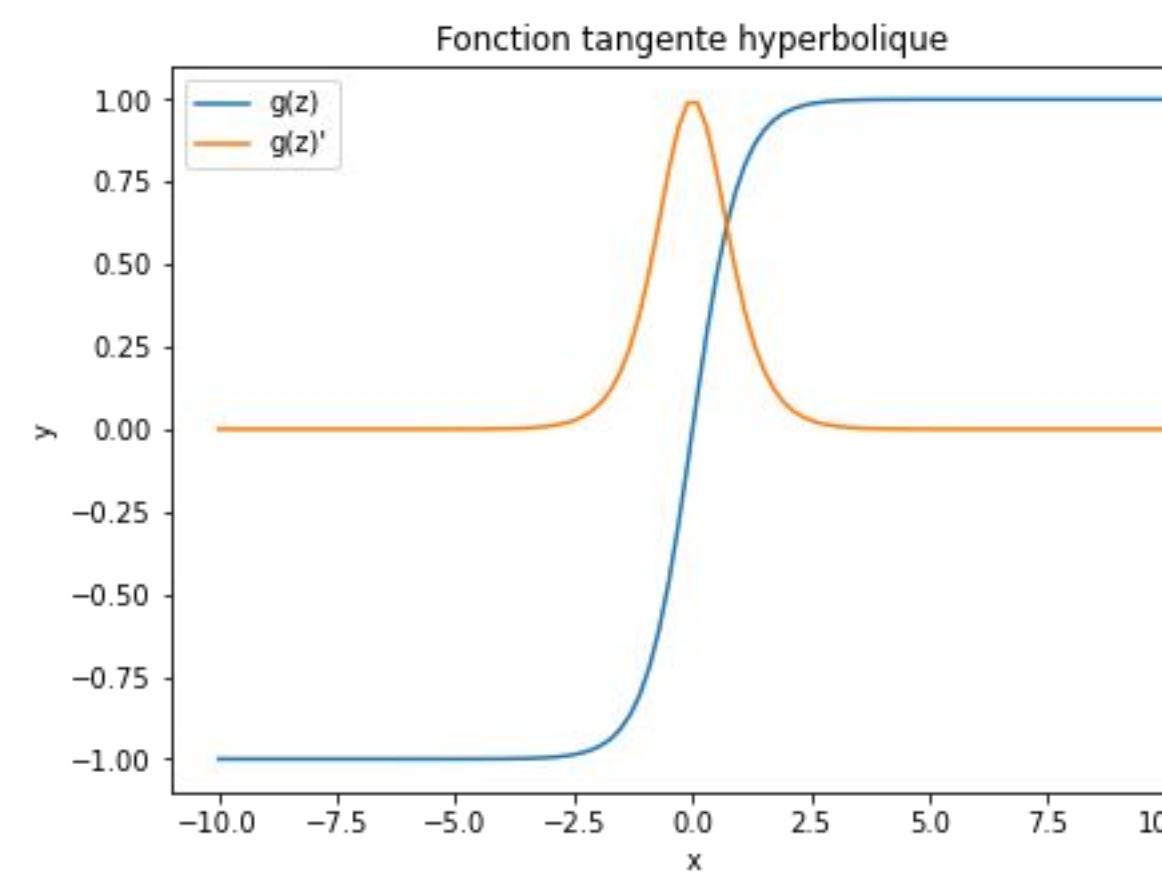
```
torch.nn.functional.tanh(z)
```

Importance des fonctions d'activation

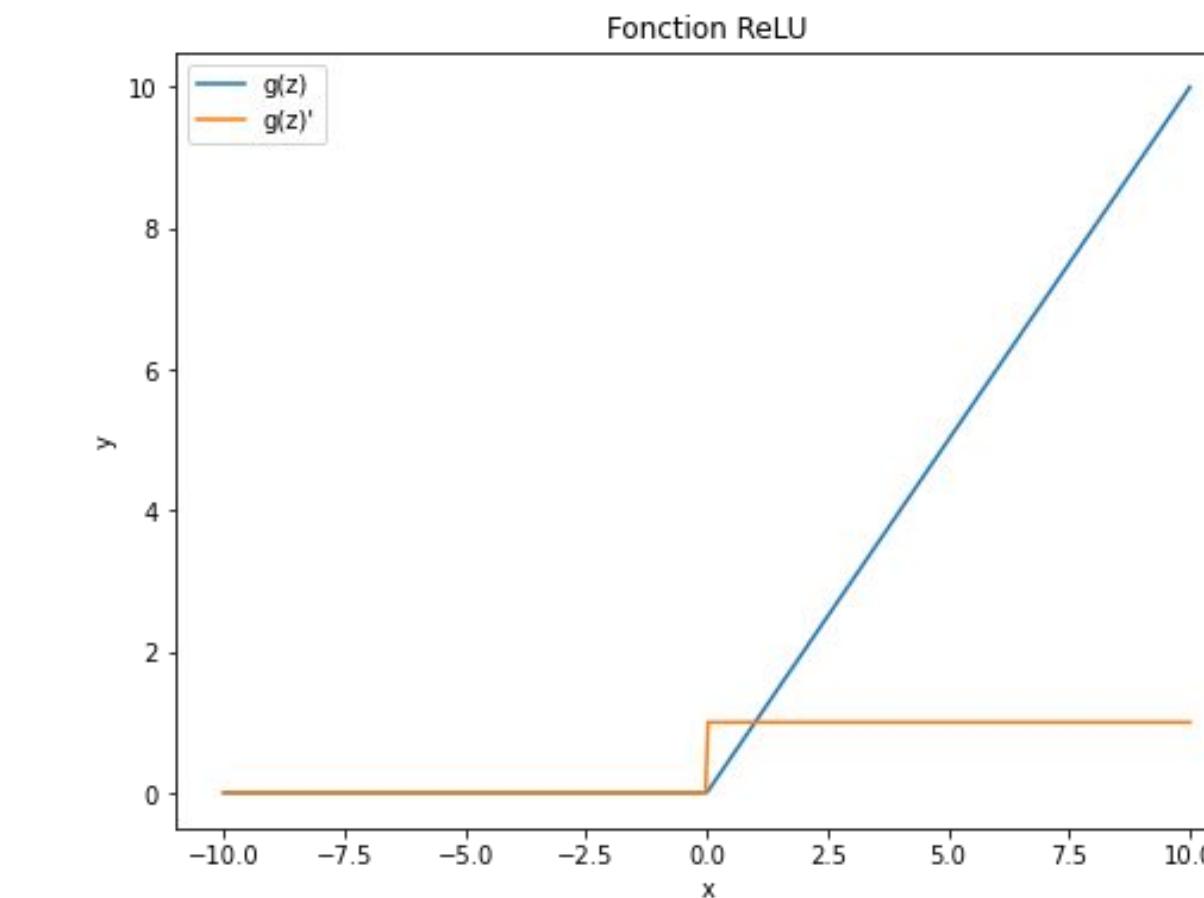
- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.



```
torch.nn.functional.sigmoid(z)
```



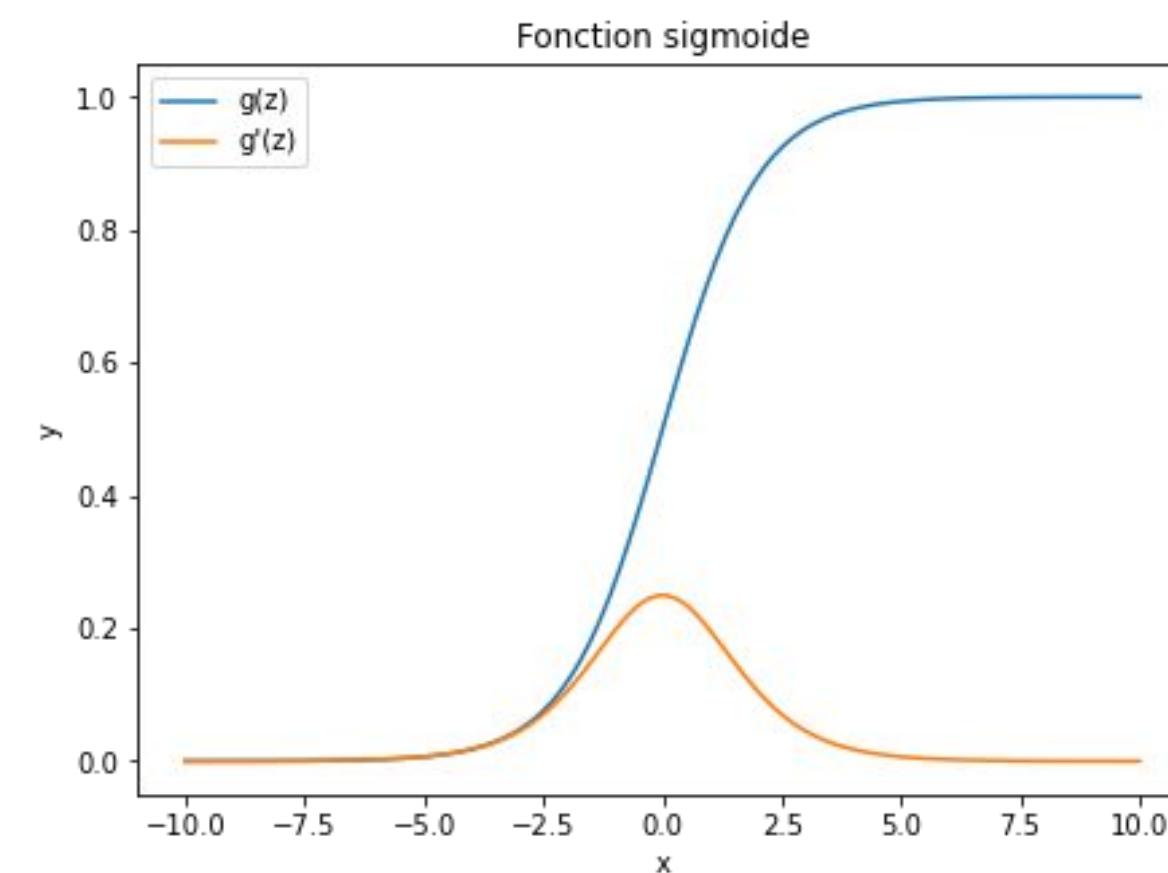
```
torch.nn.functional.tanh(z)
```



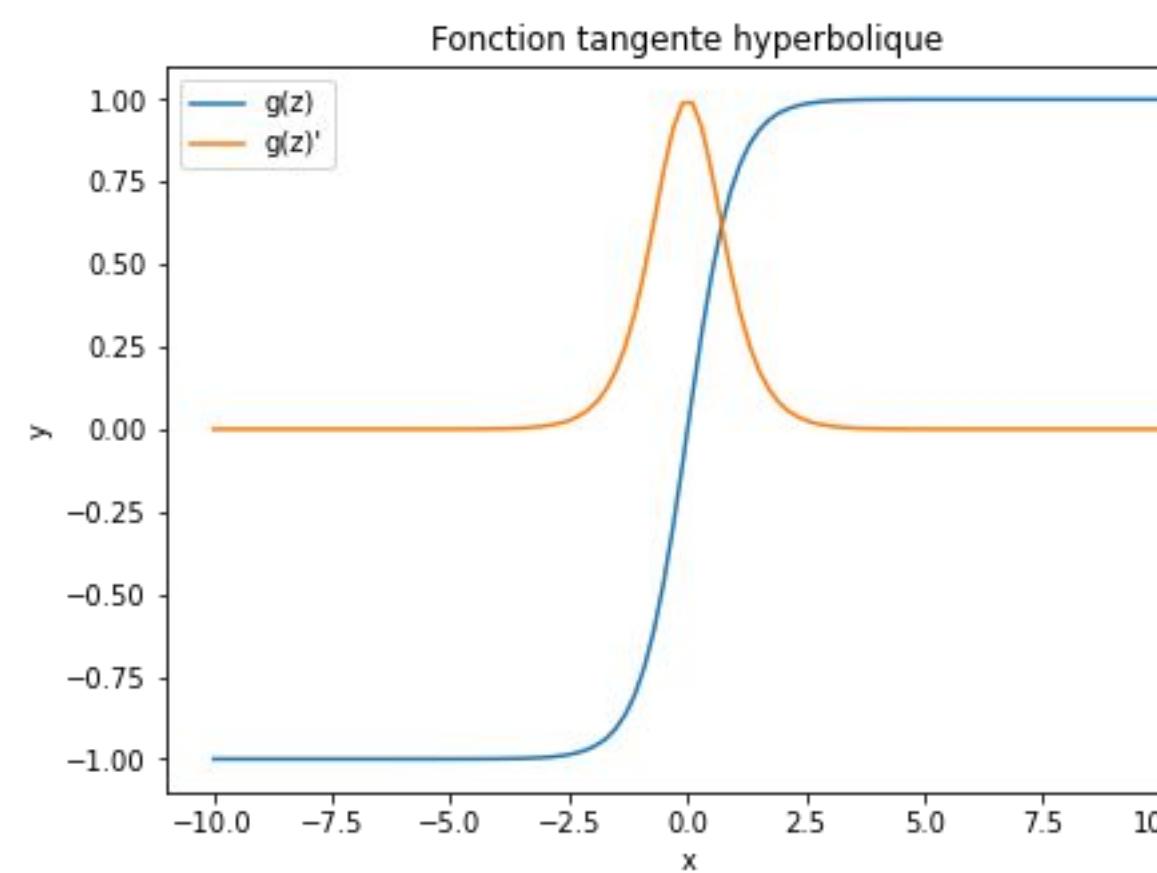
```
torch.nn.functional.relu(z)
```

Importance des fonctions d'activation

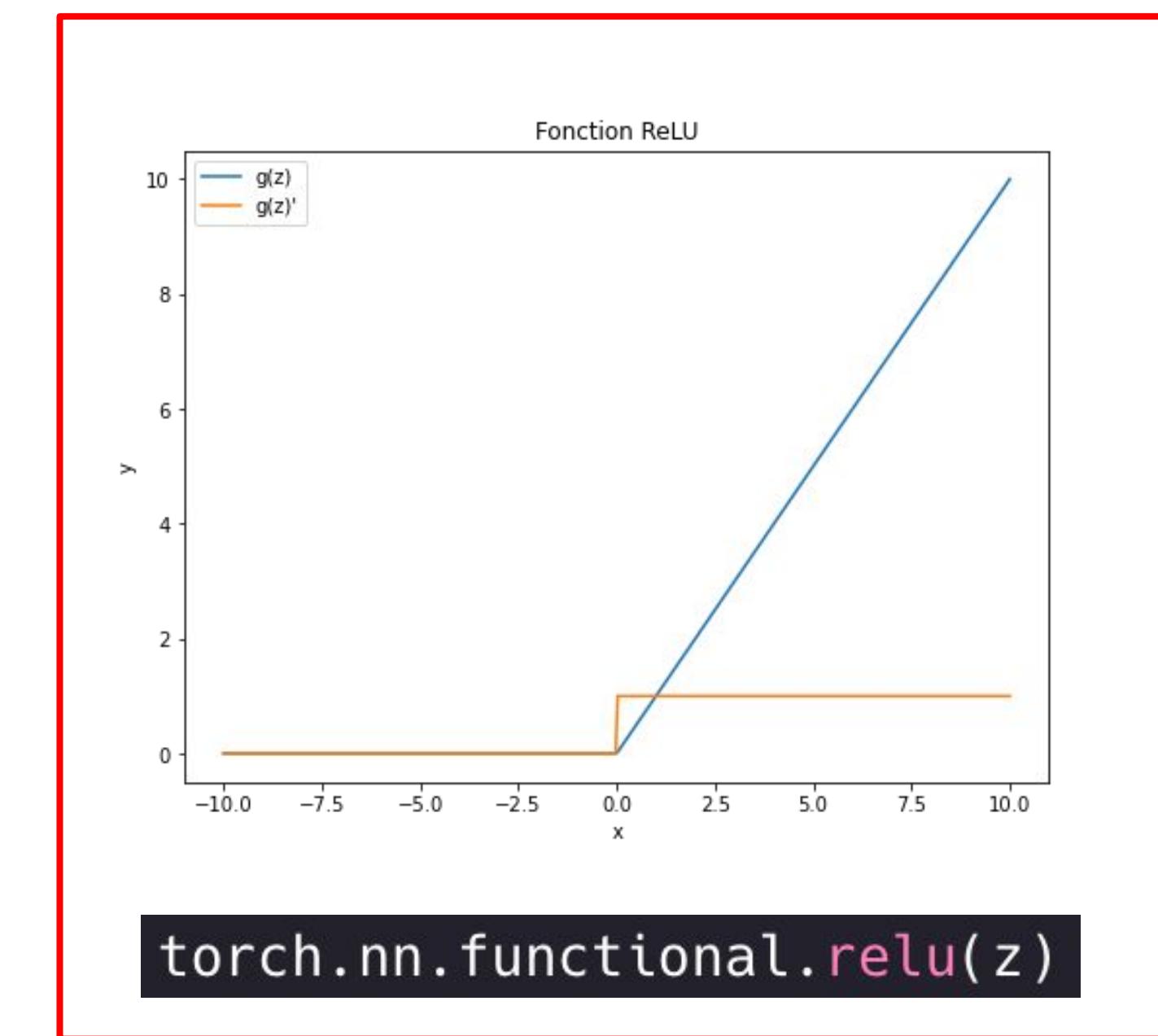
- Introduisent de la non-linéarité, permettant au réseau de neurones d'identifier des motifs complexes au sein des données.
- Sans fonction d'activation, le réseau de neurones serait équivalent à un modèle linéaire.
- De nombreux problèmes du monde réel impliquent des modèles et des relations complexes qui ne peuvent être capturés que par des fonctions non linéaires.



```
torch.nn.functional.sigmoid(z)
```



```
torch.nn.functional.tanh(z)
```



```
torch.nn.functional.relu(z)
```

Importance des fonctions d'activation

	Avantages	Inconvénients
	<ul style="list-style-type: none"> - Valeurs comprises entre 0 et 1, ce qui la rend adapté dans le cas de l'utilisation d'un neurone de sortie pour des tâches de classification binaire. - La sigmoïde est une fonction lisse et différentiable. Bonne propriété pour le processus d'optimisation. 	<ul style="list-style-type: none"> - Calcul coûteux en raison de l'exponentiel. - Problème de saturation quand $g(z) = 0$ ou 1 car $g'(z) = 0$. Le gradient de la fonction de coût disparaît. C'est ce qu'on appelle le « Vanishing gradient ». - Fonction non centrée en zéro. Le gradient de tous les poids connectés au même neurone sera soit positif ou soit négatif.
	<ul style="list-style-type: none"> - Fonction centrée en zéros. 	<ul style="list-style-type: none"> - Calcul coûteux en raison des exponentiels. - Problème de vanishing gradient. Quand la fonction de rapproche de -1 ou 1, la dérivée se rapproche de 0.
	<ul style="list-style-type: none"> - Calcul et optimisation plus rapide - Ne souffre pas du problème de vanishing gradient car non bornée du côté positive. 	<ul style="list-style-type: none"> - Problème de « ReLU mourants » car les valeurs négatives restent toujours à zéro, ce qui entraîne un gradient nul pour les neurones concernés. - La fonction n'est pas différentiable en zéros.

Perceptron avec PyTorch



```
import torch
import torch.nn as nn
import torch.nn.functional as F

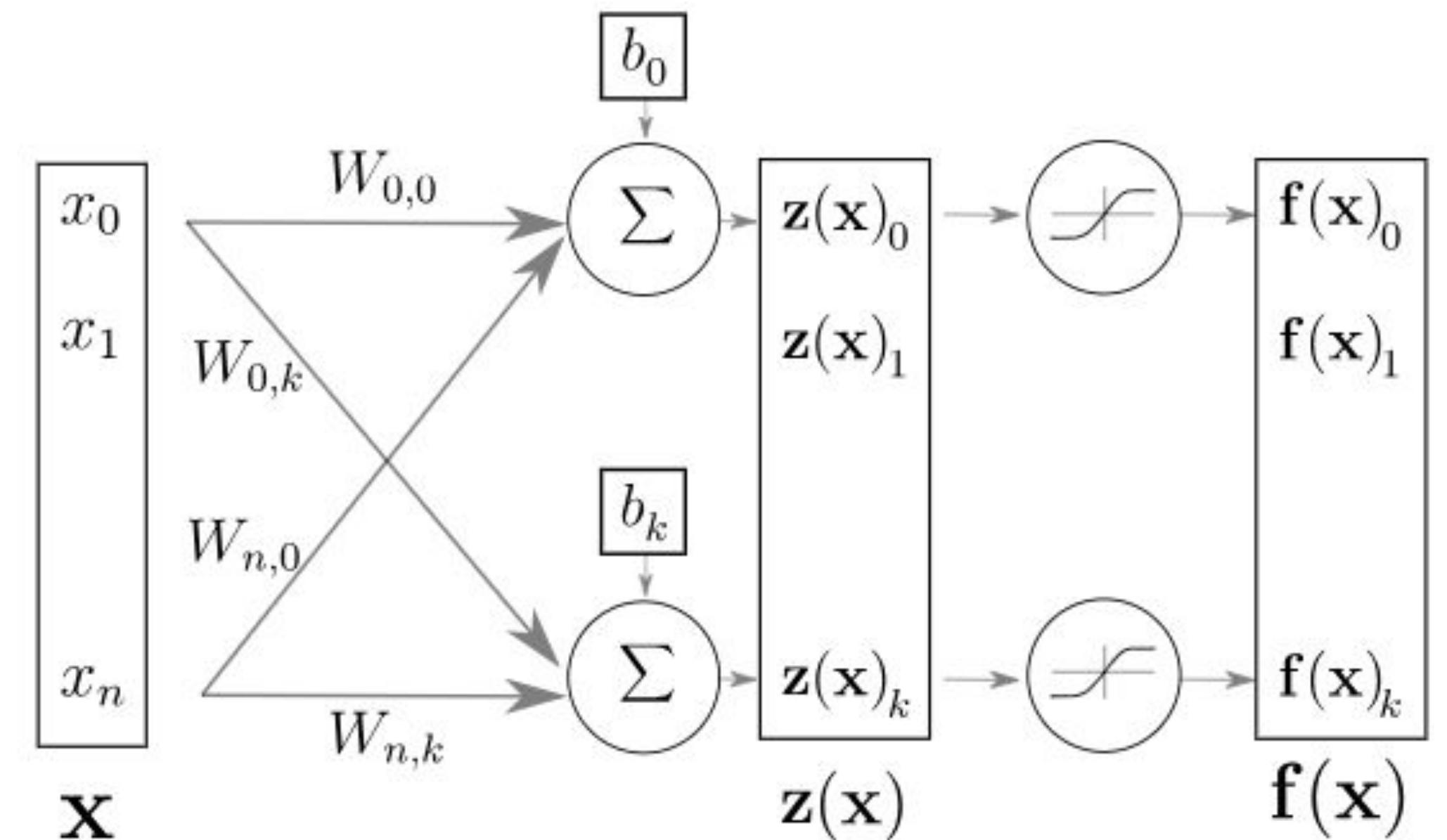
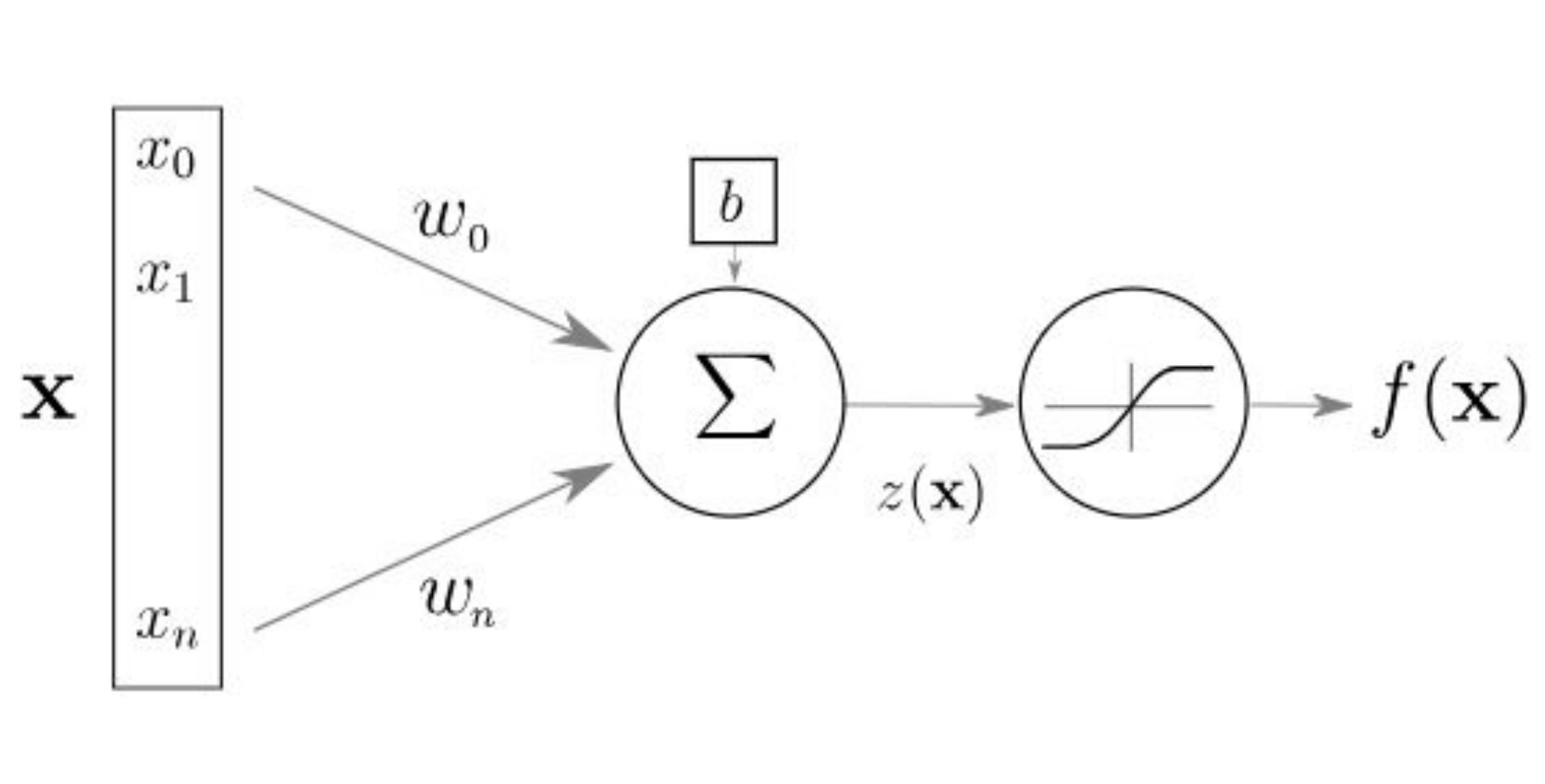
class Perceptron(nn.Module):
    def __init__(self, input_dim):
        super(Perceptron, self).__init__()

        # Définition d'une couche entièrement connectée
        # Cette couche est composé de input_dim neurones
        # en entrée et de un neurone en sortie
        self.fc = nn.Linear(input_dim, 1)

    def forward(self, x): # x correspond aux données en entrée
        # Propagation forward à travers la couche fc
        x = self.fc(x)
        # Application de la fonction d'activation ReLU
        # à la sortie de fc
        x = F.relu(x)
        return x
```

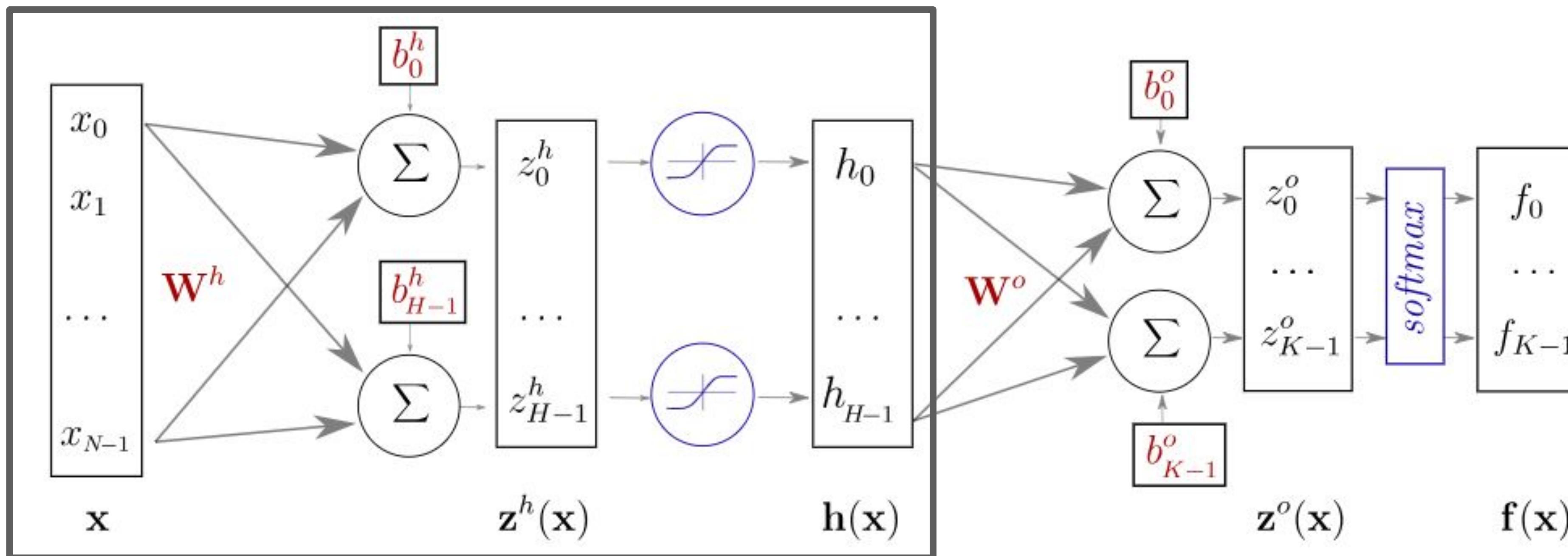
Comment ça marche ?

Réseau de neurones à 1 couche cachée



Comment ça marche ?

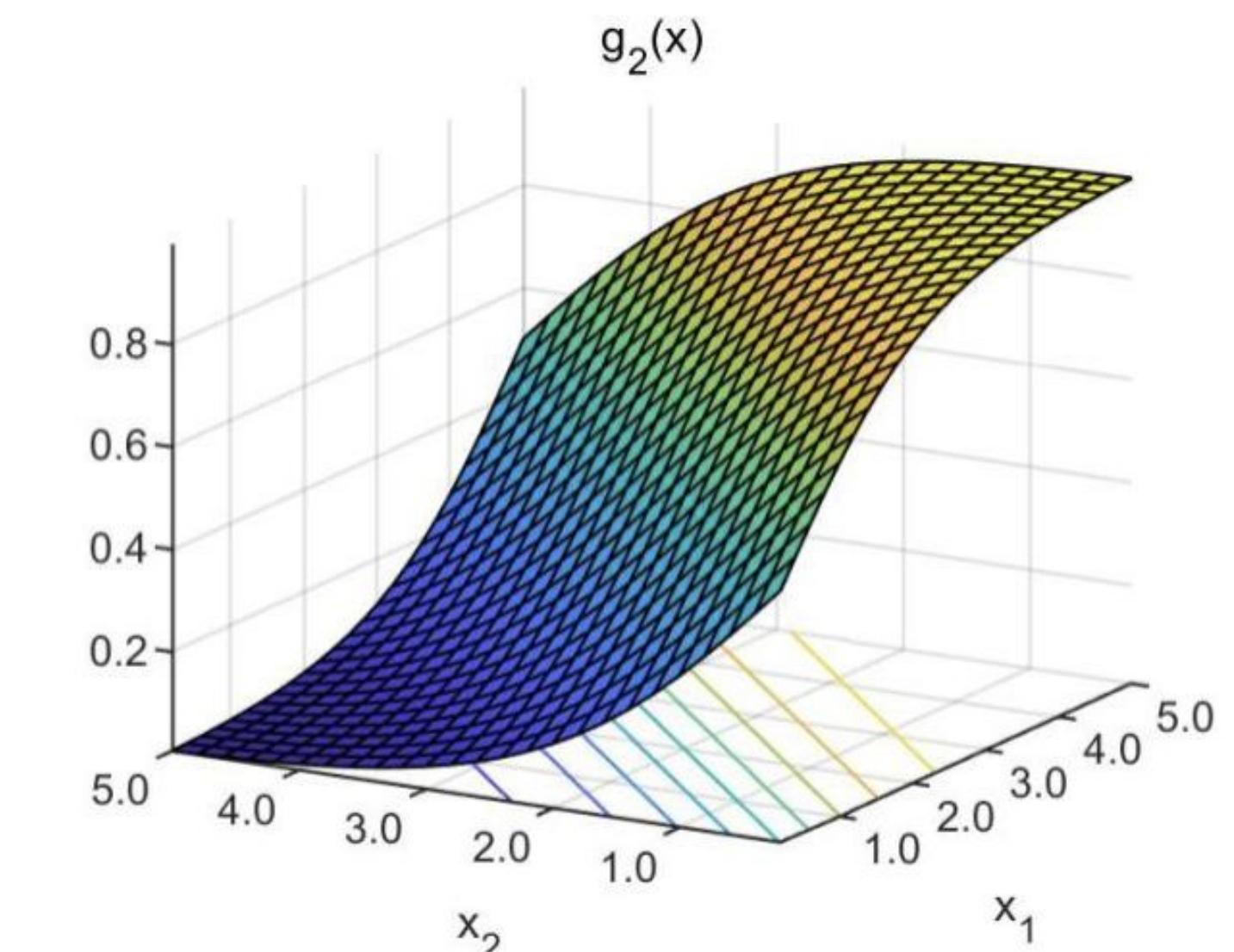
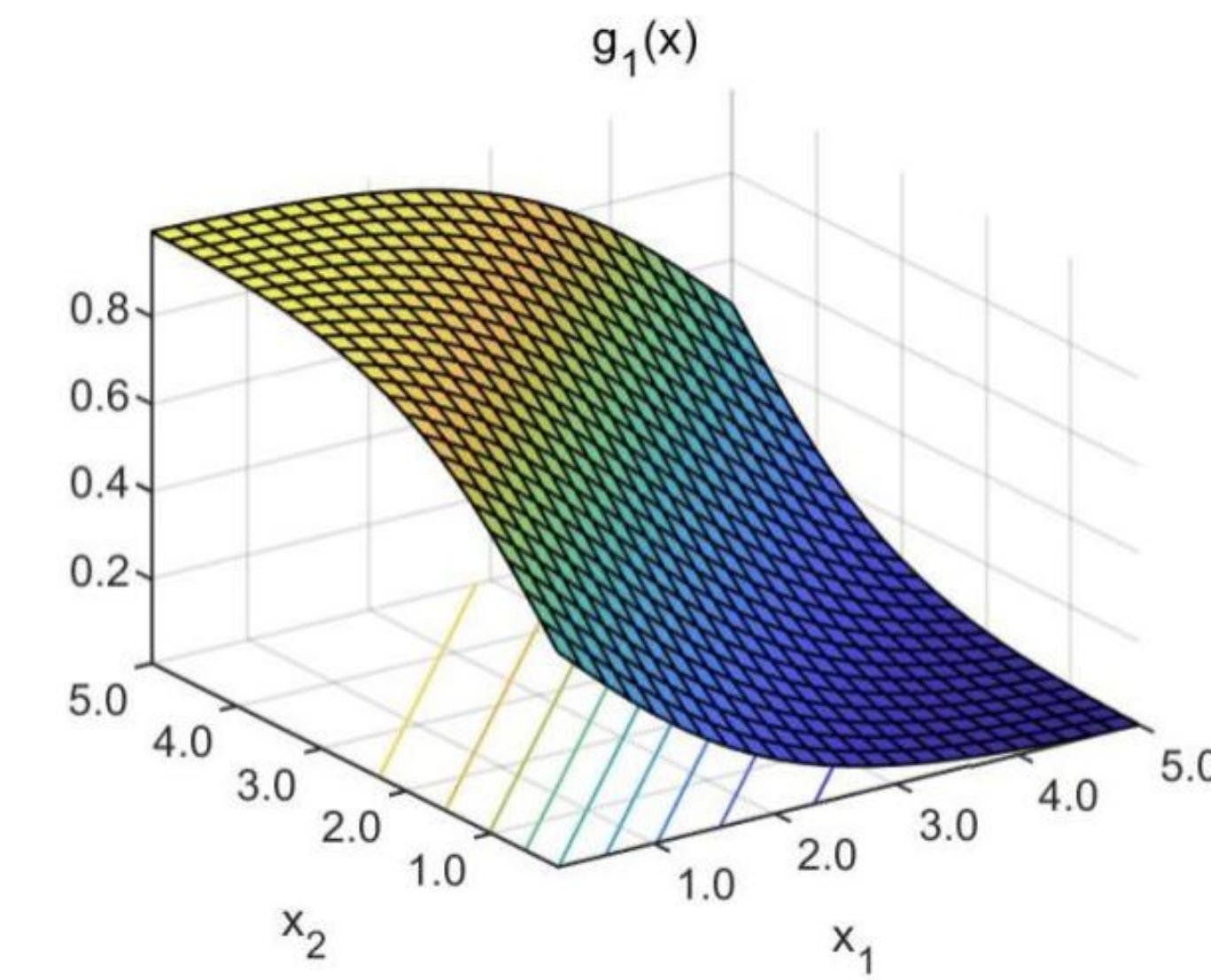
Réseau de neurones à 1 couche cachée



Comment ça marche ?

Fonction SOFTMAX

$$g_j(x) = \frac{e^{x_j}}{\sum_{i=1}^n e^{x_i}}, \quad j = 1 : n$$

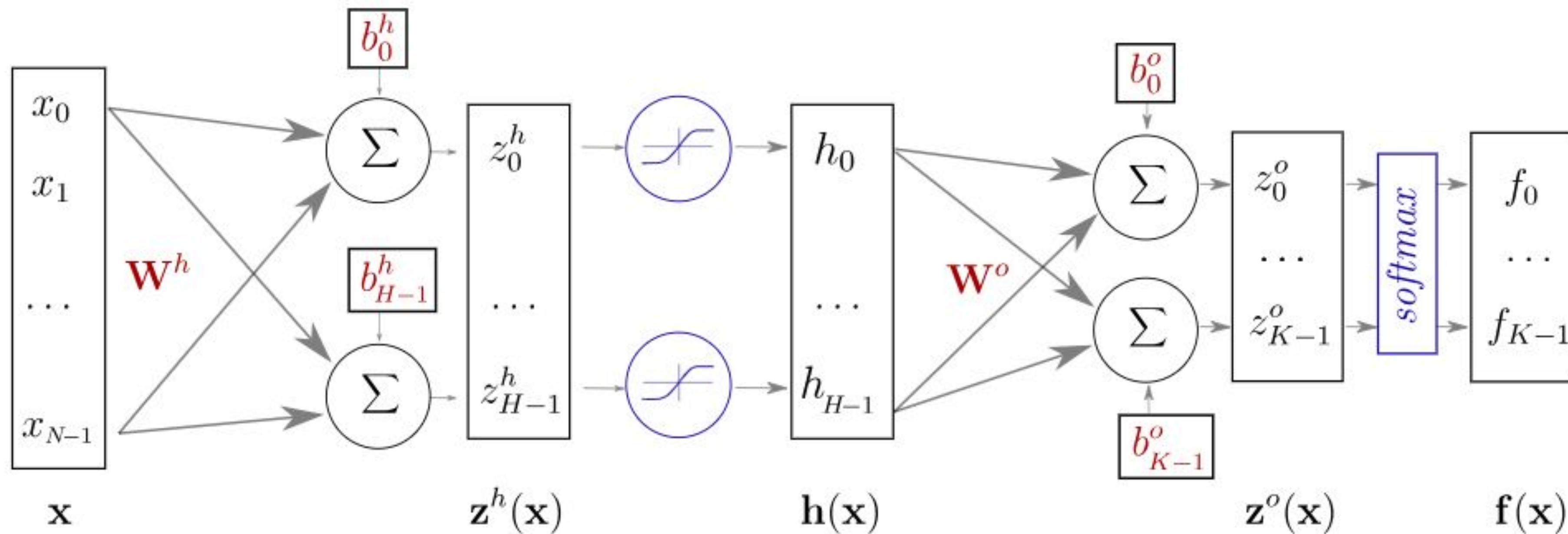


→ Adapté à la classification multi-classe

→ Transforme la sortie du réseau en distribution de probabilité

Comment ça marche ?

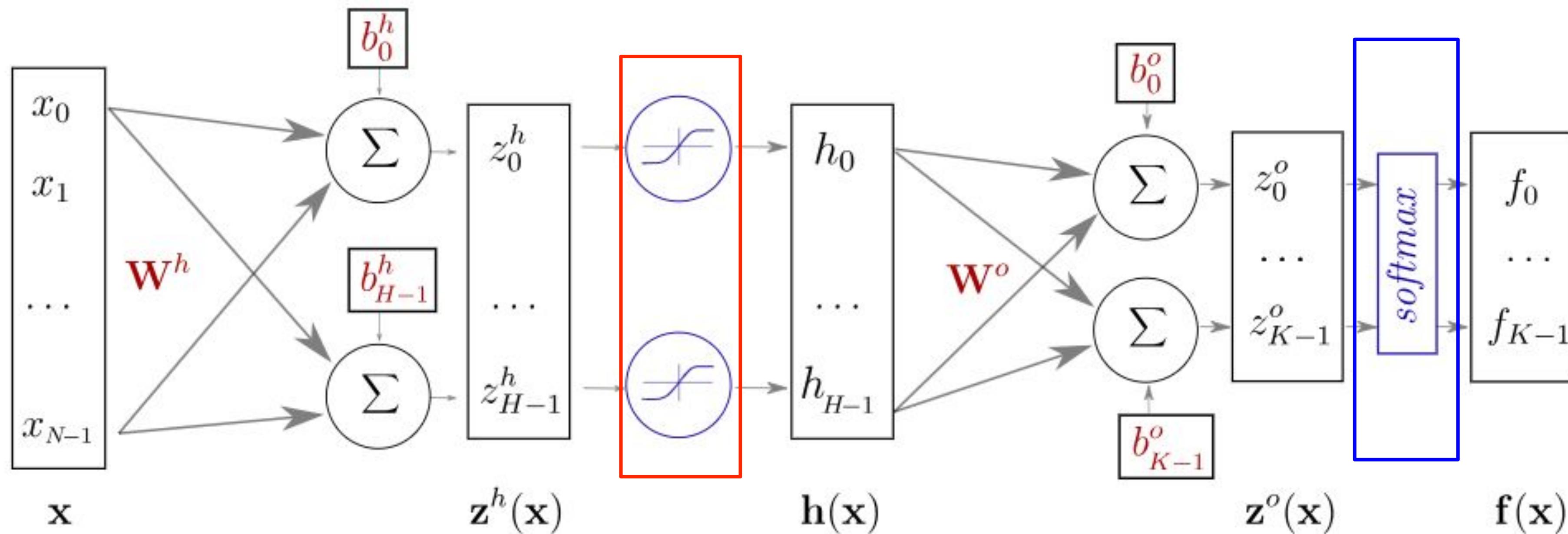
Réseau de neurones à 1 couche cachée



$$h(x) = g(W^h x + b^h)$$

Comment ça marche ?

Réseau de neurones à 1 couche cachée

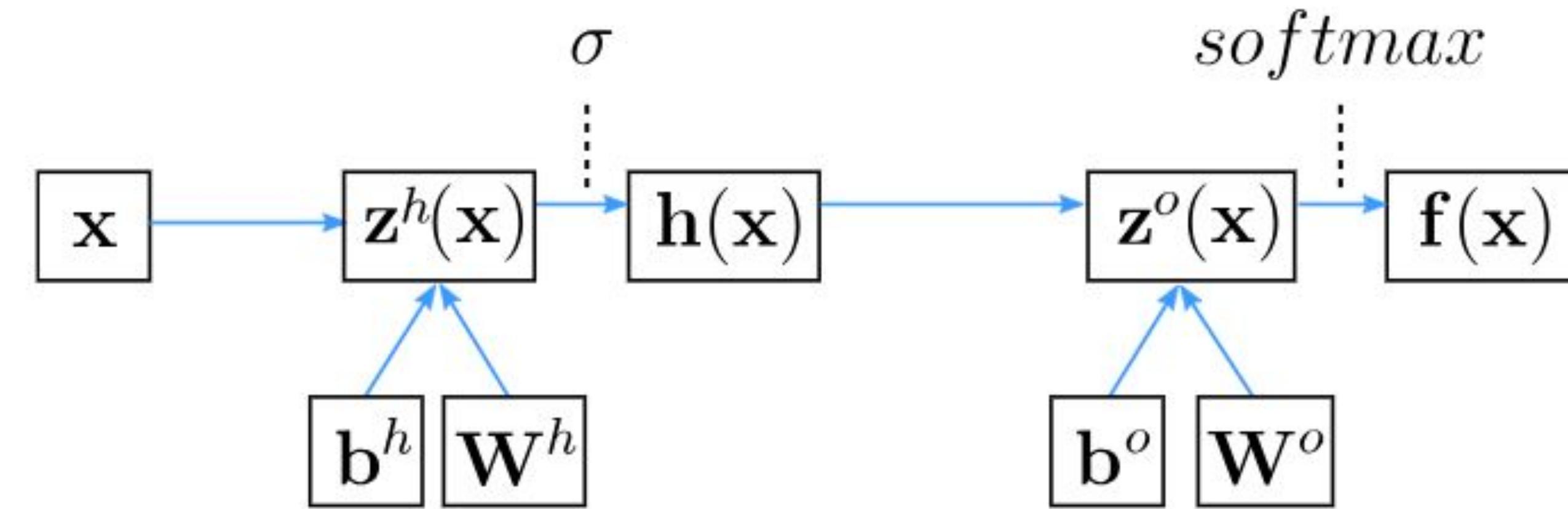


$$h(x) = g(W^h x + b^h)$$

$$f(x) = \text{softmax}(W^o h(x) + b^o)$$

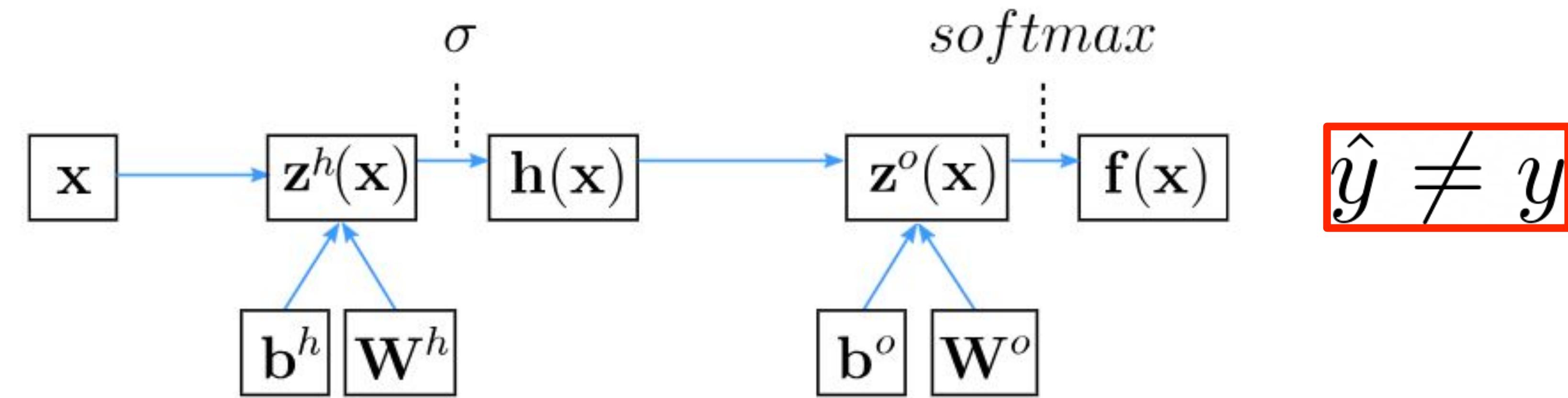
Comment ça marche ?

Réseau de neurones à 1 couche cachée



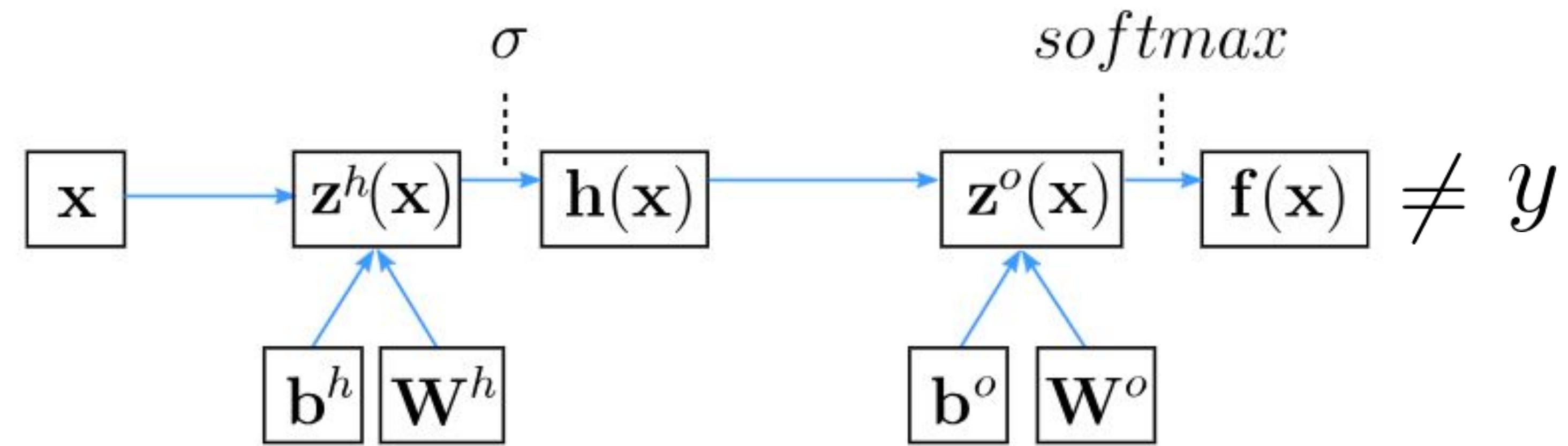
Comment ça marche ?

L'apprentissage: un problème d'optimisation



Comment ça marche ?

L'apprentissage: un problème d'optimisation



Objectif: Trouver les meilleurs paramètres $\theta = (W^h, b^h, W^o, b^o)$

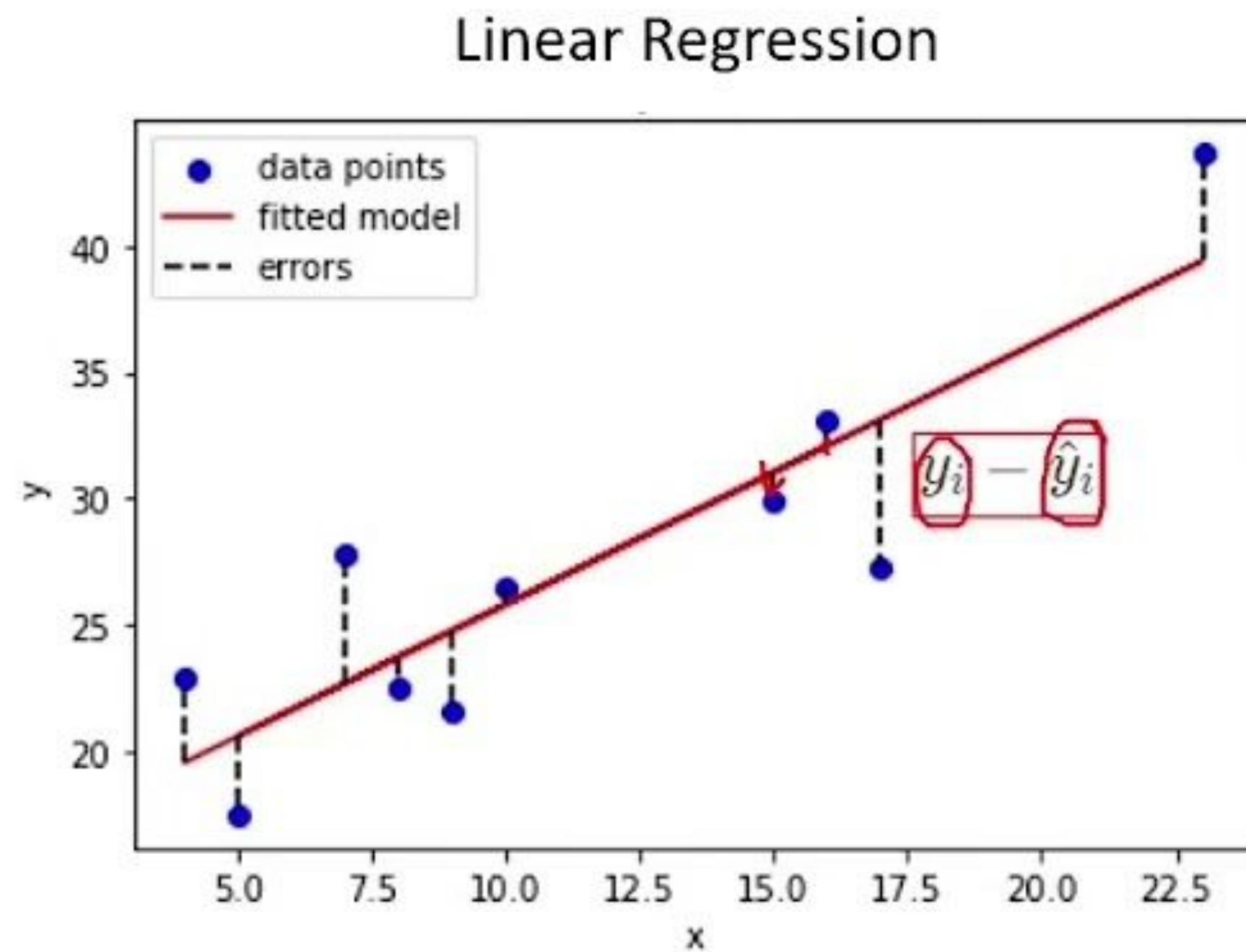
pour minimiser une **fonction coût** l

Fonctions de coût classiques

Fonctions de coût classiques

Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \rightarrow \text{Régression}$$



Fonctions de coût classiques

Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \rightarrow \text{Régression}$$

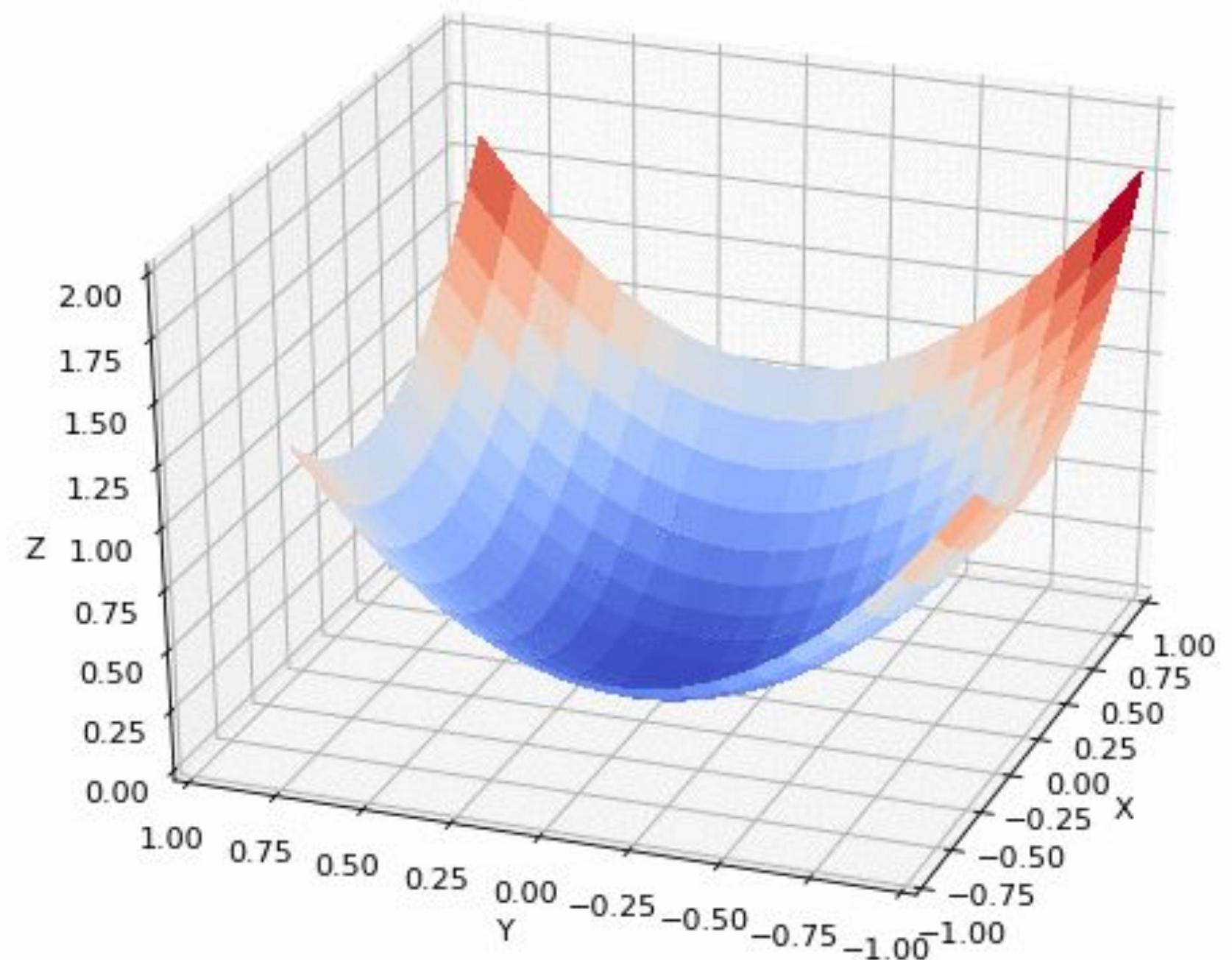
Categorical Cross-Entropy (CCE)

$$CCE = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^M y_{ij} \log(\hat{y}_{ij}) \rightarrow \text{Classification}$$

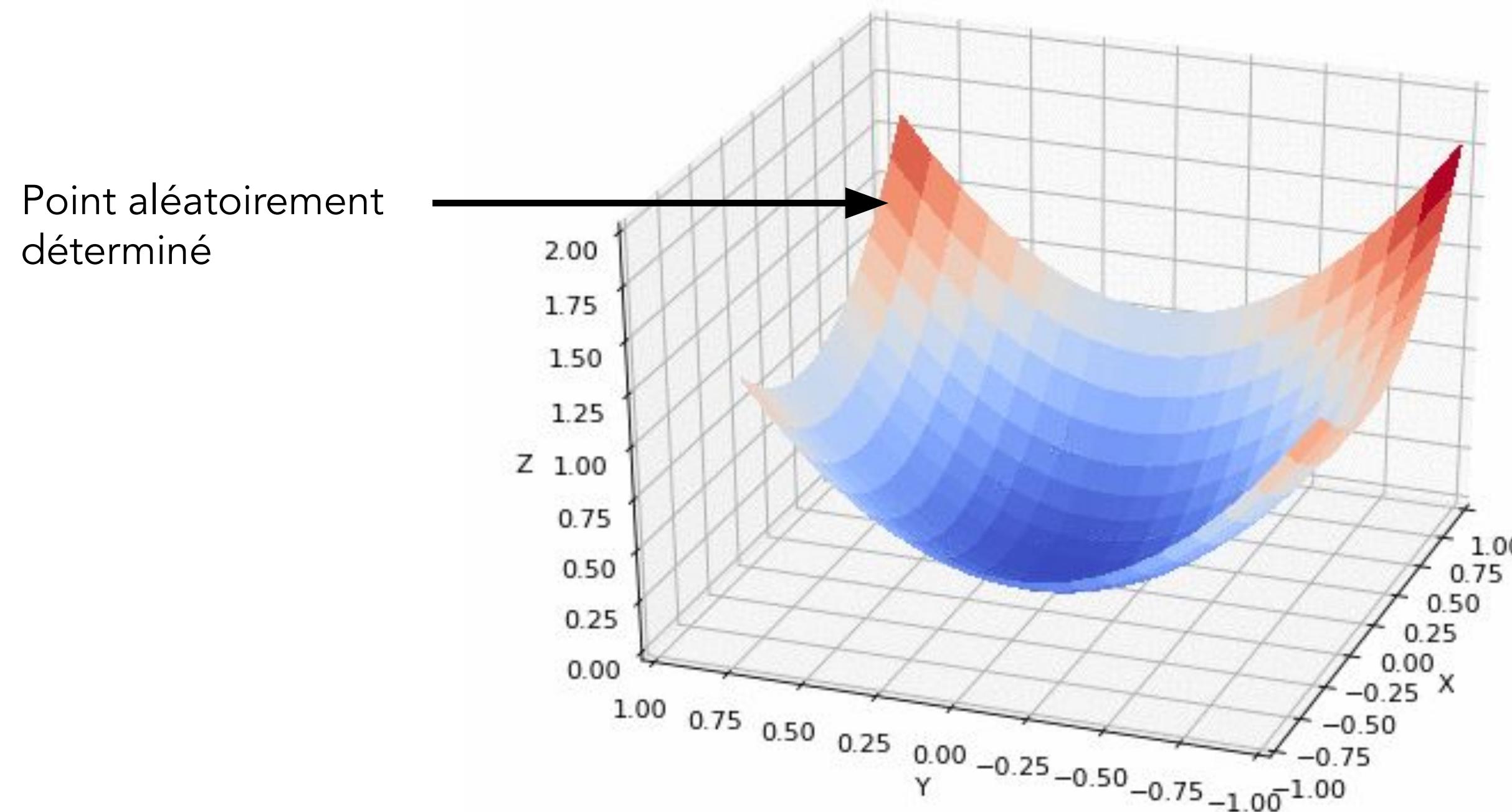
Binary Cross-Entropy (BCE)

$$BCE = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Descente de Gradient



Descente de Gradient



Point aléatoirement
déterminé

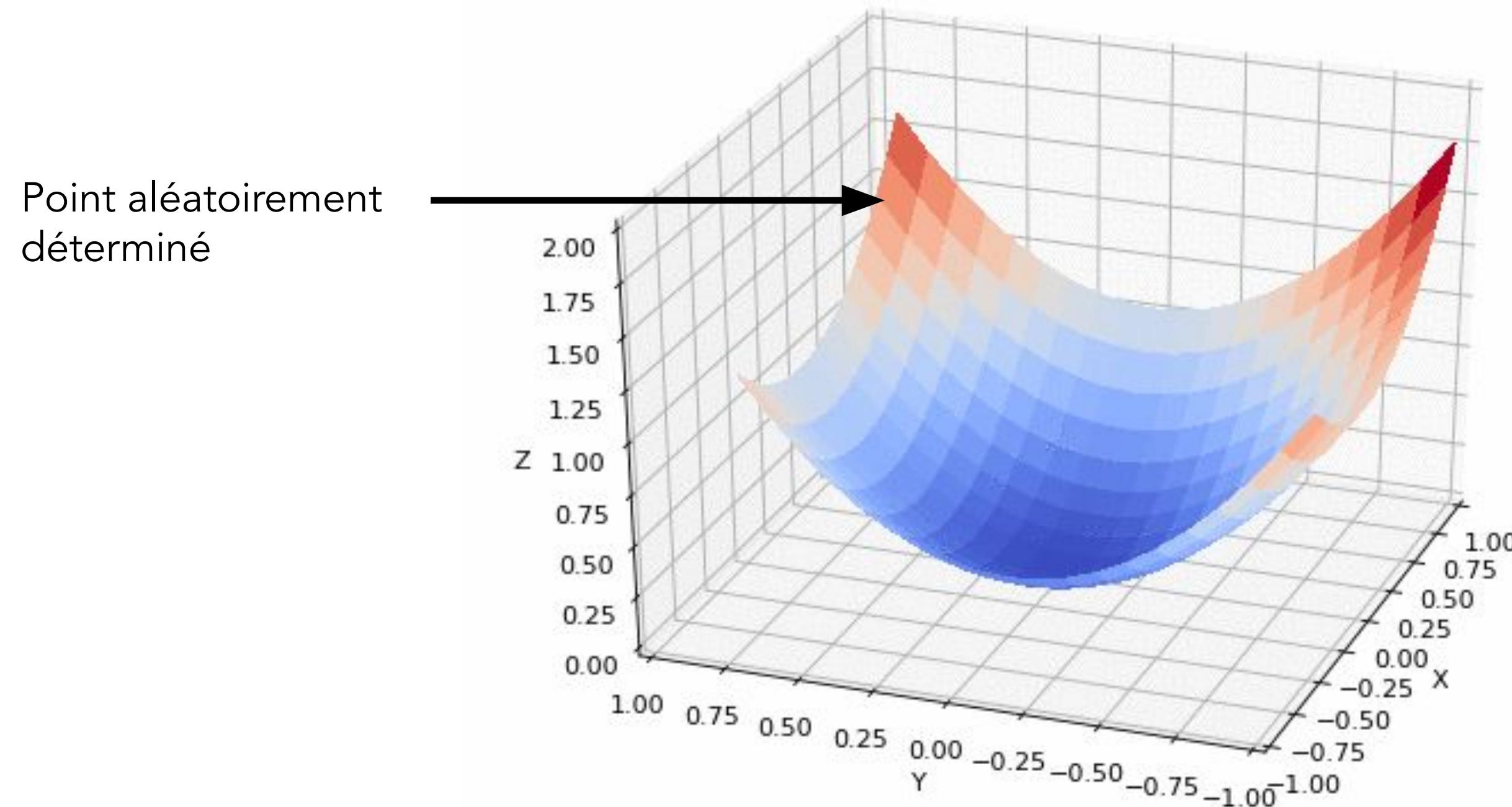
Initialisation aléatoire des paramètres

Calcul du gradient

Faire un petit pas dans la direction opposée au gradient

Continuer jusqu'à convergence

Descente de Gradient



Point aléatoirement
déterminé

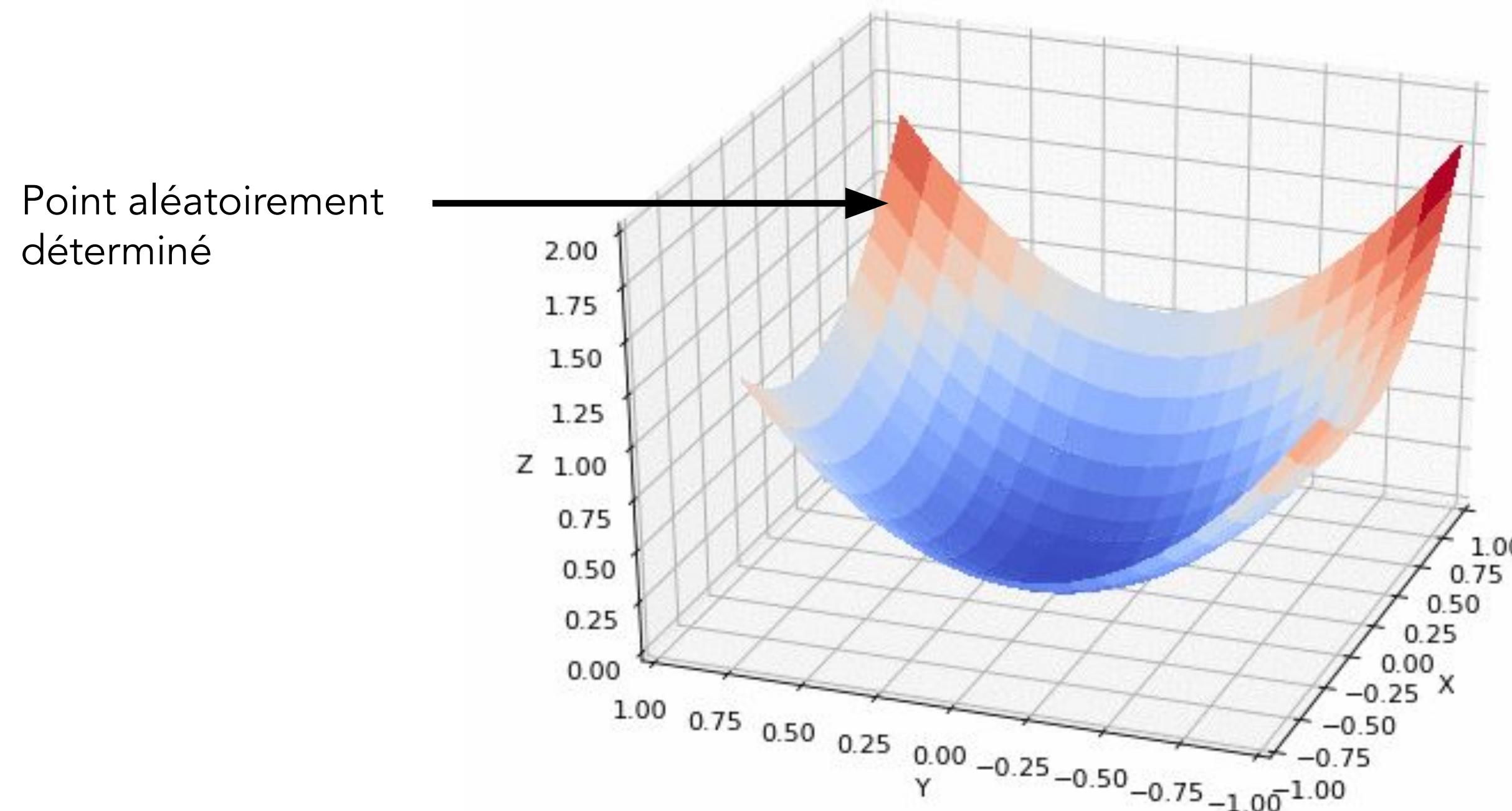
Initialisation aléatoire des paramètres

Calcul du gradient de la fonction coût $\delta = \nabla_{\theta} l(f(x), y)$

Faire un petit pas dans la direction opposée au
gradient $\theta \leftarrow \theta - \eta \delta$

Continuer jusqu'à convergence

Descente de Gradient



Point aléatoirement
déterminé

Initialisation aléatoire des paramètres

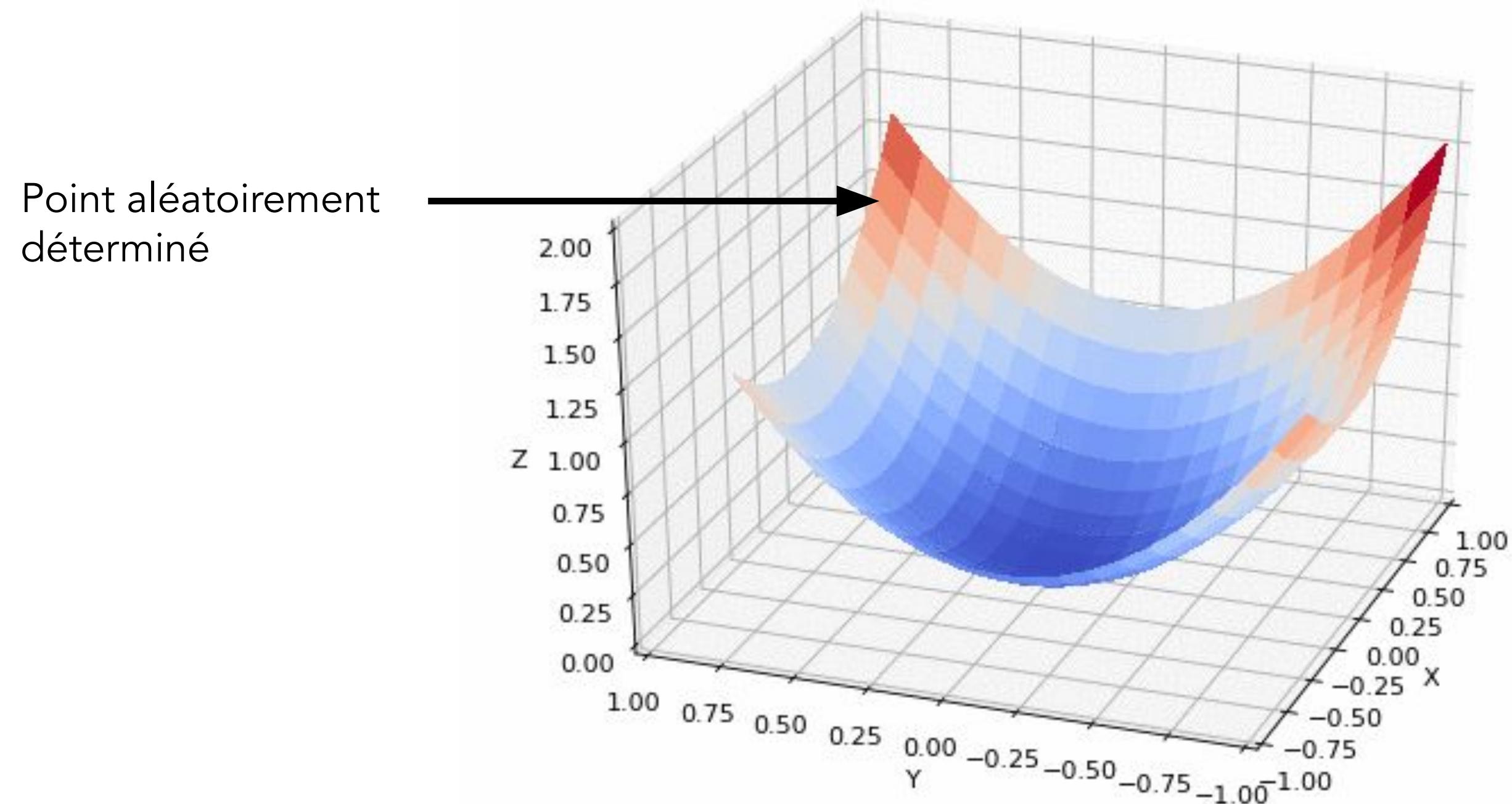
Calcul du gradient de la fonction coût $\delta = \nabla_{\theta} l(f(x), y)$

**Faire un petit pas dans la direction opposée au
gradient** $\theta \leftarrow \theta - \eta \delta$

Continuer jusqu'à convergence

* Learning rate

Descente de Gradient



Point aléatoirement
déterminé

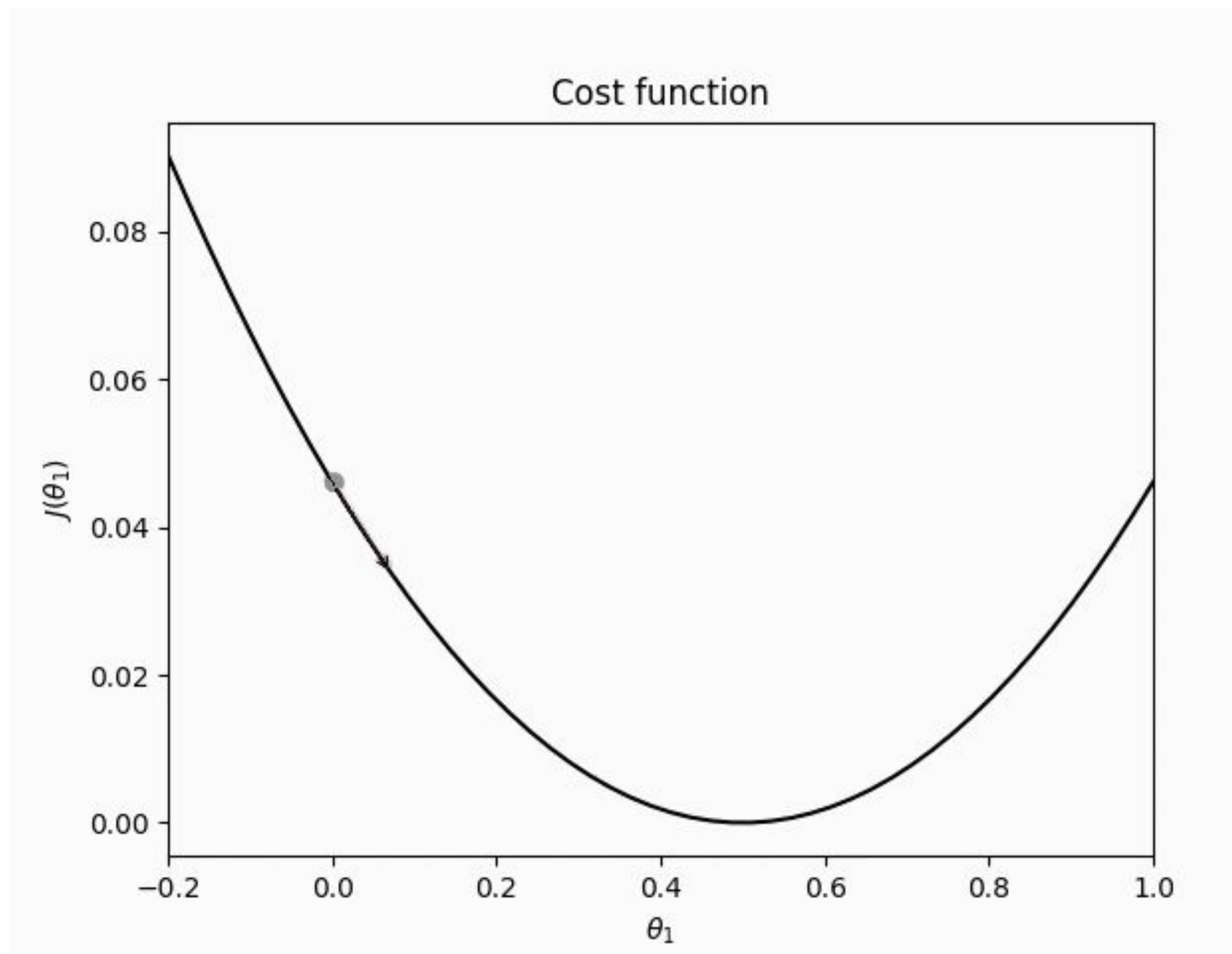
Initialisation aléatoire des paramètres

Calcul du gradient de la fonction coût $\delta = \nabla_{\theta} l(f(x), y)$

Faire un petit pas dans la direction opposée au
gradient $\theta \leftarrow \theta - \eta \delta$

Continuer jusqu'à convergence

Descente de Gradient



Initialisation aléatoire des paramètres

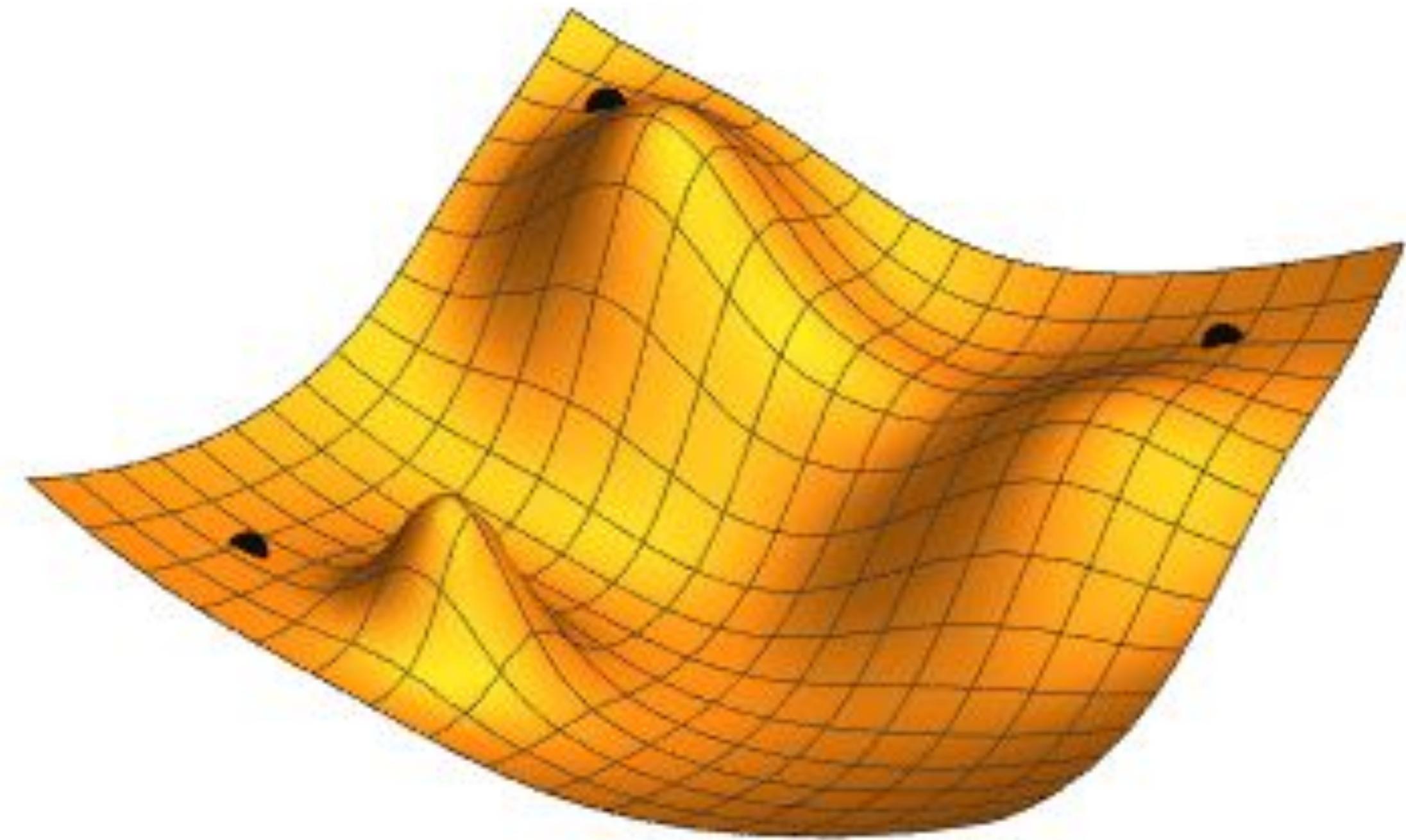
Calcul du gradient de la fonction coût $\delta = \nabla_{\theta} l(f(x), y)$

Faire un petit pas dans la direction opposée au gradient $\theta \leftarrow \theta - \eta \delta$

Continuer jusqu'à convergence

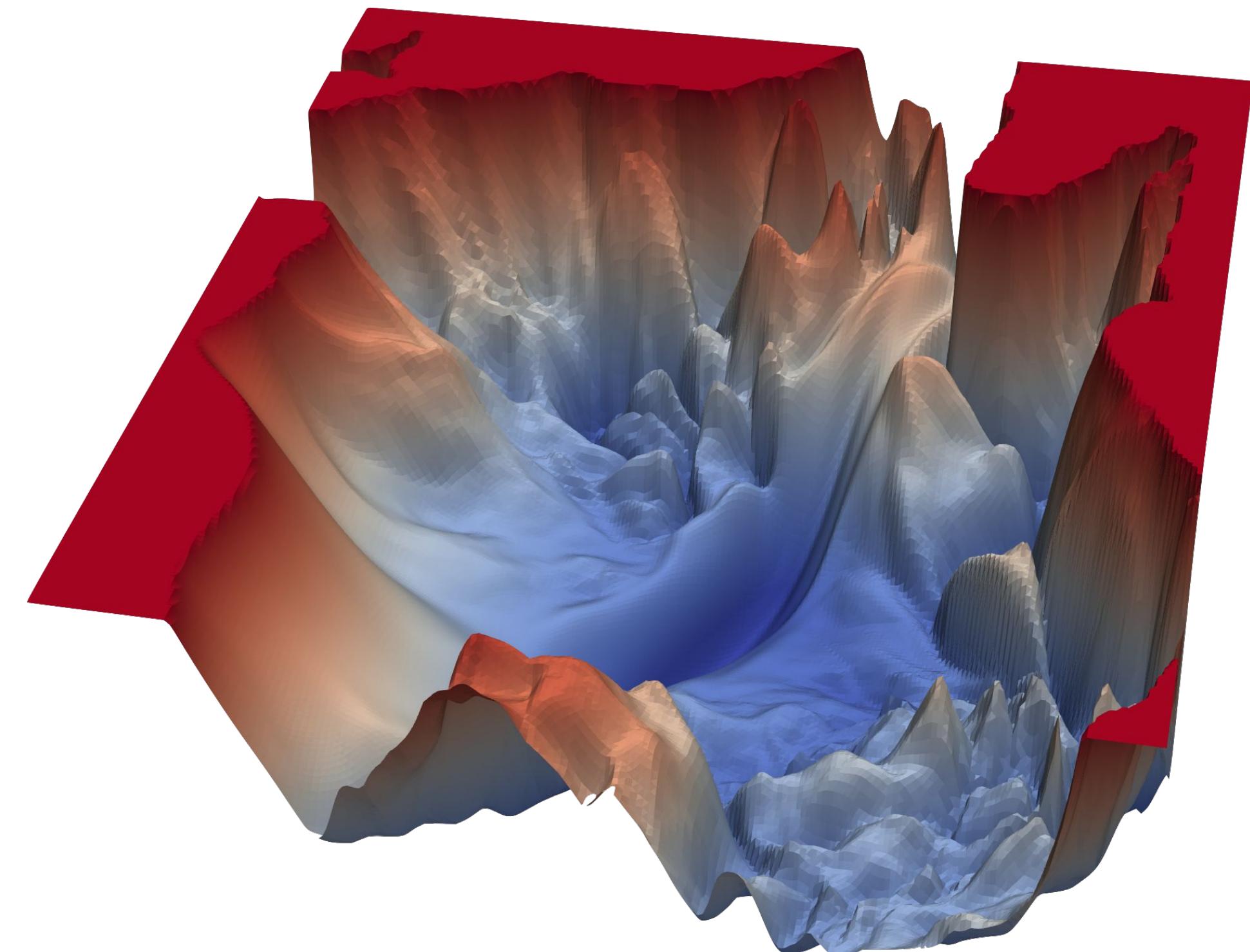
Défis en matière d'optimisation

Trouver le minimum global de la fonction de coût peut être difficile, car la descente de gradient peut se retrouver piégée dans des minima locaux.



Défis en matière d'optimisation

Exemple d'une représentation 3D des contours d'une fonction de coût relatif au réseau de neurones VGG16 sur l'ensemble de donnée Cifar-10.



Source : <https://www.cs.umd.edu/~tomg/projects/landscapes/>

Comment ça marche ?

Calculer le gradient: backpropagation

Problème: Comment calculer le gradient pour faire $\theta \leftarrow \theta - \eta \delta$?

Comment ça marche ?

Calculer le gradient: backpropagation

Problème: Comment calculer le gradient pour faire $\theta \leftarrow \theta - \eta \delta$?

- Utilisation de l'algorithme « **Backpropagation** » pour ajuster les paramètres des réseaux de neurones afin de minimiser l'erreur de prédiction.
- Cette méthode fonctionne en calculant les gradients par rapport à l'erreur à chaque couche du réseau, puis en propageant ces gradients de manière rétroactive pour mettre à jour tous les poids.

Comment ça marche ?

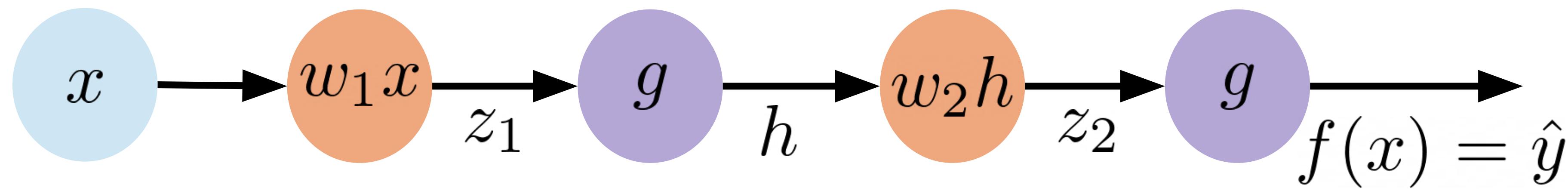
Calculer le gradient: backpropagation

Problème: Comment calculer le gradient pour faire $\theta \leftarrow \theta - \eta \delta$?

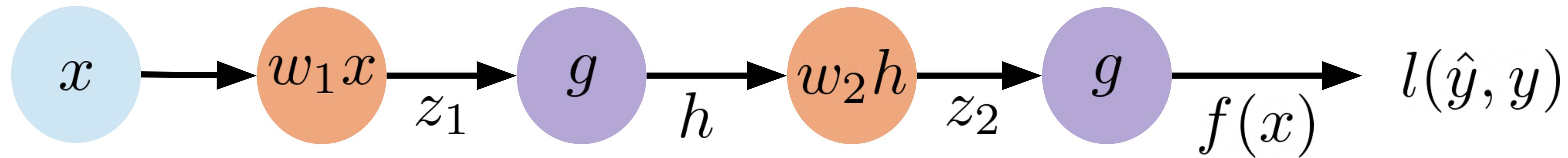
- Utilisation de l'algorithme « **Backpropagation** » pour ajuster les paramètres des réseaux de neurones afin de minimiser l'erreur de prédiction.
- Cette méthode fonctionne en calculant les gradients par rapport à l'erreur à chaque couche du réseau, puis en propageant ces gradients de manière rétroactive pour mettre à jour tous les poids.

Vidéo 3Blue1Brown: <https://www.youtube.com/watch?v=Ilg3gGewQ5U>

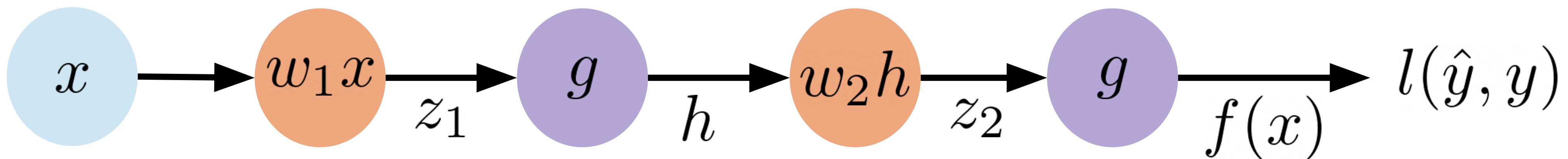
Backpropagation



Backpropagation



Backpropagation

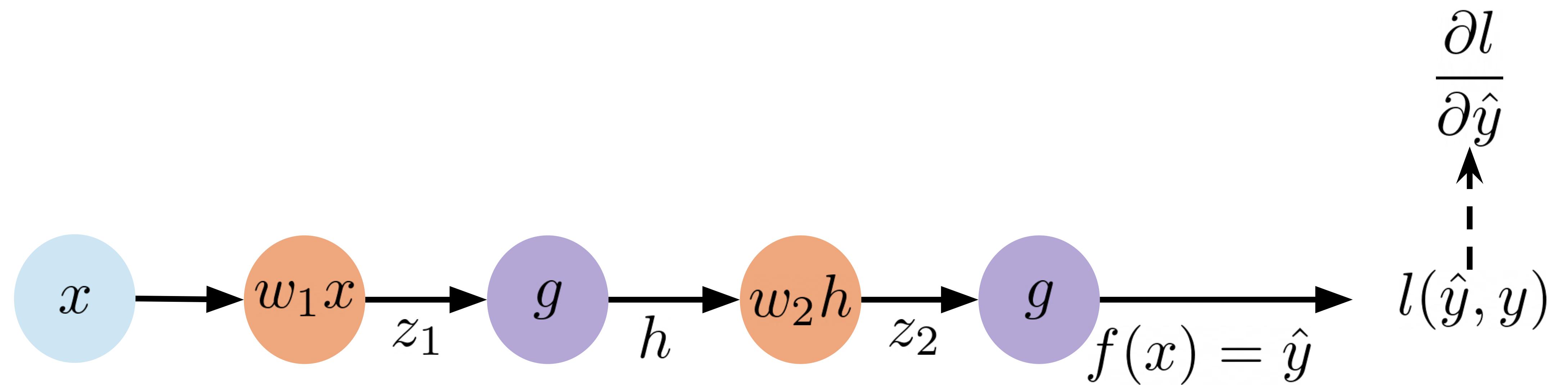


Question: Comment un petit changement des poids affecte la loss ?

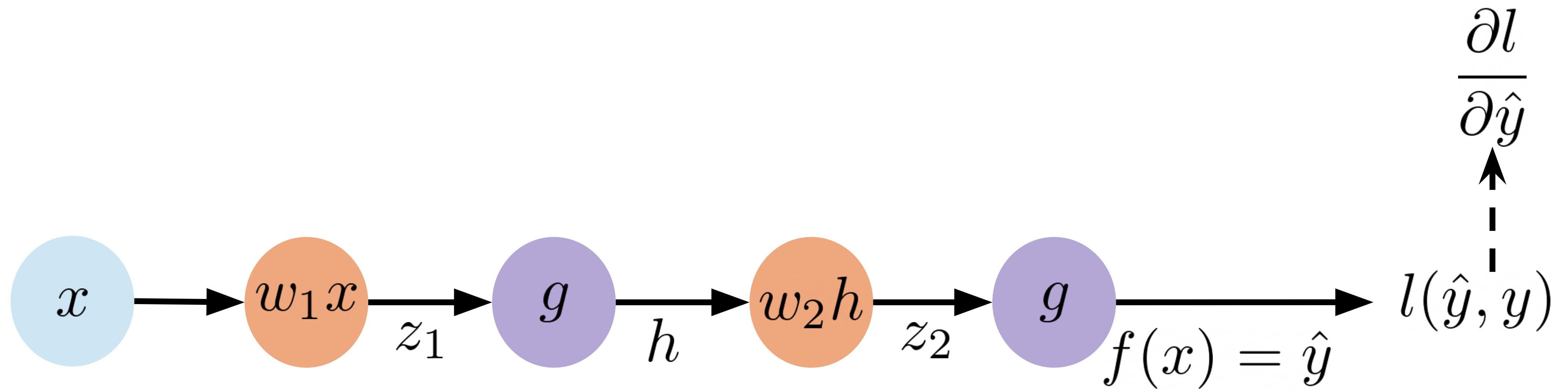
$$\frac{\partial l}{\partial w_1} = ?$$

$$\frac{\partial l}{\partial w_2} = ?$$

Backpropagation

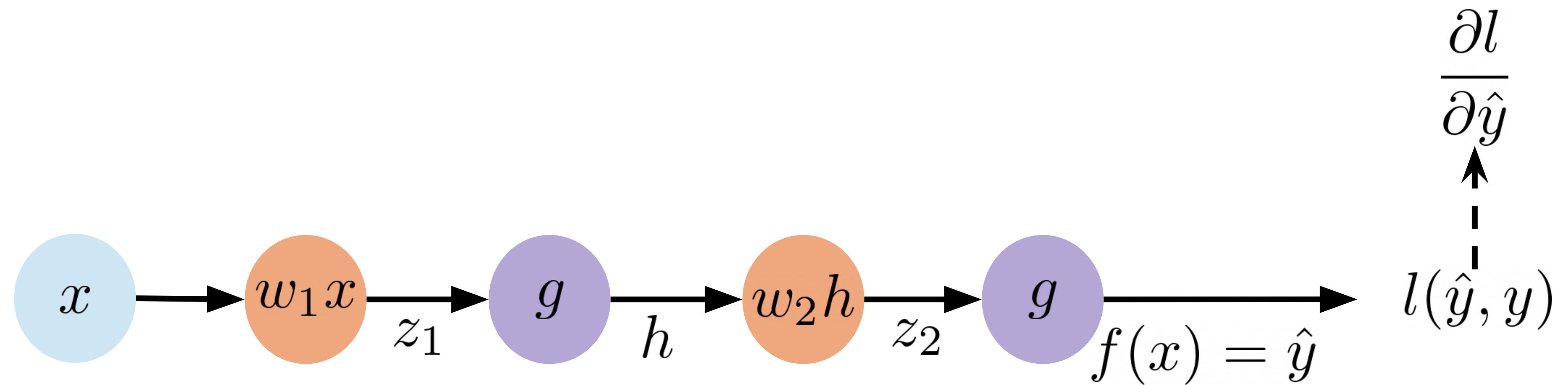


Backpropagation



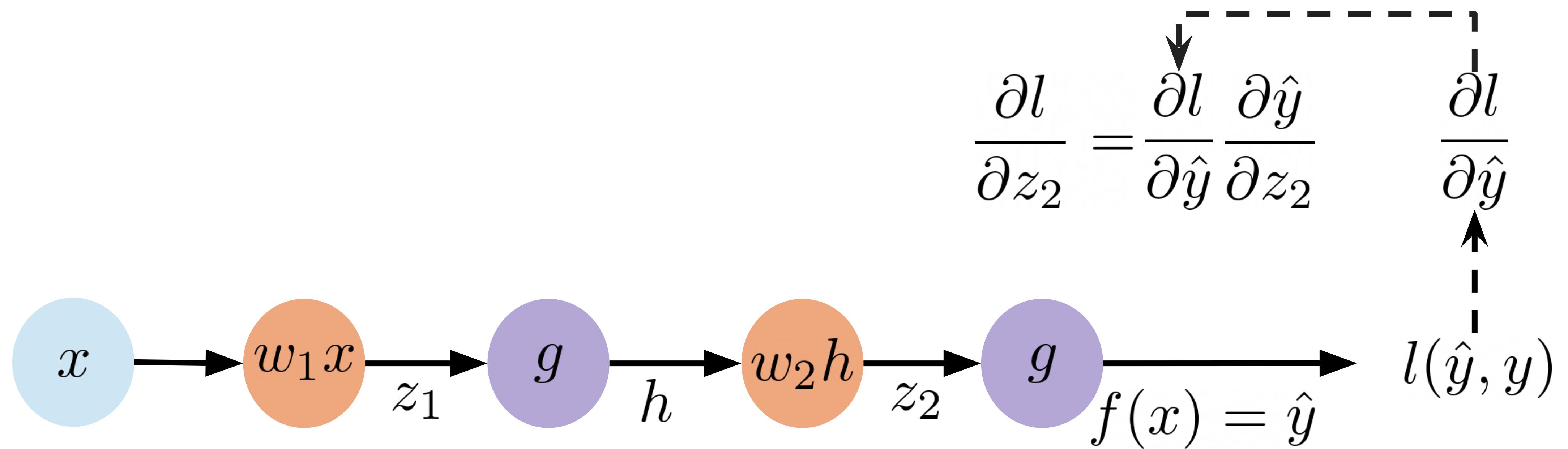
$$l(\hat{y}, y) = \|\hat{y} - y\|^2$$

Backpropagation

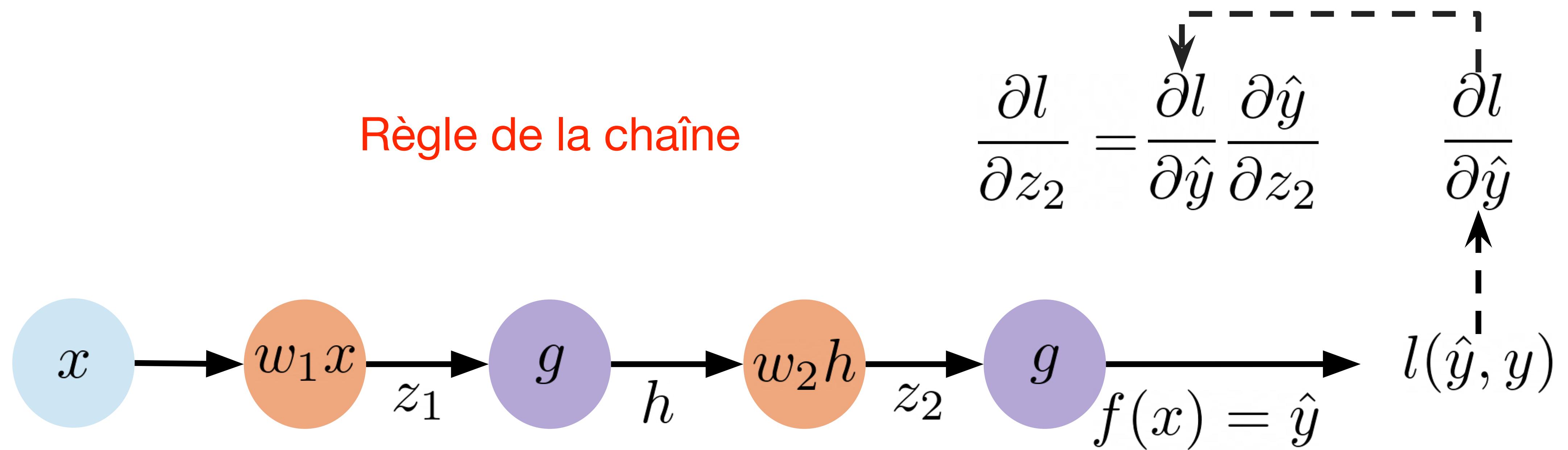


$$\hat{y} = g(z_2)$$

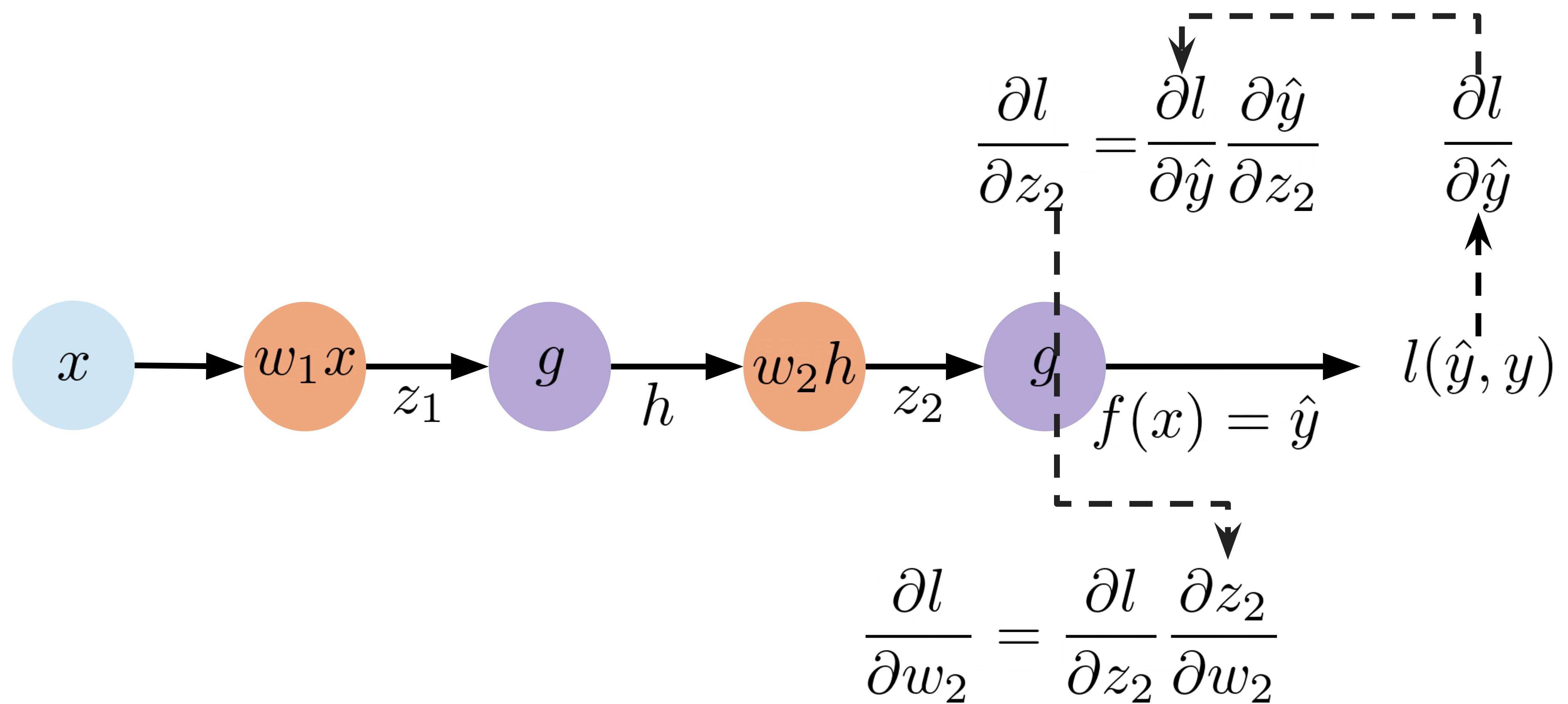
Backpropagation



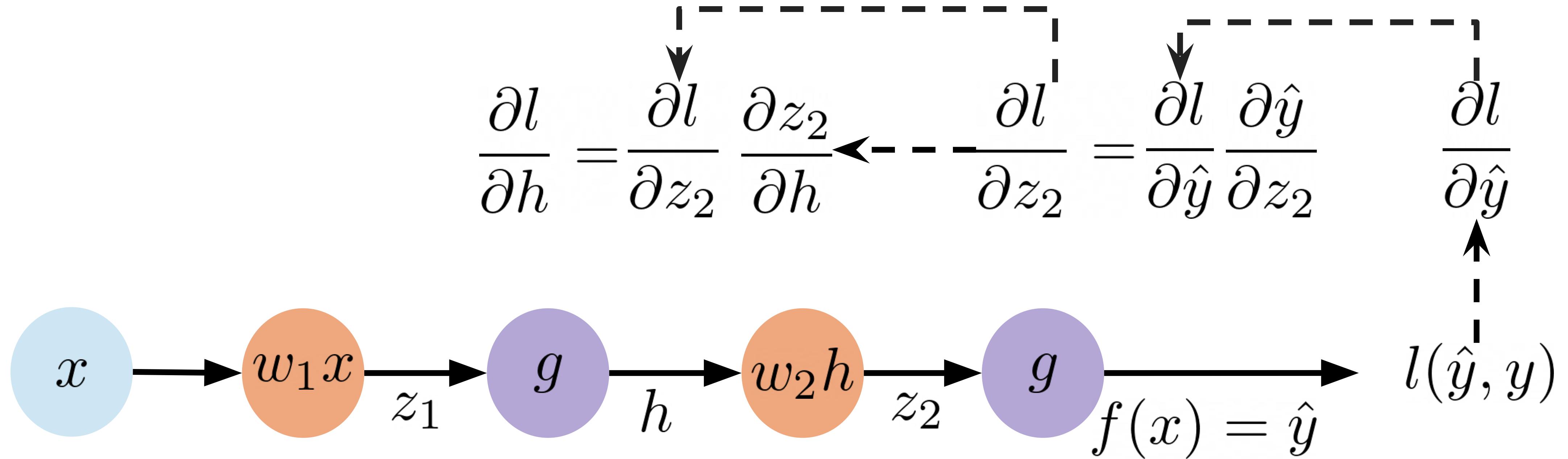
Backpropagation



Backpropagation

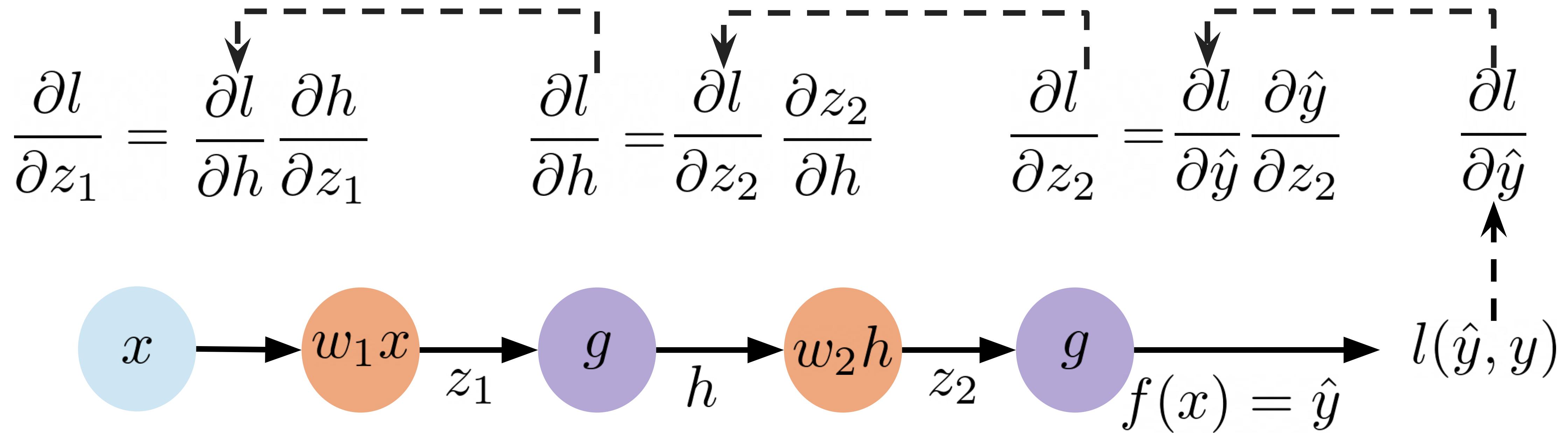


Backpropagation



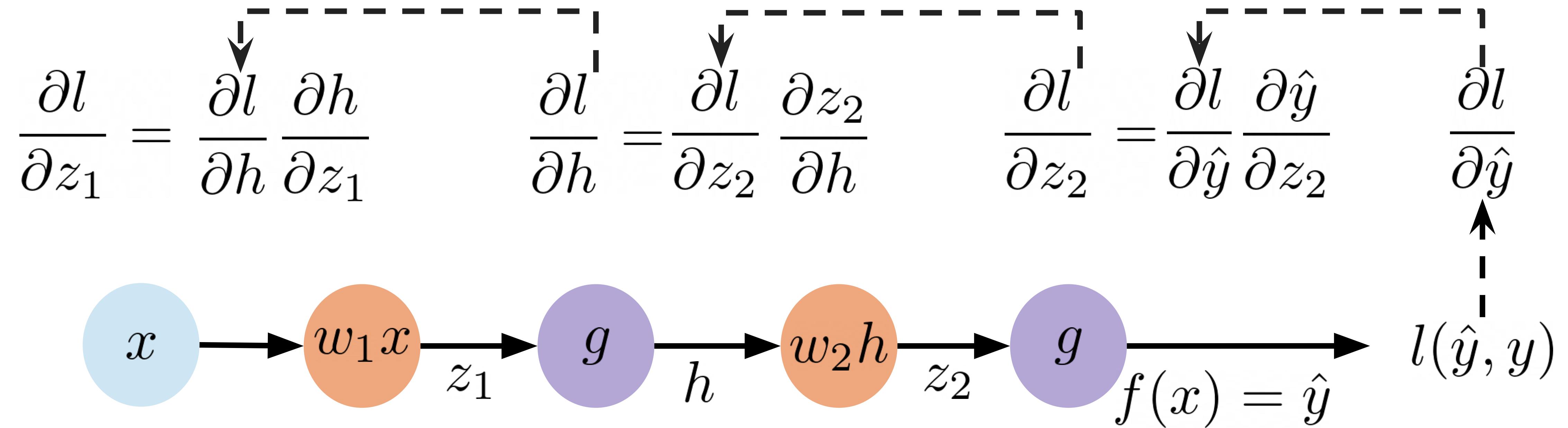
$$\frac{\partial l}{\partial w_2} = \frac{\partial l}{\partial z_2} \frac{\partial z_2}{\partial w_2}$$

Backpropagation



$$\frac{\partial l}{\partial w_2} = \frac{\partial l}{\partial z_2} \frac{\partial z_2}{\partial w_2}$$

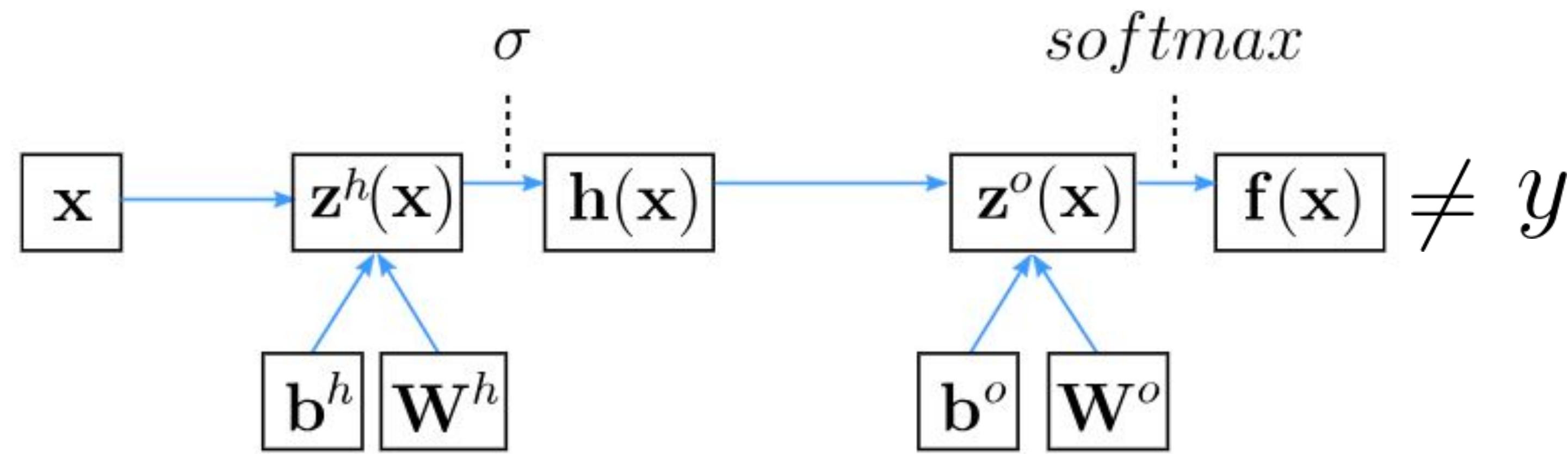
Backpropagation



$$\frac{\partial l}{\partial w_1} = \frac{\partial l}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

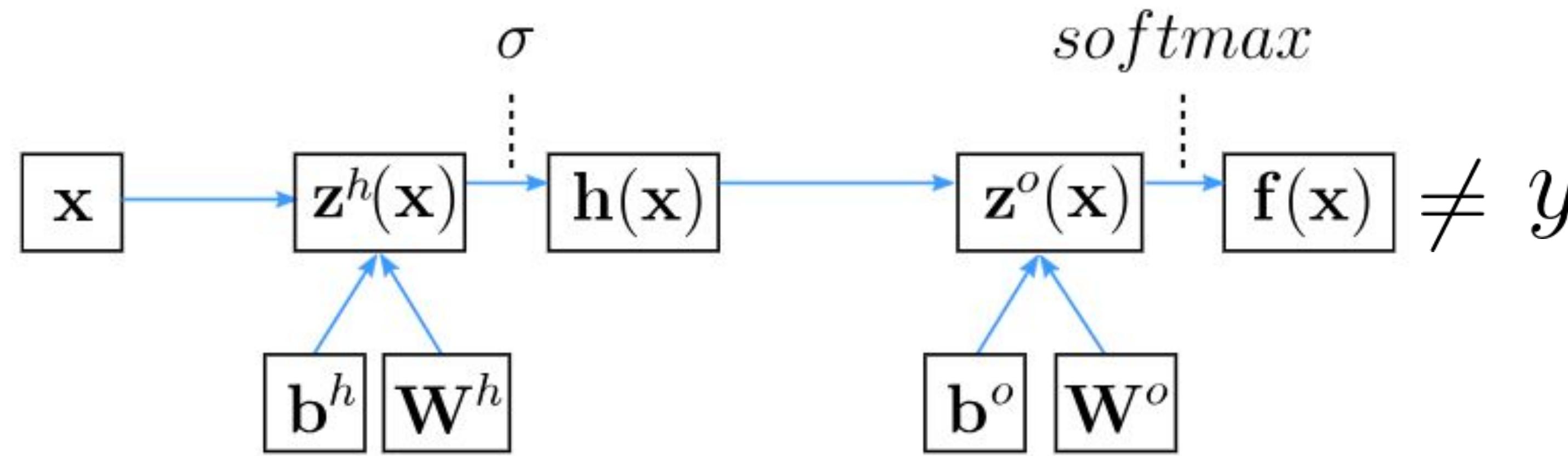
$$\frac{\partial l}{\partial w_2} = \frac{\partial l}{\partial z_2} \frac{\partial z_2}{\partial w_2}$$

Backpropagation



$$\nabla l = \left[\frac{\partial l}{\partial W^o}, \frac{\partial l}{\partial b^o}, \frac{\partial l}{\partial W^h}, \frac{\partial l}{\partial b^h} \right]$$

Backpropagation

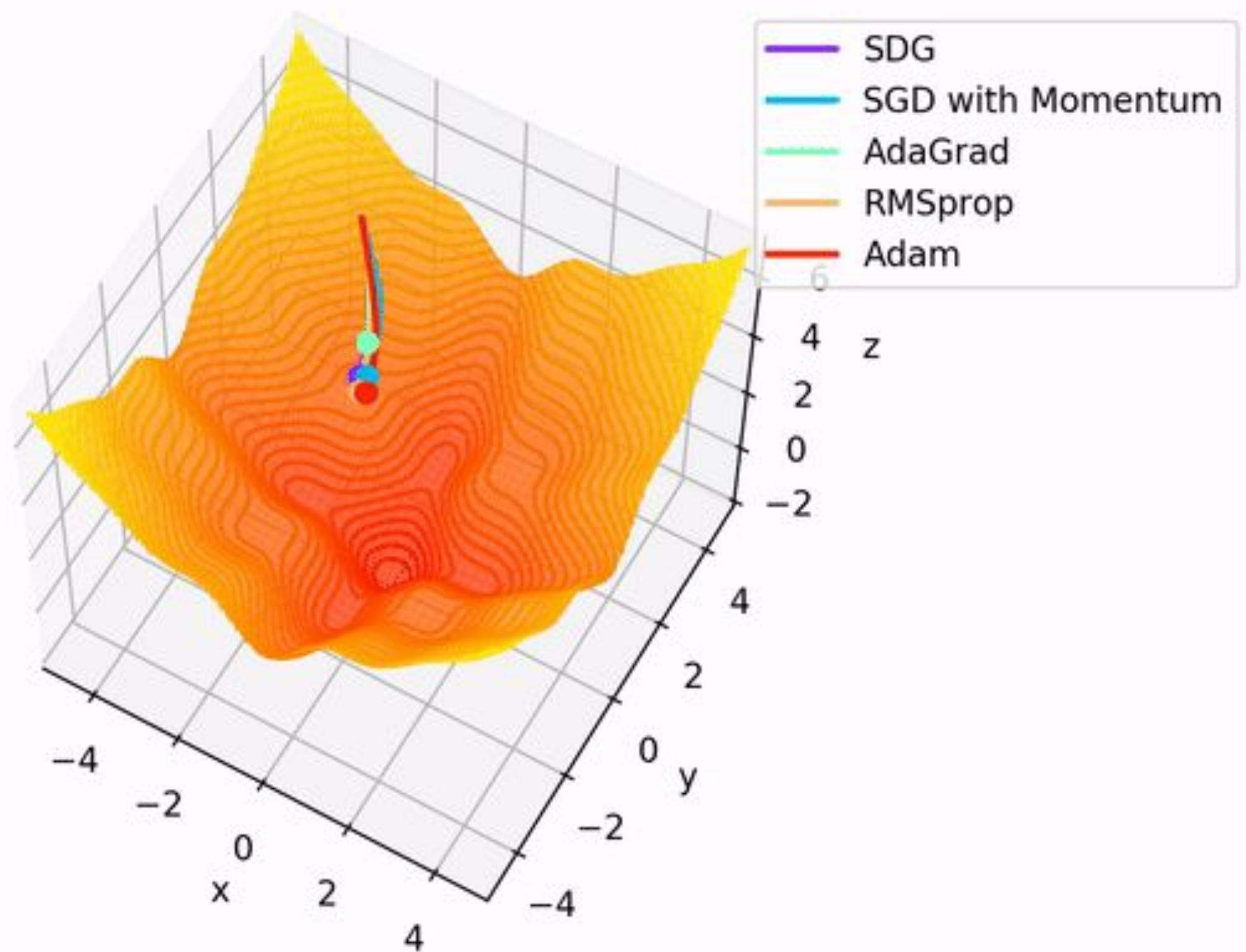


$$\theta \leftarrow \theta - \eta \delta \quad \delta = \nabla_{\theta} l(f(x), y) \quad \nabla l = \left[\frac{\partial l}{\partial W^o}, \frac{\partial l}{\partial b^o}, \frac{\partial l}{\partial W^h}, \frac{\partial l}{\partial b^h} \right]$$

Moyenne sur les données
d'entraînement

Optimizers

Optimizer Comparison



Optimizers

Stochastic Gradient Descent (SGD)

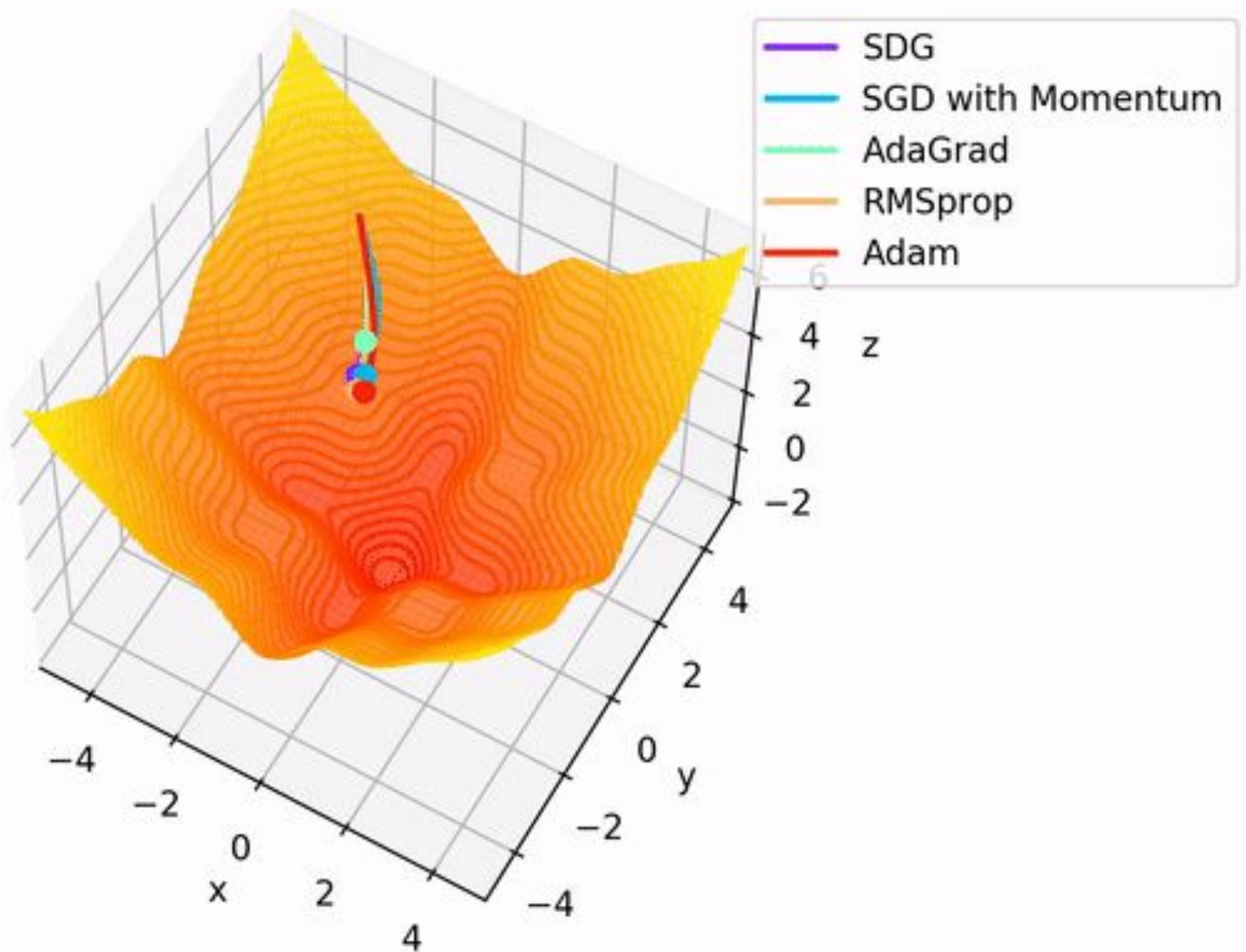
SGD + Moment

AdaGrad

RMSprop

Adam

Optimizer Comparison



Optimizers

Stochastic Gradient Descent (SGD)

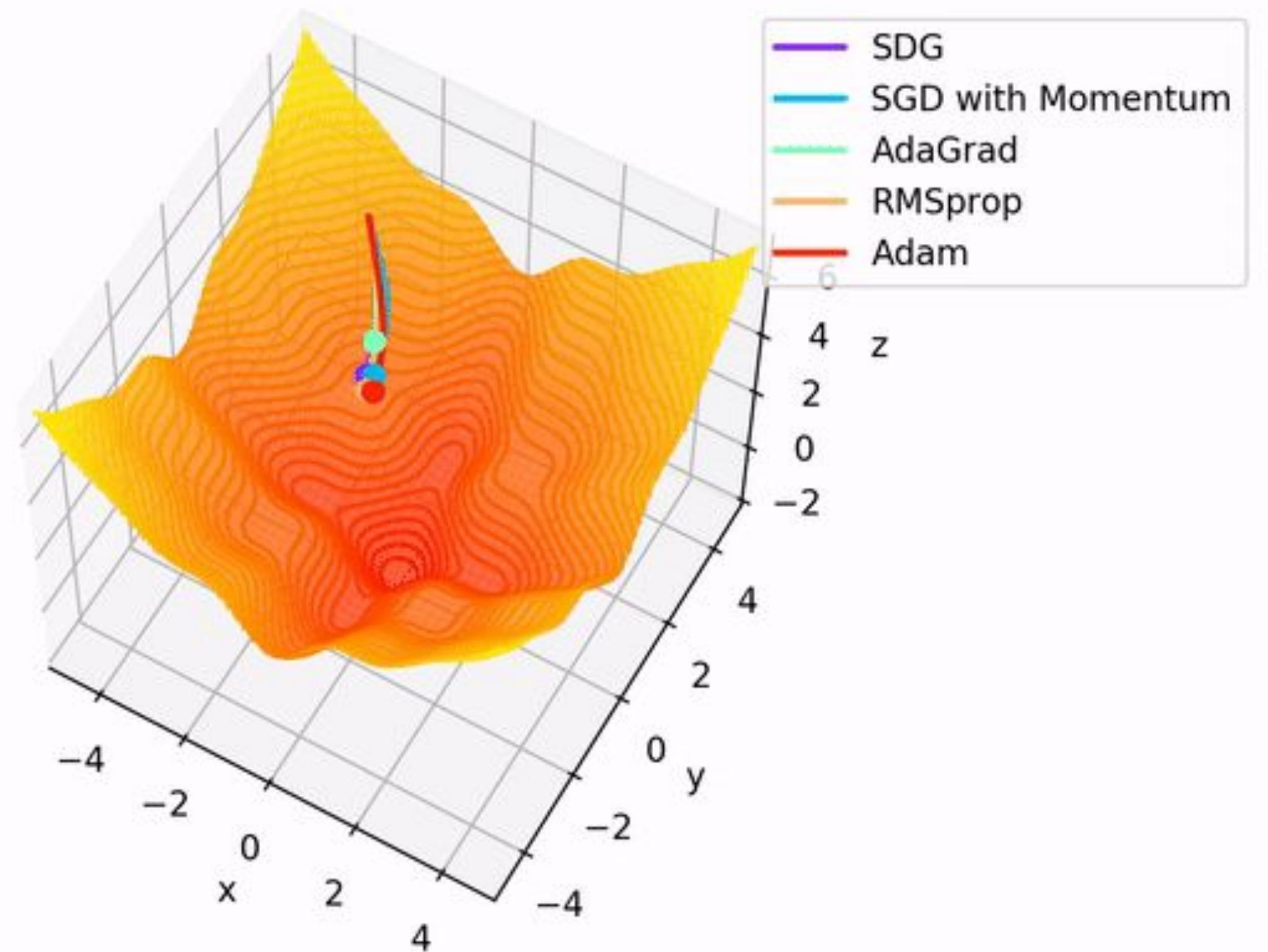
SGD + Moment

AdaGrad

RMSprop

Adam

Optimizer Comparison



Récap

Recette de l'entraînement d'un modèle

1. Préparer la donnée → Dataloader

1. Définir / construire le modèle → ??

1. Définir un objectif → ??

1. Optimiser les paramètres → ??

1. Evaluer

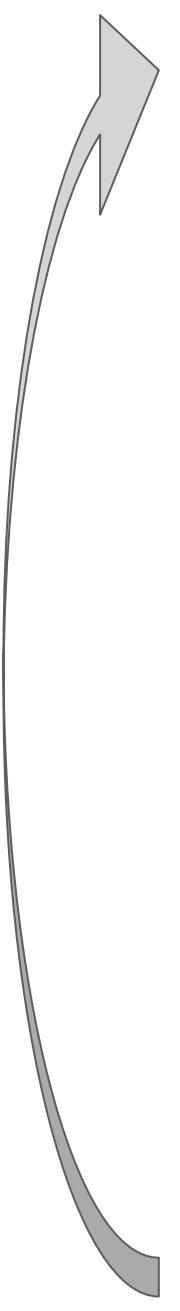
Récap

Recette de l'entraînement d'un modèle

- 
1. Préparer la donnée → Dataloader
 1. Définir / construire le modèle → Réseaux de neurones
 1. Définir un objectif → ??
 1. Optimiser les paramètres → ??
 1. Evaluer

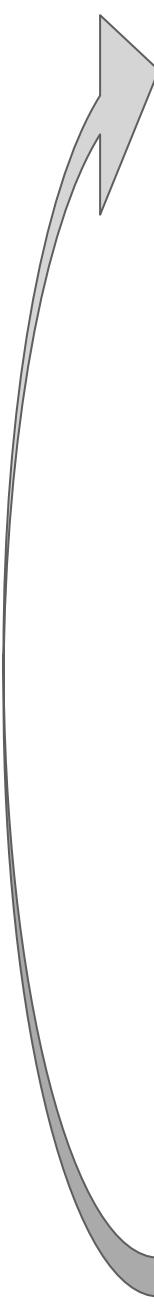
Récap

Recette de l'entraînement d'un modèle

- 
1. Préparer la donnée → Dataloader
 1. Définir / construire le modèle → Réseaux de neurones
 1. Définir un objectif → Fonction coût
 1. Optimiser les paramètres → ??
 1. Evaluer

Récap

Recette de l'entraînement d'un modèle

- 
1. Préparer la donnée → Dataloader
 1. Définir / construire le modèle → Réseaux de neurones
 1. Définir un objectif → Fonction coût
 1. Optimiser les paramètres → Boucle d'entraînement
 1. Evaluer

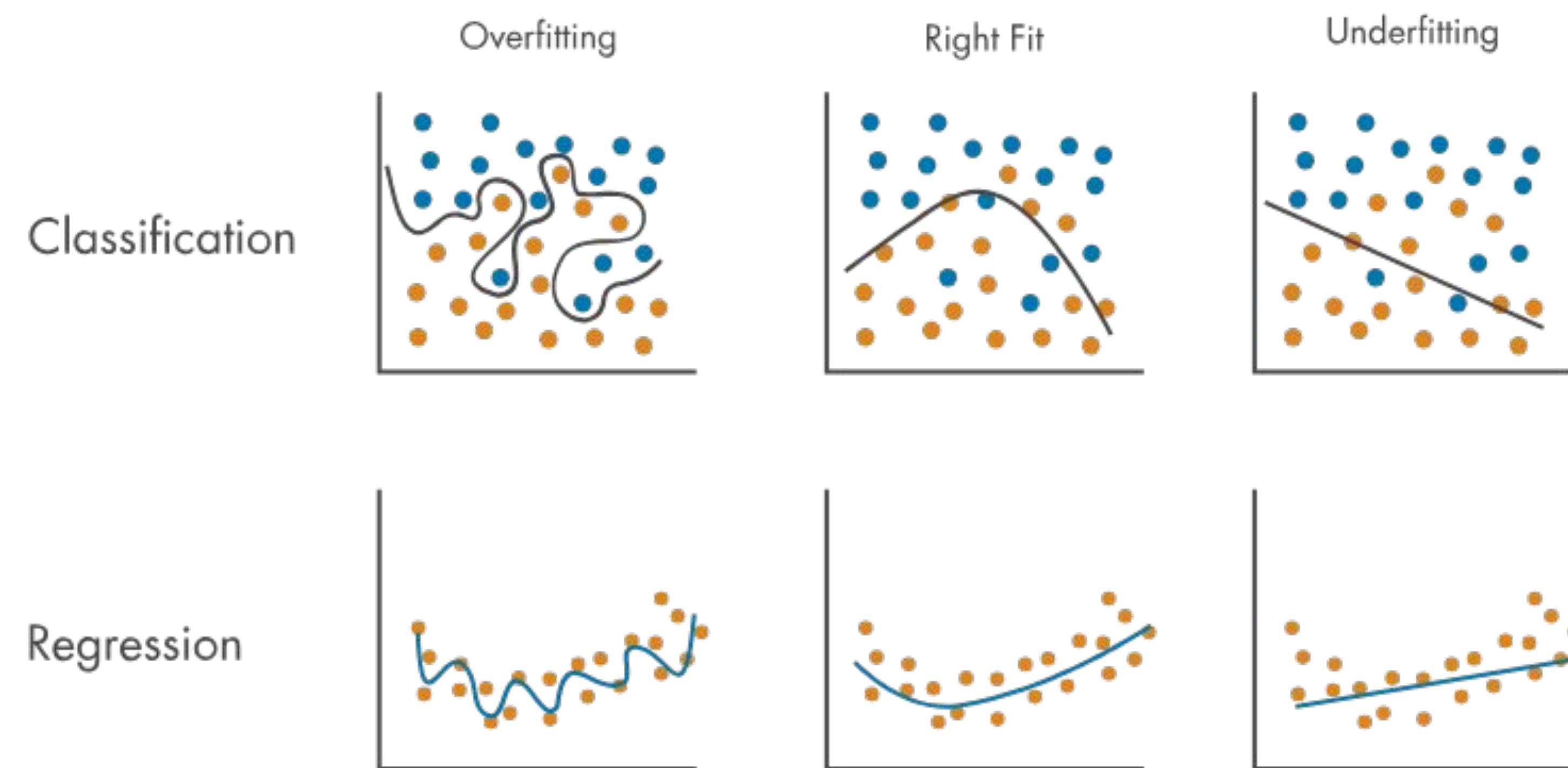
Batch size

- Le batch size fait référence au nombre de données utilisé pour la phase de « forward propagation » et « backward pass ».
- En deep learning, les jeux de données sont souvent assez volumineux et il n'est pas efficace d'introduire tout l'ensemble de données au réseau de neurones en une seule fois. Il est plus approprié d'utiliser des lots de données plus petit.
- Une taille de batch plus petite est plus efficace en termes de mémoire et peut parfois conduire à une convergence plus rapide mais toutefois peut introduire du bruit dans le processus d'apprentissage.
- Une taille de batch plus importante peut fournir une estimation plus précise du gradient, mais peut nécessiter plus de mémoire et de calculs.
- Le choix de la taille de batch est un hyperparamètre. Il est recommandé d'utiliser des valeurs qui sont des puissances de 2.

Nombre d'Epoch

- ◎ Une epoch correspond à un passage complet de l'entièr jeu de données dans le réseau de neurones.
- ◎ Le nombre d'epoch est un hyperparamètre important qui détermine le nombre de fois que le réseau de neurone sera entraîné à partir du jeu de données.
- ◎ Un entraînement avec un nombre d'epoch plus élevé peut conduire à de meilleure performance, mais le risque de sur-apprentissage (overfitting) augmente aussi.
- ◎ Cet hyperparamètre est généralement fixé en fonction de l'expérimentation et des performances sur le jeu de données de validation. L'idée est de continuer à entraîner le réseau jusqu'à que les performances sur les données de validation commencent à se dégrader.

Overfitting



Comment vaincre l'Overfitting ?

La lutte contre l'overfitting est un aspect critique en apprentissage profond pour s'assurer que les modèles généralisent bien sur de nouvelles données. Il existe plusieurs méthodes et techniques pour lutter contre l'overfitting dans l'apprentissage profond :

Comment vaincre l'Overfitting ?

La lutte contre l'overfitting est un aspect critique en apprentissage profond pour s'assurer que les modèles généralisent bien sur de nouvelles données. Il existe plusieurs méthodes et techniques pour lutter contre l'overfitting dans l'apprentissage profond :

- ➊ Plus de données

Comment vaincre l'Overfitting ?

La lutte contre l'overfitting est un aspect critique en apprentissage profond pour s'assurer que les modèles généralisent bien sur de nouvelles données. Il existe plusieurs méthodes et techniques pour lutter contre l'overfitting dans l'apprentissage profond :

- ◎ Plus de données
- ◎ Augmentation des données: Application de transformations aléatoires qui font sens sur l'ensemble d'apprentissage.



Comment vaincre l'Overfitting ?

La lutte contre l'overfitting est un aspect critique en apprentissage profond pour s'assurer que les modèles généralisent bien sur de nouvelles données. Il existe plusieurs méthodes et techniques pour lutter contre l'overfitting dans l'apprentissage profond :

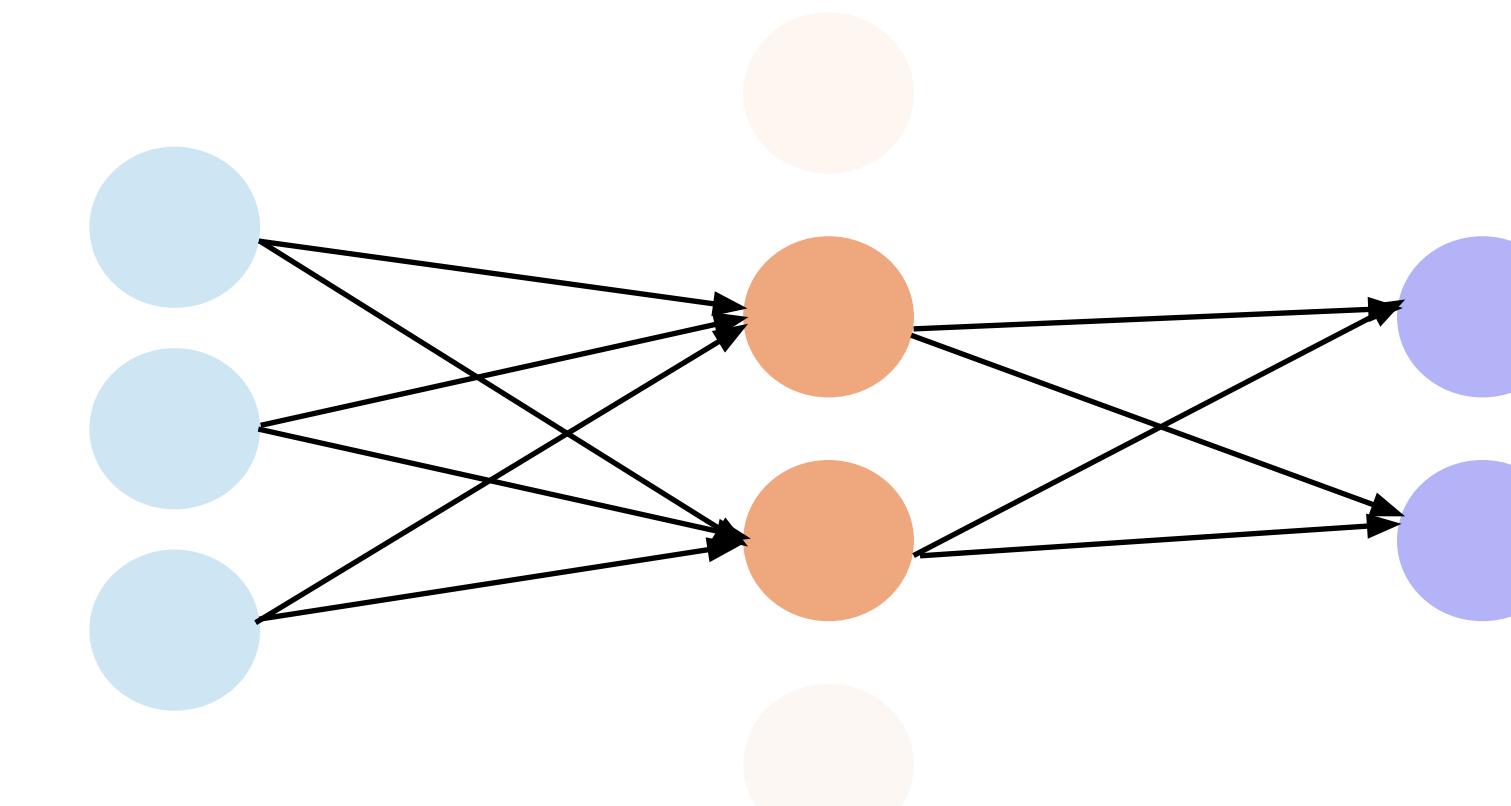
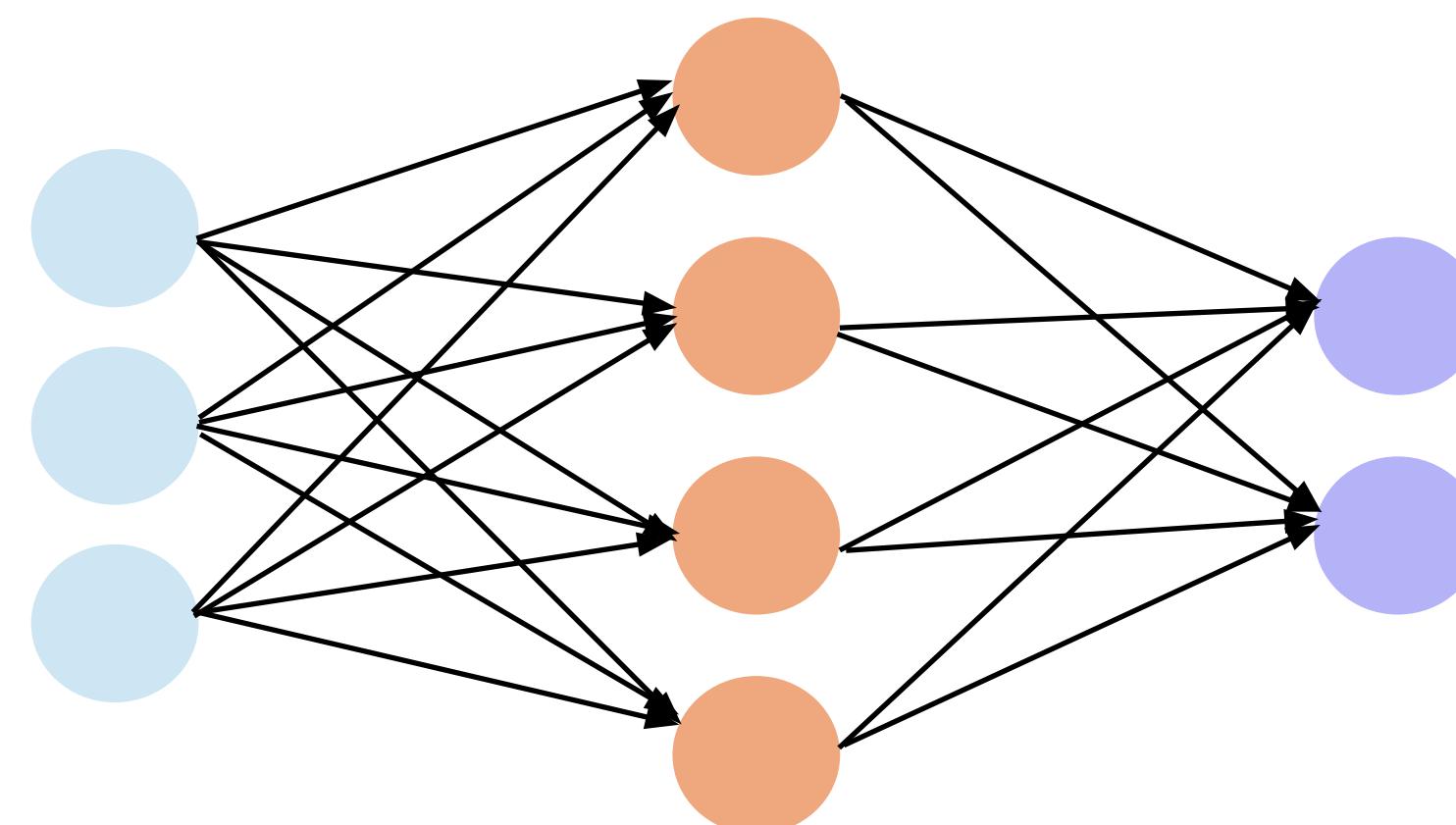
- ◎ **Plus de données**
- ◎ **Augmentation des données:** Application de transformations aléatoires qui font sens sur l'ensemble d'apprentissage.



- ◎ **Régularisation de la fonction coût:** Ajouter un pénalité pour les valeurs extrême

Tricks: Dropout

- Mettre aléatoirement un pourcentage des neurones à zéros pendant l'entraînement.
- Cela permet d'entraîner un modèle plus robuste et qui généralise mieux en évitant qu'un seul neurone ou groupe de neurones ne devienne trop spécialisé et trop dépendant des données d'apprentissage.



Mise à 0 de 50% des neurones
dans la couche cachée

Tricks: Batch Normalization

- « Batch normalization » (BatchNorm) est une technique d'apprentissage profond utilisée pour améliorer l'entraînement des réseaux neuronaux.
- Elle fonctionne en normalisant l'entrée de chaque couche dans un mini batch de données.
- Permet une convergence plus rapide pendant l'apprentissage, ce qui implique un entraînement de réseau de neurones plus profonds de manière plus efficace.

Tricks: Early Stopping

- Le « early stopping » ou en français l'arrêt anticipé est une stratégie visant à interrompre le processus d'apprentissage d'un réseau de neurones avant qu'il n'ait terminé toutes les epochs.
- Il s'agit d'évaluer périodiquement les performances du modèle sur un ensemble de données de validation distinct des données d'apprentissage.
- La décision d'arrêter l'entraînement est basée sur une mesure de performance choisie (par exemple, les valeurs de la fonction de coût sur les données de validation ou à partir de mesure tel que l'accuracy).
- L'objectif est de détecter le moment où les performances du modèle sur l'ensemble de validation commencent à se détériorer ou à atteindre un plateau.

TP 1: MLP et FC pour classification MNIST

