Intelligence Artificielle
Réseaux de neurones & Deep Learning

Akka Zemmari
zemmari@u-bordeaux.fr

LaBRI, Université de Bordeaux - CNRS

2019-2020

Contenu

Introduction

D'abords il y a le neurone

... puis les réseaux de neurones

et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

Outline

Introduction

D'abords il y a le neurone

... puis les réseaux de neurones

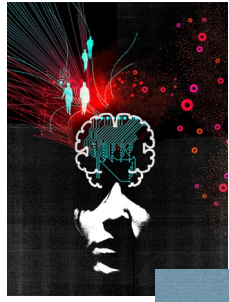
et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

Deep Learning



[Deep Dream Generator / Le Monde 2017]



[MIT review technology, 2017]

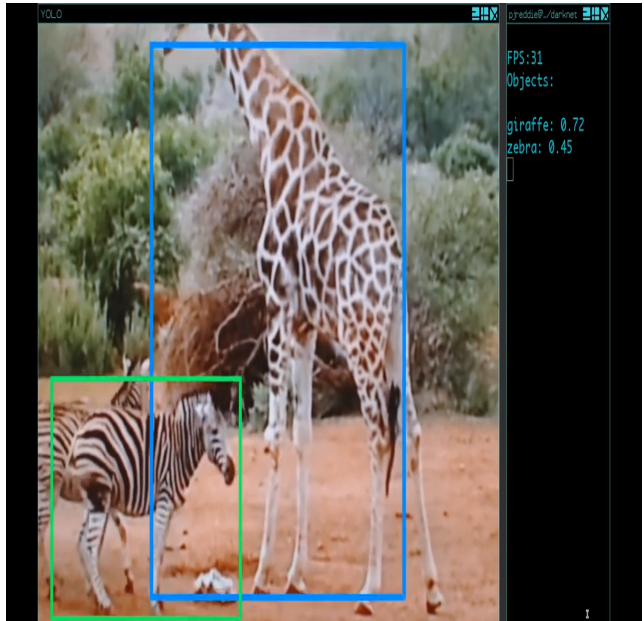


[New Economist, 2015]

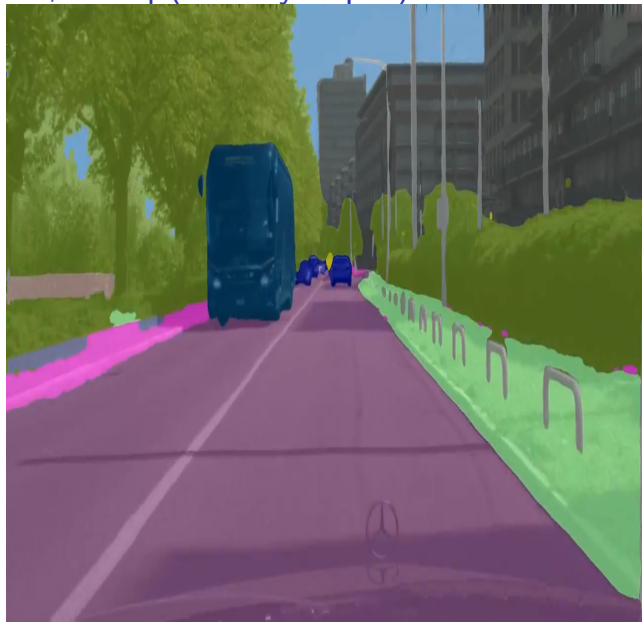


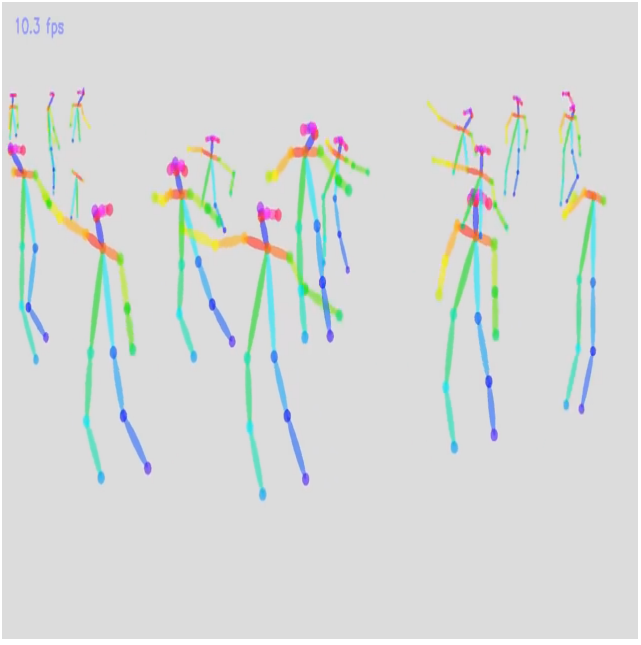
[L'Express, 2017]

Classification+Localization

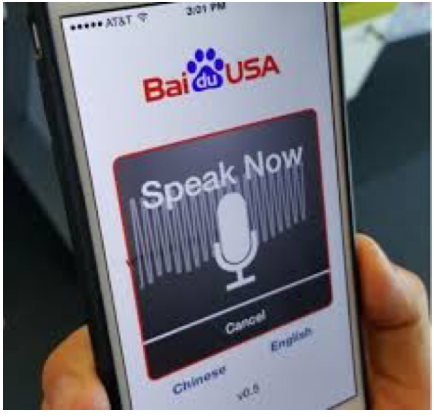


RefineNet [Lin et al, 2016] (on Cityscapes)



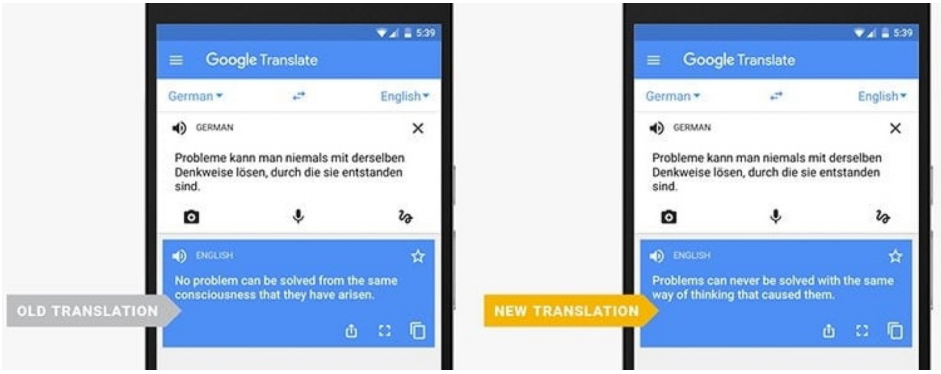

10.3 fps

Deep Speech

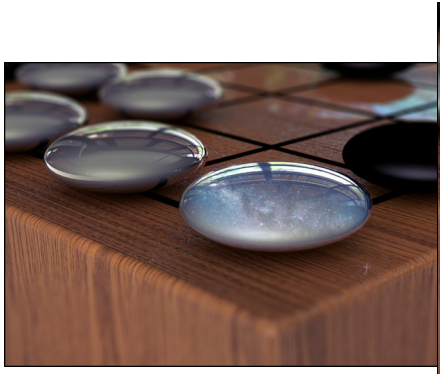


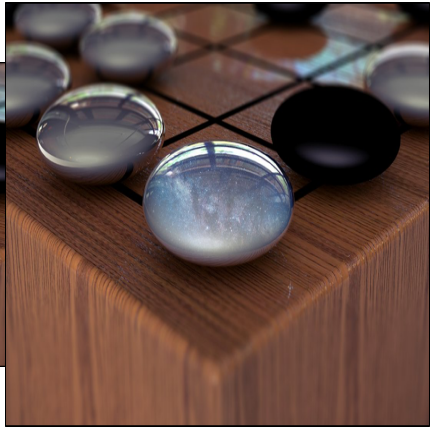
Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin

Machine Translation



Alpha Go





Et quelques intérrogations ...

Microsoft débranche son robot intelligent devenu raciste et pro-nazis en 24 heures

25 mars 2016, 10:58



Et quelques intérrogations ...



Et quelques intérrogations ...

http://moralmachine.mit.edu

Quelques dates et quelques noms

- 1943 : (Mc Culloch (neuro-physiologiste) et Pitts (logicien)) proposent les premières notions de neurone formel.
- 1959 : (Rosenblatt) définit un réseau de neurones avec une couche d'entrée et une sortie.
- 1960 : (Widrow et Hoff) ADALINE
- 1969 : (Minsky, Papert) Problème XOR
- 1986 : (Rumelhart et. al) MLP et backpropagation
- 1992 : (Vapnik et. al) SVM
- 1998 : (LeCun et. al) LeNet
- 2010 : (Hinton et. al) Deep Neural Networks
- 2012 : (Krizhevsky, Hinton et. al) AlexNet, ILSVRC'2012, GPU - 8 couches 2014 GoogleNet - 22 couches
- 2015 : Inception (Google) - Deep Dream
- 2016 : ResidualNet (Microsoft/Facebook) - 152 couches

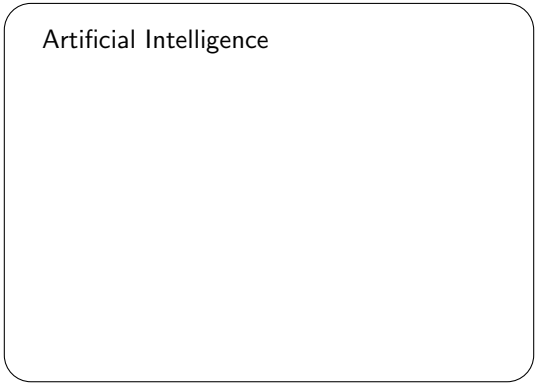
The New York Times

Turing Award Won by Three Pioneers in Artificial Intelligence

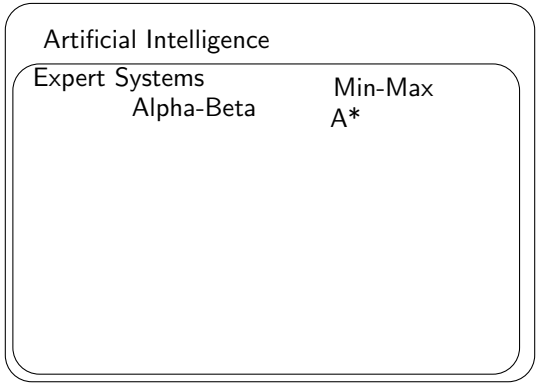


From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built.

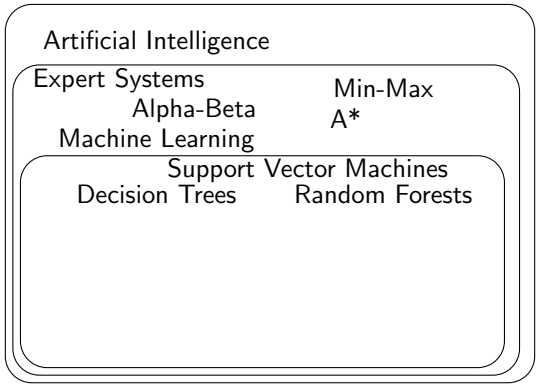
Artificial Intelligence/Machine Learning/Deep Learning



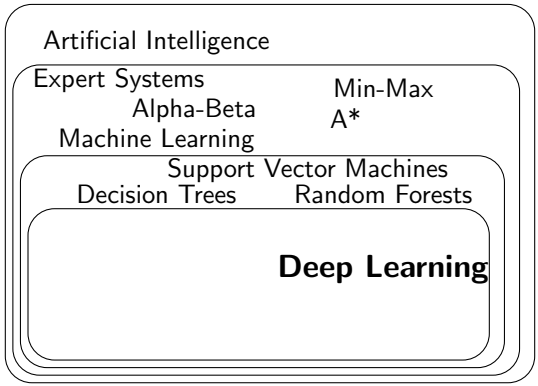
Artificial Intelligence/Machine Learning/Deep Learning



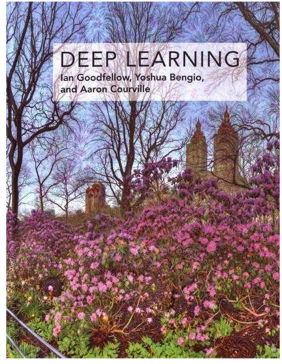
Artificial Intelligence/Machine Learning/Deep Learning



Artificial Intelligence/Machine Learning/Deep Learning



Références du cours - Livres



■ **Deep Learning** ■,
I. Goodfellow, Y. Bengio, A. Courville,

Outline

Introduction

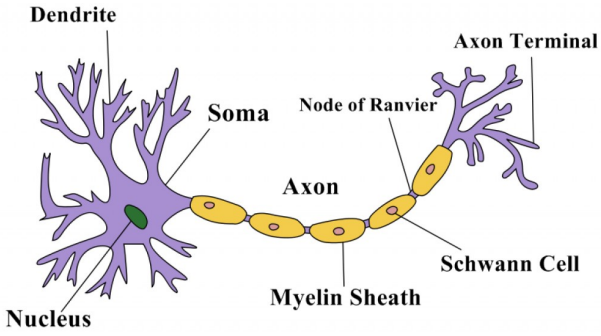
D'abords il y a le neurone

... puis les réseaux de neurones

et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

D'abords il y a le neurone



D'abords il y a le neurone

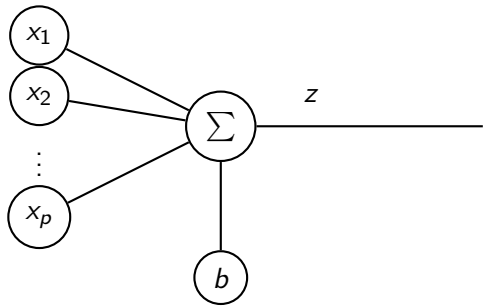


Figure – Un neurone formel.

D'abords il y a le neurone

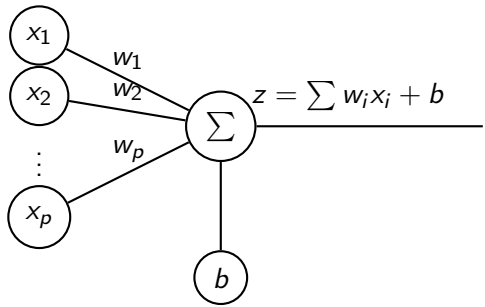


Figure – Un neurone formel.

DEMO

D'abords il y a le neurone

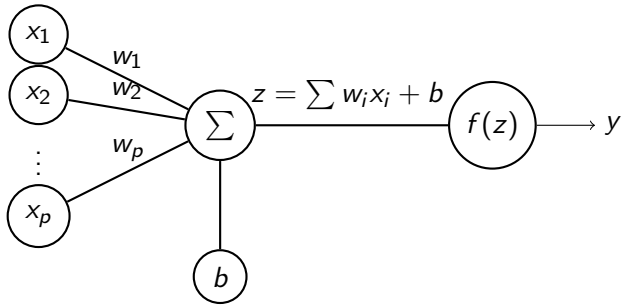


Figure – Un neurone formel.

D'abords il y a le neurone

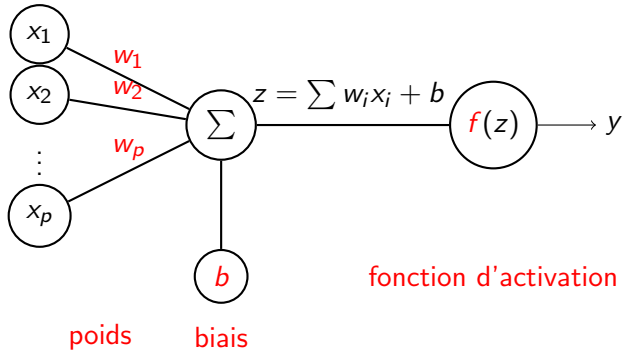
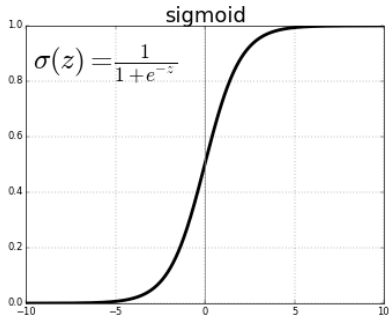


Figure – Un neurone formel.

D'abords il y a le neurone

Exemple de fonction d'activation : sigmoïd

$$f(z) = \frac{1}{1 + e^{-z}}.$$



DEMO

D'abords il y a le neurone

Autres fonctions d'activation :

- fonction linéaire : f est la fonction identité,
- seuil : $f(z) = \mathbb{1}_{[0,\infty[}(z)$,
- ReLU : $f(z) = \max(0, z)$ (rectified linear unit),
- radiale : $f(z) = \sqrt{1/2\pi}e^{(-z^2/2)}$,
- stochastique : $f() = 1$ avec la probabilité $1/(1 + e^{-z/H})$, 0 sinon (H intervient comme une température dans un algorithme de recuit simulé),
- ...

D'abords il y a le neurone

De manière compacte :

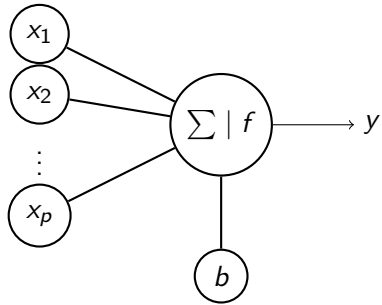


Figure – Un neurone formel.

Outline

Introduction

D'abords il y a le neurone

... puis les réseaux de neurones

et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

Réseau de neurones

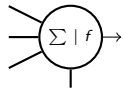


Figure – Un réseau de neurones.

Réseau de neurones

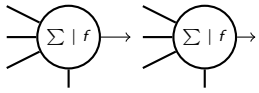


Figure – Un réseau de neurones.

Réseau de neurones

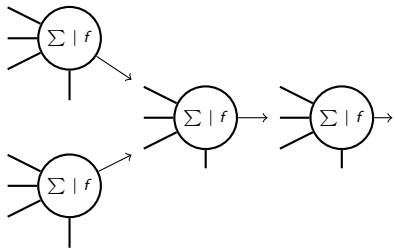


Figure – Un réseau de neurones.

Réseau de neurones

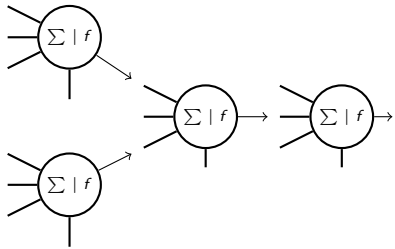


Figure – Un réseau de neurones.

- Les neurones ont chacun leurs propres poids et biais,

Réseau de neurones

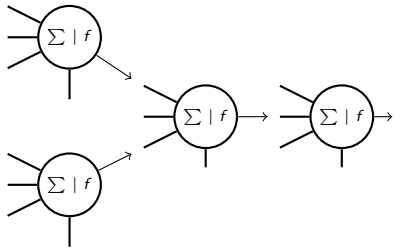
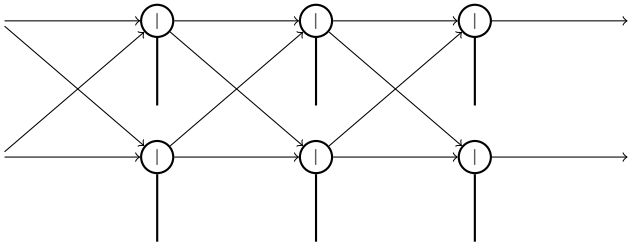


Figure – Un réseau de neurones.

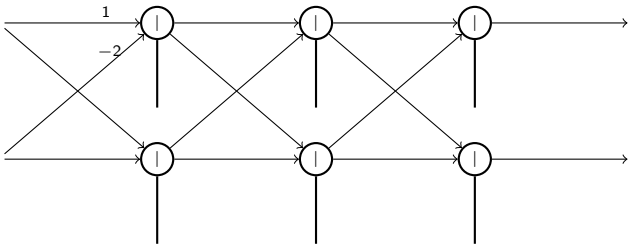
- Les neurones ont chacun leurs propres poids et biais,
- Les paramètres du réseau sont alors les différents poids et les différents biais. On note tous ces paramètres θ.

DEMO

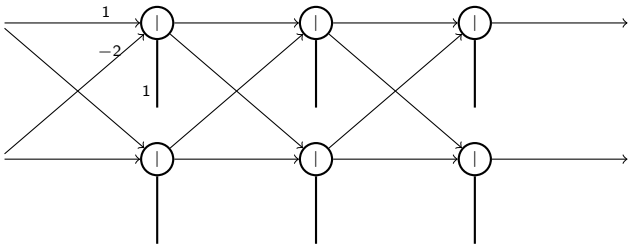
Réseaux de neurones complètement connectés



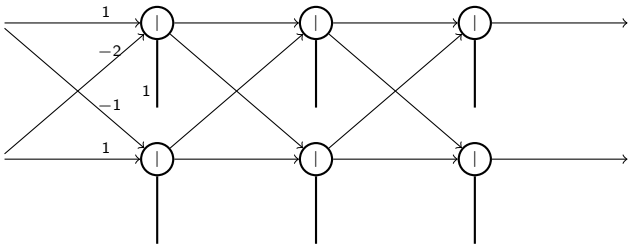
Réseaux de neurones complètement connectés



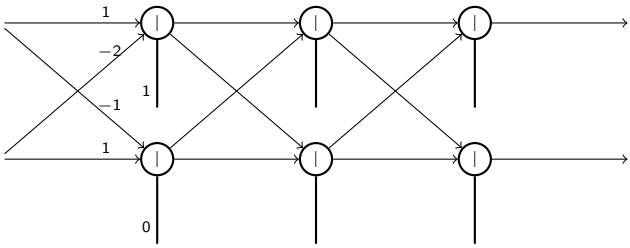
Réseaux de neurones complètement connectés



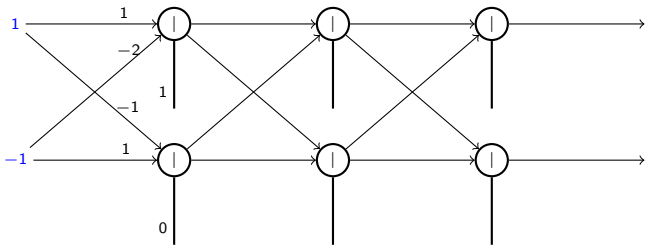
Réseaux de neurones complètement connectés



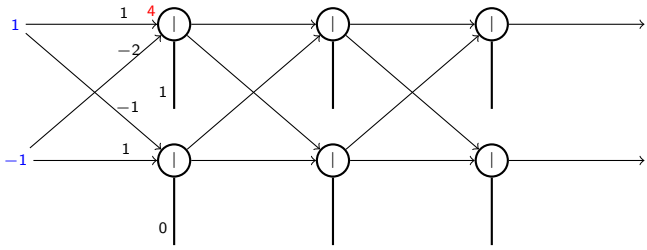
Réseaux de neurones complètement connectés



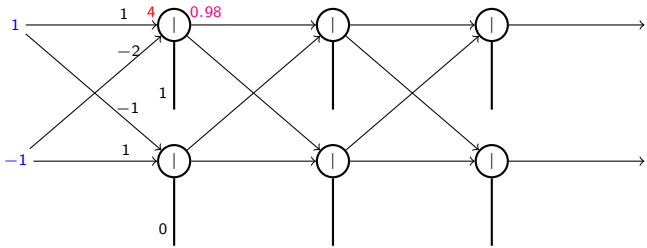
Réseaux de neurones complètement connectés



Réseaux de neurones complètement connectés

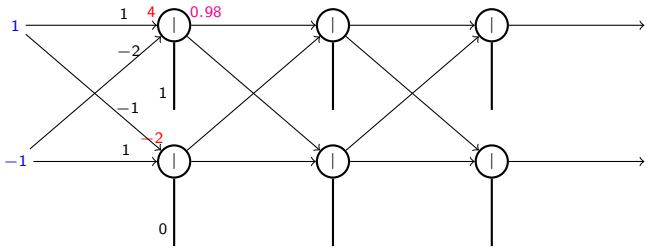


Réseaux de neurones complètement connectés

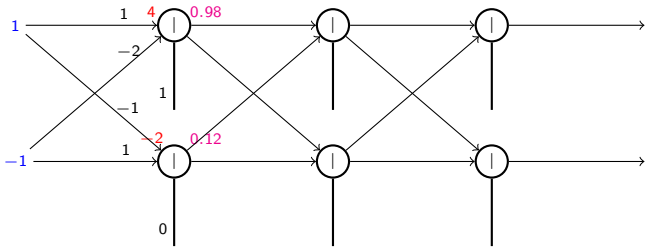


Si f est la fonction sigmoïd : $f(z) = \frac{1}{1-e^{-z}}$

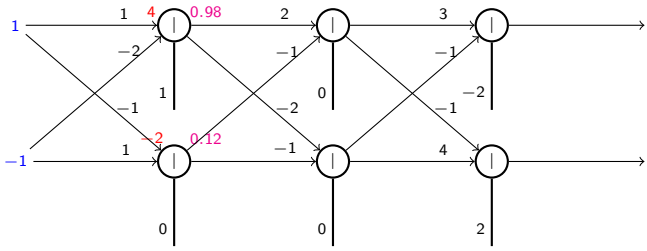
Réseaux de neurones complètement connectés



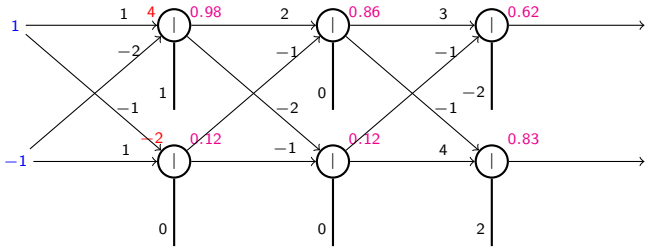
Réseaux de neurones complètement connectés



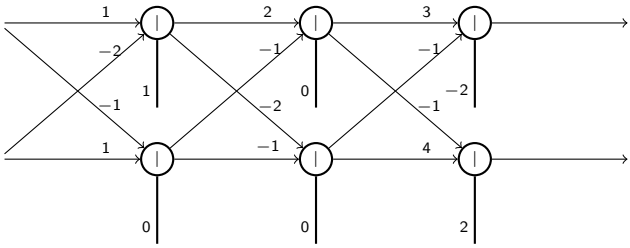
Réseaux de neurones complètement connectés



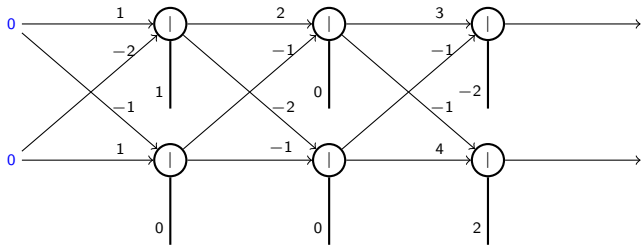
Réseaux de neurones complètement connectés



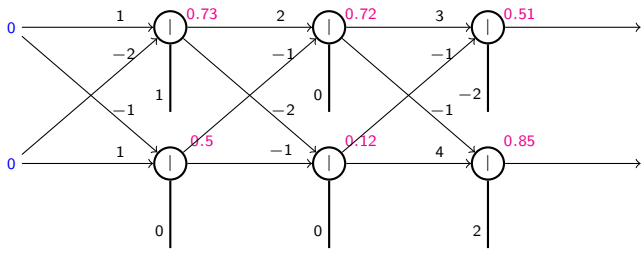
Réseaux de neurones complètement connectés



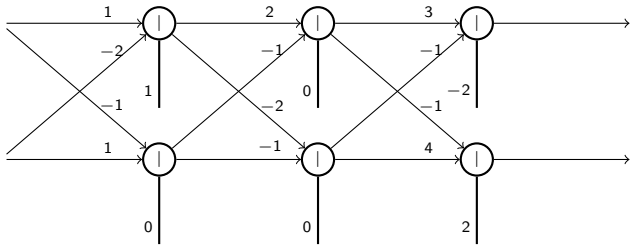
Réseaux de neurones complètement connectés



Réseaux de neurones complètement connectés

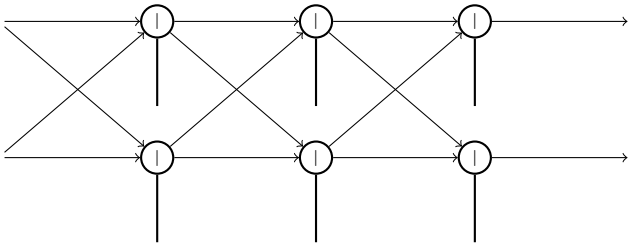


Réseaux de neurones complètement connectés



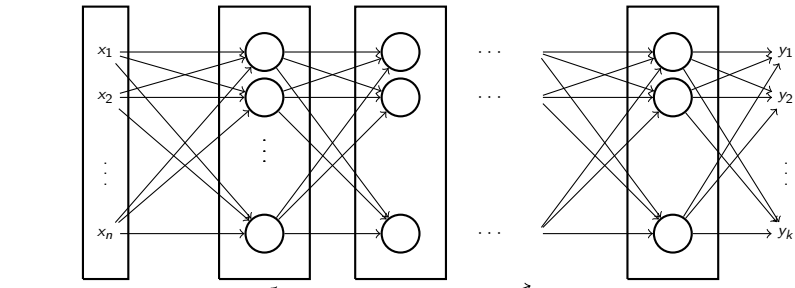
Fixer les paramètres $\theta \longrightarrow$ fixer la fonction.

Réseaux de neurones complètement connectés



Donner la structure du réseau $\rightarrow$ définir un ensemble de fonctions.

Réseaux à plusieurs couches



Apprentissage des paramètres

- Question :
  Comment déterminer les paramètres des neurones $w = (w_1, w_2, \cdots, w_p)$ et b.
- Réponse :
  Descente du gradient

Apprentissage des paramètres

- Question :
  Comment déterminer les paramètres des neurones $w = (w_1, w_2, \cdots, w_p)$ et b.
- Réponse :
  Descente du gradient patience ...

Outline

Introduction

D'abords il y a le neurone

... puis les réseaux de neurones

et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

Deep Learning = (plusieurs) couches cachées.

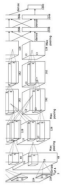
Exemples de réseaux connus

Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers

16.4%



AlexNet (2012)

19 layers

7.3%



VGG (2014)

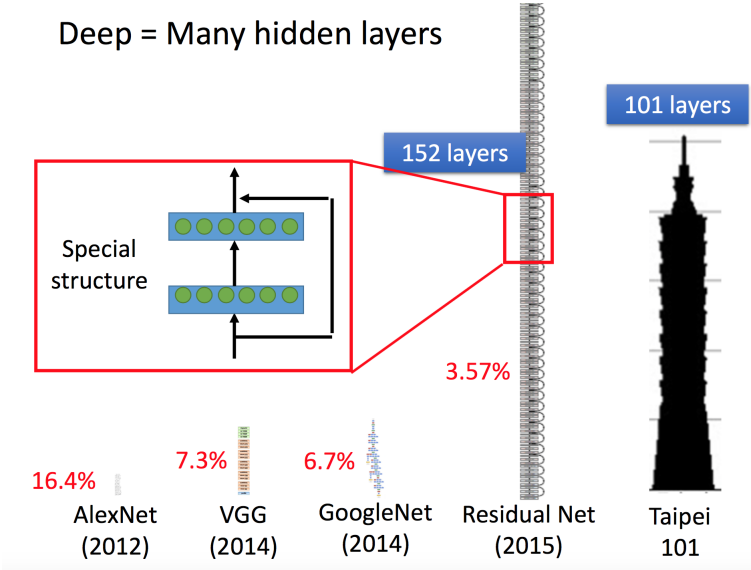
22 layers

6.7%



GoogleNet (2014)

Exemples de réseaux connus



Pourquoi le deep

Un exemple :

- Apprendre une fonction AND : une seule couche

Pourquoi le deep

Un exemple :

- Apprendre une fonction AND : une seule couche
- Apprendre une fonction OR : une seule couche

Pourquoi le deep

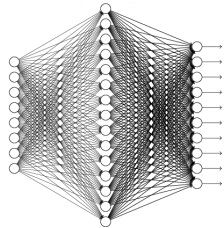
Un exemple :

- Apprendre une fonction AND : une seule couche
- Apprendre une fonction OR : une seule couche
- Apprendre une fonction XOR : une seule couche ne suffit pas ...

Pourquoi le deep

Théorème d'universalité

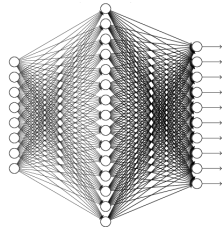
Toute fonction continue $f : \mathbb{R}^n \to \mathbb{R}^n$ peut être approximée par un réseau de neurones à une couche cachée.



Pourquoi le deep

Théorème d'universalité

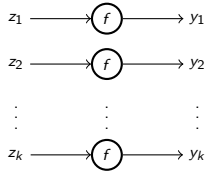
Toute fonction continue $f : \mathbb{R}^n \to \mathbb{R}^n$ peut être approximée par un réseau de neurones à une couche cachée.



Question : pourquoi des réseaux profonds au lieu de réseaux à une couche ?

Couche de sortie

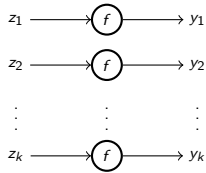
Ce que nous avons vu jusque là :



Problème : les sorties y_i prennent des valeurs quelconques et difficiles à interpréter ...

Couche de sortie

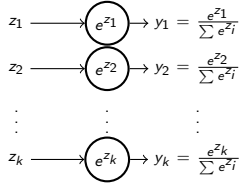
Ce que nous avons vu jusque là :



Problème : les sorties y_i prennent des valeurs quelconques et difficiles à interpréter ...
Solution : transformer les sorties en "probabilités".

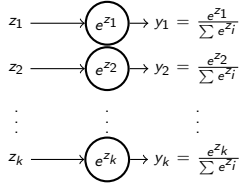
Couche de sortie

La couche de Softmax :
Solution : ajouter une couche de Softmax :



Couche de sortie

La couche de Softmax :
Solution : ajouter une couche de Softmax :

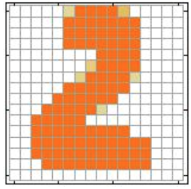


On obtient ainsi une distribution de probabilité :

- $0 < y_i < 1, \ \forall i,$
- $\sum_{i=1}^{k} y_i = 1.$

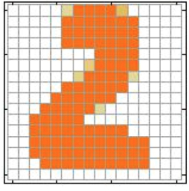
Exemple : reconnaissance d'écriture

Input **Output**



Exemple : reconnaissance d'écriture

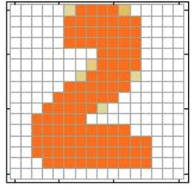
Input **Output**



16 x 16 = 256

Exemple : reconnaissance d'écriture

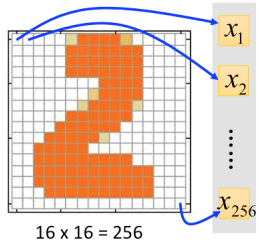
Input **Output**



16 x 16 = 256

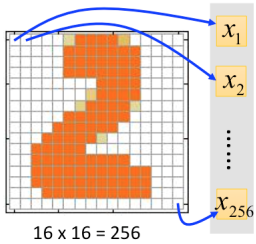
Exemple : reconnaissance d'écriture

Input **Output**



Exemple : reconnaissance d'écriture

Input **Output**



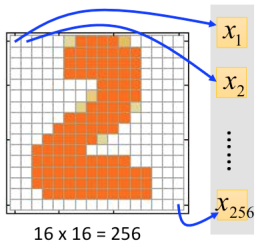
16 x 16 = 256

Ink → 1

No ink → 0

Exemple : reconnaissance d'écriture

Input

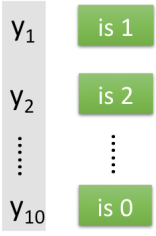


16 x 16 = 256

Ink → 1

No ink → 0

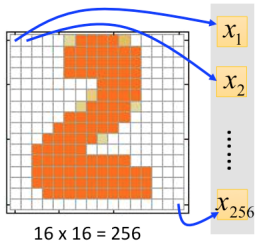
Output



Each dimension represents the confidence of a digit.

Exemple : reconnaissance d'écriture

Input

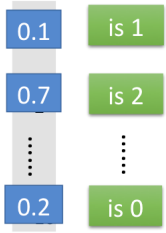


16 x 16 = 256

Ink → 1

No ink → 0

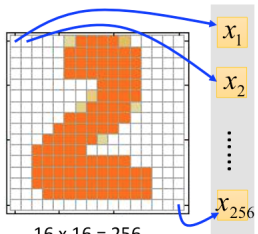
Output



Each dimension represents the confidence of a digit.

Exemple : reconnaissance d'écriture

Input



16 x 16 = 256

Ink → 1
No ink → 0

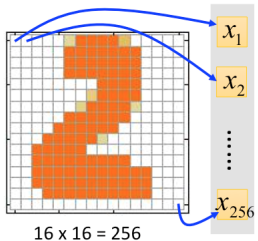
Output



Each dimension represents the confidence of a digit.

Exemple : reconnaissance d'écriture

Input

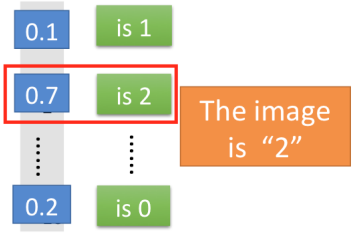


16 x 16 = 256

Ink → 1

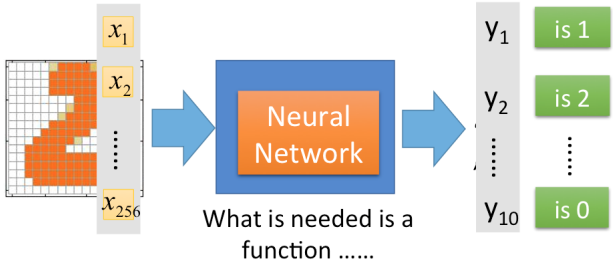
No ink → 0

Output



Each dimension represents the confidence of a digit.

Exemple : reconnaissance d'écriture



Questions/Réponses

- Combien de couches mettre dans le réseaux ?

Questions/Réponses

- Combien de couches mettre dans le réseaux ?
- Peut-on concevoir la structure du réseau ?

Questions/Réponses

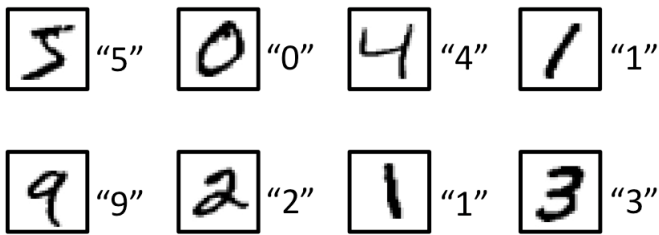
- Combien de couches mettre dans le réseaux ?
- Essayer, apprendre des erreurs et avoir de l'intuition ...
- Peut-on concevoir la structure du réseau ?

Questions/Réponses

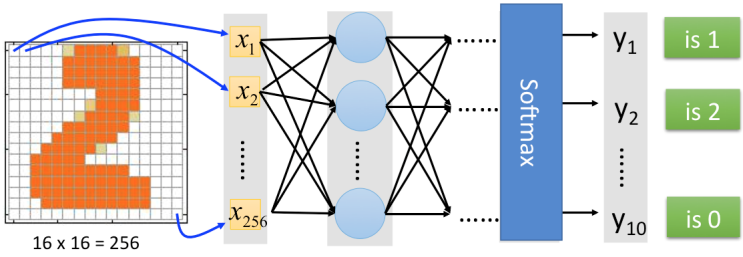
- Combien de couches mettre dans le réseaux ?
- Essayer, apprendre des erreurs et avoir de l'intuition ...
- Peut-on concevoir la structure du réseau ?
- Réseaux de neurones à convolution ...

Données d'entrainement

Les Images et leurs étiquettes :



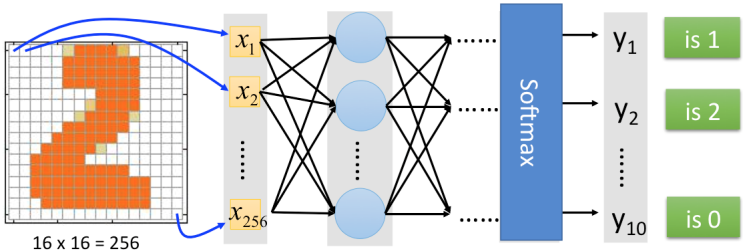
Objectif de l'entrainement



Ink → 1

No ink → 0

Objectif de l'entrainement



Ink → 1
No ink → 0

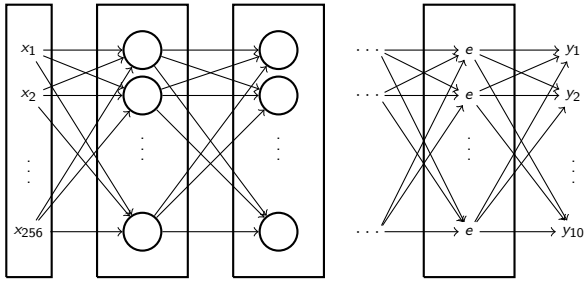
Input: 

$\Rightarrow y_1$ a la plus grande valeur,

Input: 

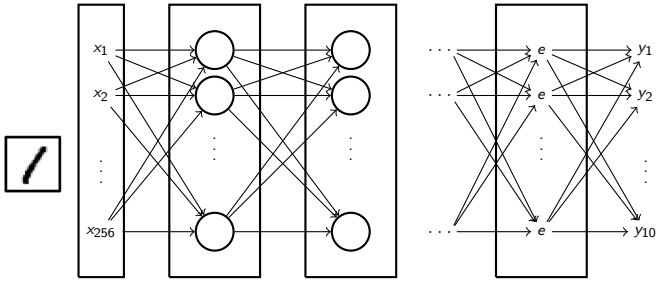
$\Rightarrow y_2$ a la plus grande valeur,

Fonction de perte (Loss Function)



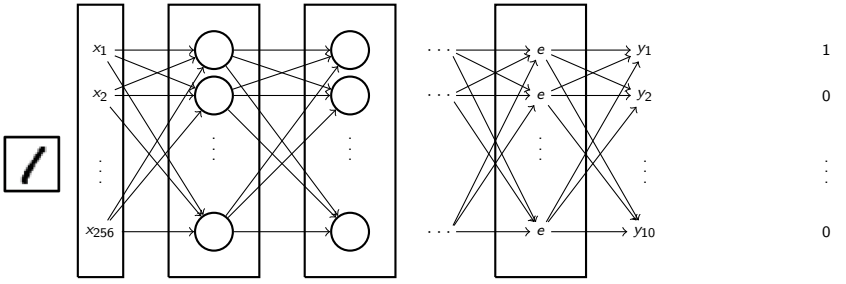
Couche de Softmax

Fonction de perte (Loss Function)

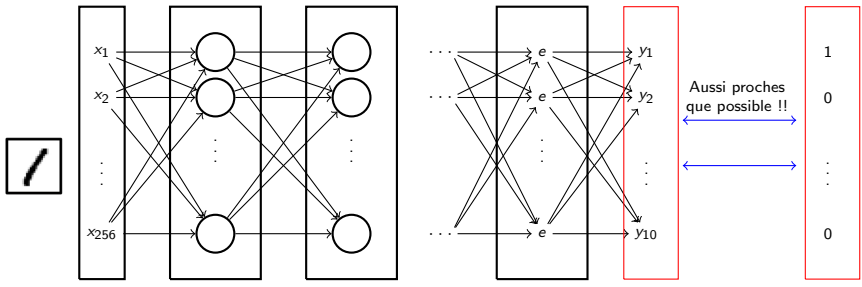


Couche de Softmax

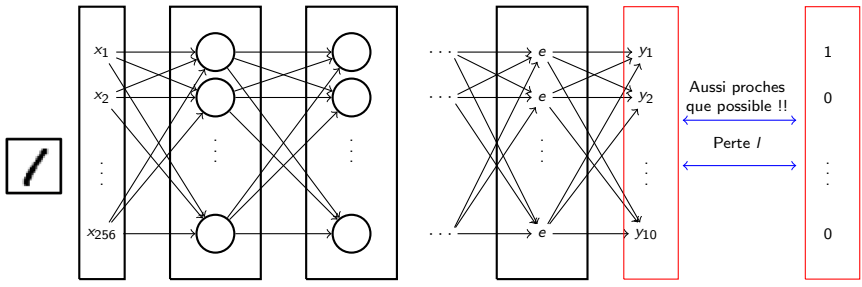
Fonction de perte (Loss Function)



Fonction de perte (Loss Function)



Fonction de perte (Loss Function)



Fonction de perte (Loss Function)

- La fonction de perte peut être :
  - La somme des carrés des écarts :

$$I = \sum_{i=1}^{10} (y_i - \hat{y}_i)^2 .$$

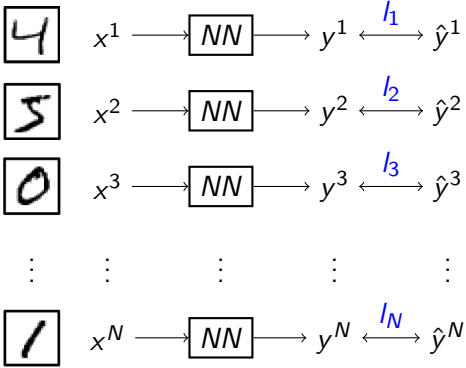
  - La *cross entropy* :

$$I = -\sum_{i=1}^{10} y_i \ln(\hat{y}_i).$$

Une bonne fonction (définie par les paramètres que l'on met sur le réseau) doit minimiser la *valeur totale* de la fonction de perte ...
Question : quelle est l'"intuition" derrière la définition de la cross-entropy ?

Perte totale

Pour tout l'ensemble de traitement :



Perte totale :

$$L = \sum_{i=1}^{N} l_i.$$

Objectif :

Minimiser cette fonction.

Outline

Introduction

D'abords il y a le neurone

... puis les réseaux de neurones

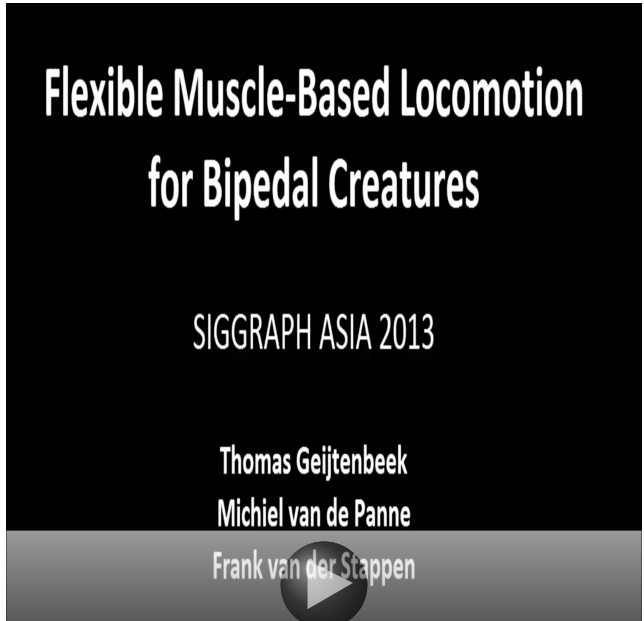
et c'est quoi alors le deep ?

et l'apprentissage dans tout ça ?

Apprentissage ?



Apprentissage ?



Apprentissage ?

Comment choisir la meilleure fonction de classification ?
Trouver les meilleurs paramètres θ qui minimisent la perte totale L.

- Solution 1. : Enumérer toutes les possibilités $\rightarrow$ infaisable ! !

Apprentissage ?

Comment choisir la meilleure fonction de classification ?
Trouver les meilleurs paramètres θ qui minimisent la perte totale L.

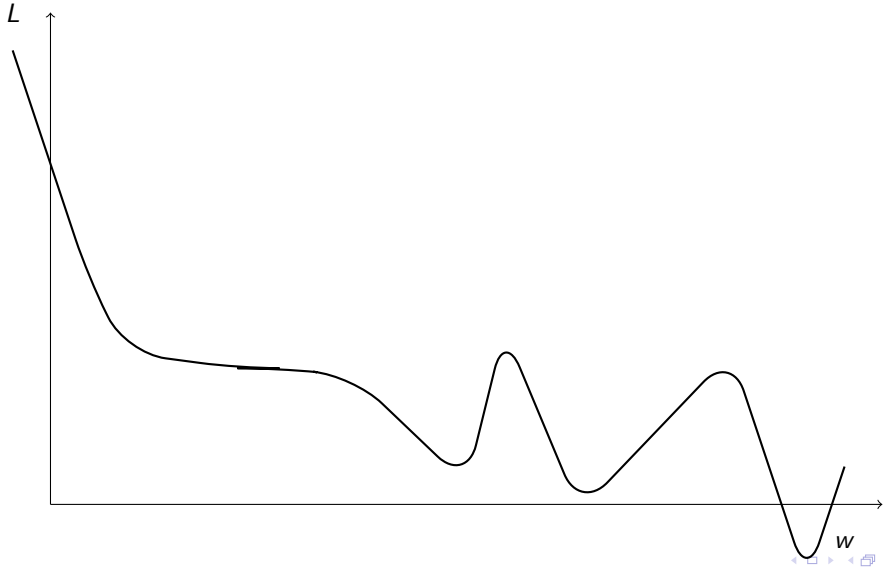
- Solution 1. : Enumérer toutes les possibilités $\rightarrow$ infaisable ! !
- Solution 2. : Descente du Gradient.

Descente du gradiant

Objectif : déterminer les paramètres θ du réseau qui minimisent la perte total L.

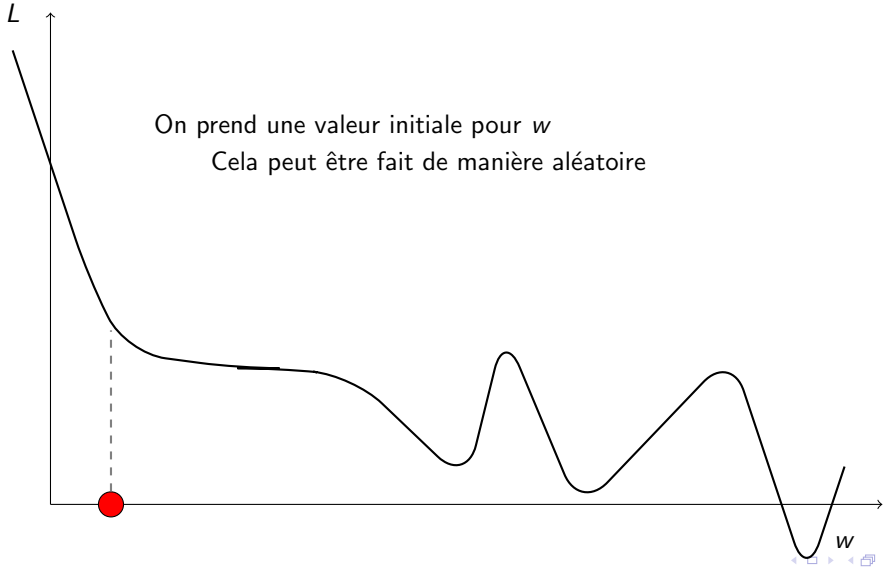
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



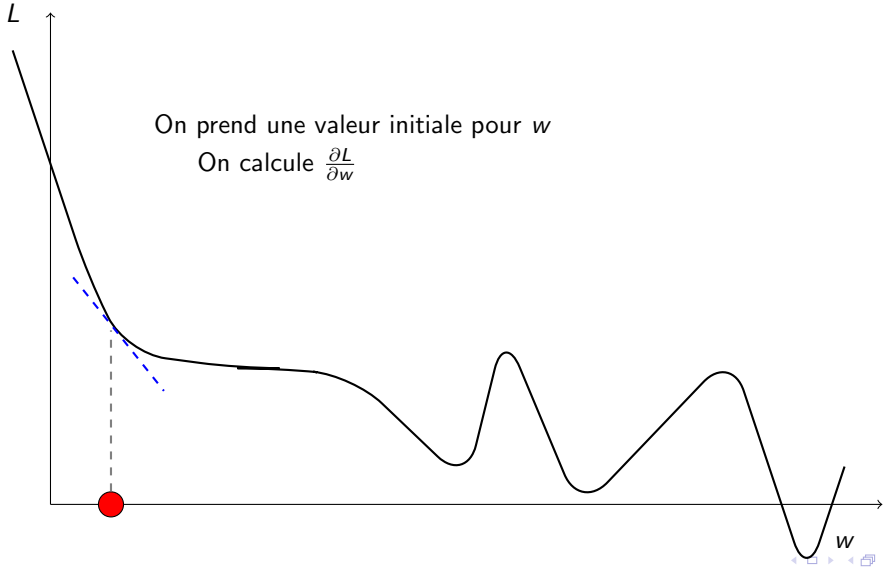
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



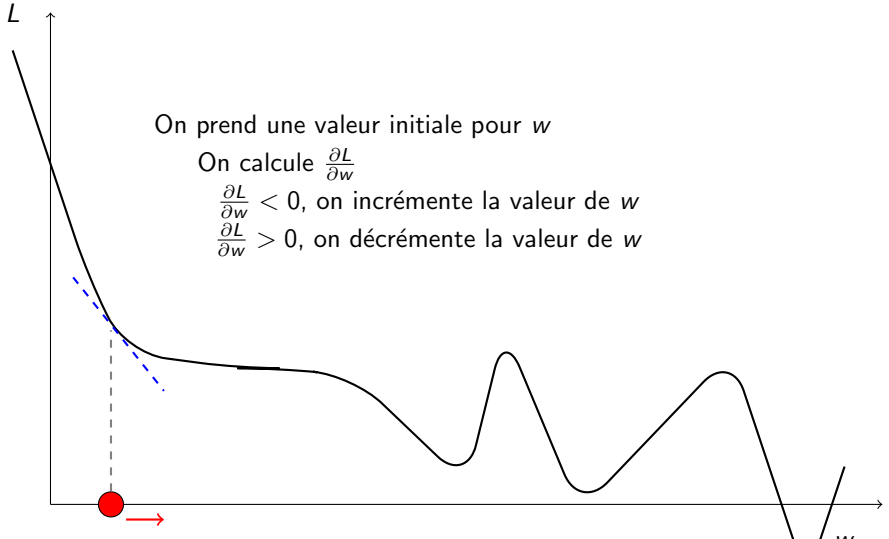
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



On prend une valeur initiale pour w

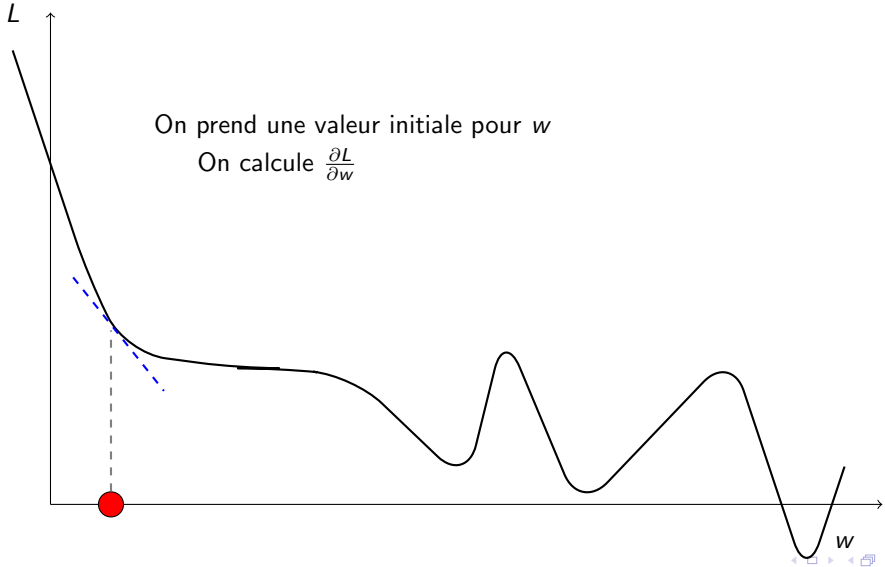
On calcule $\frac{\partial L}{\partial w}$

$\frac{\partial L}{\partial w} < 0$, on incrémente la valeur de w

$\frac{\partial L}{\partial w} > 0$, on décrémente la valeur de w

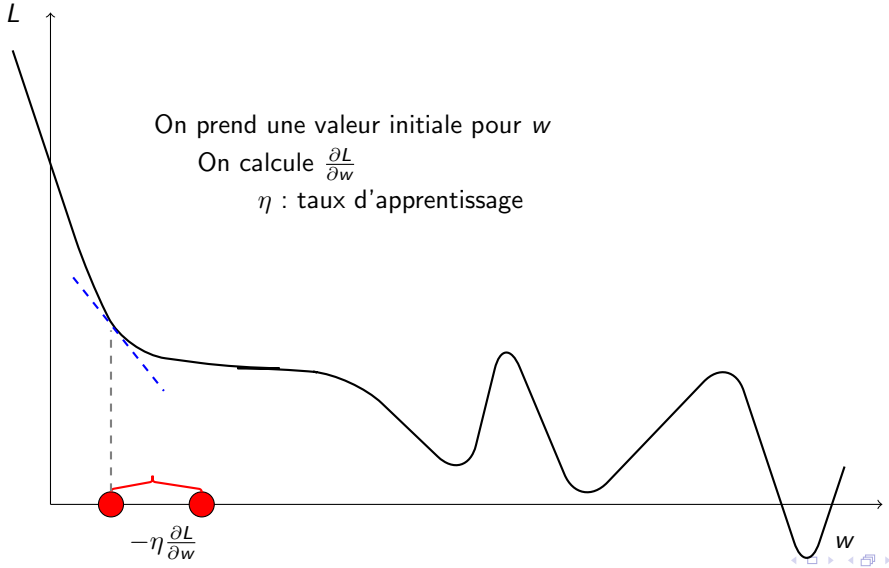
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



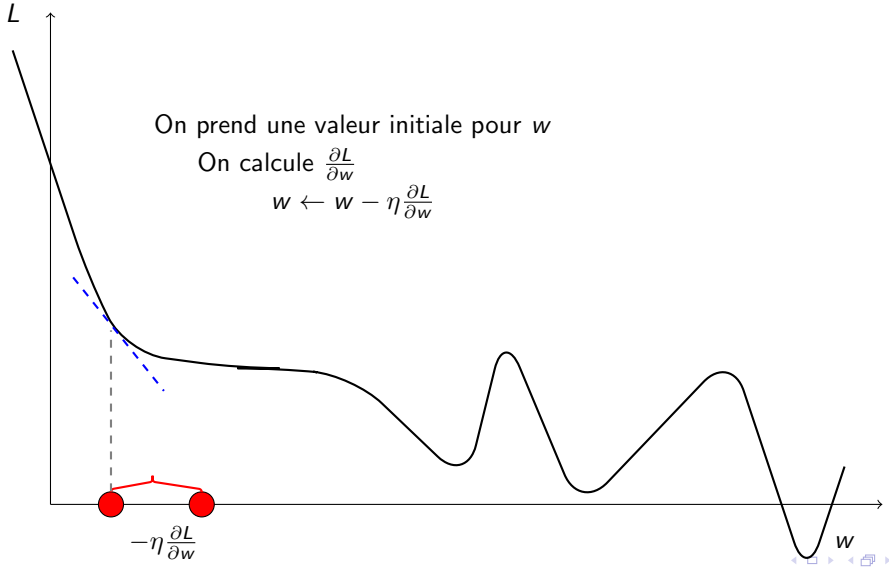
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



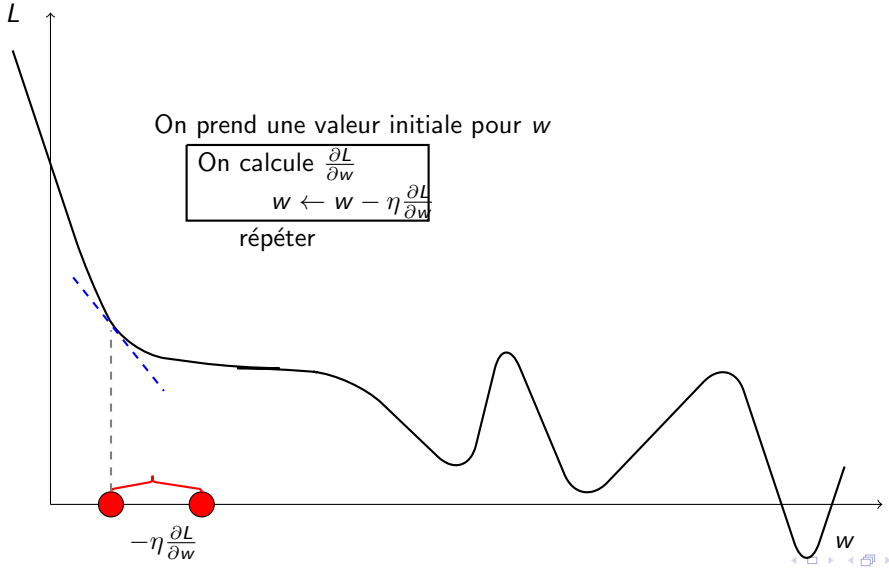
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



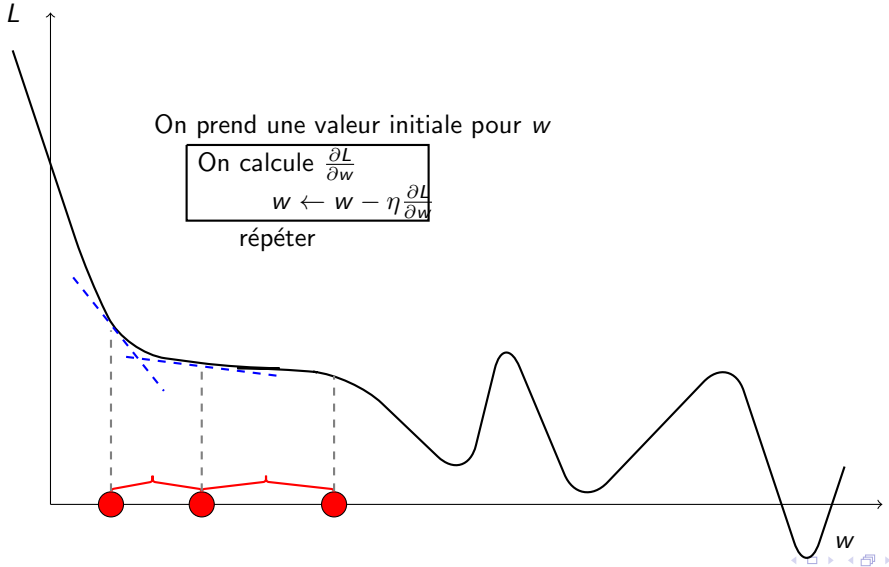
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



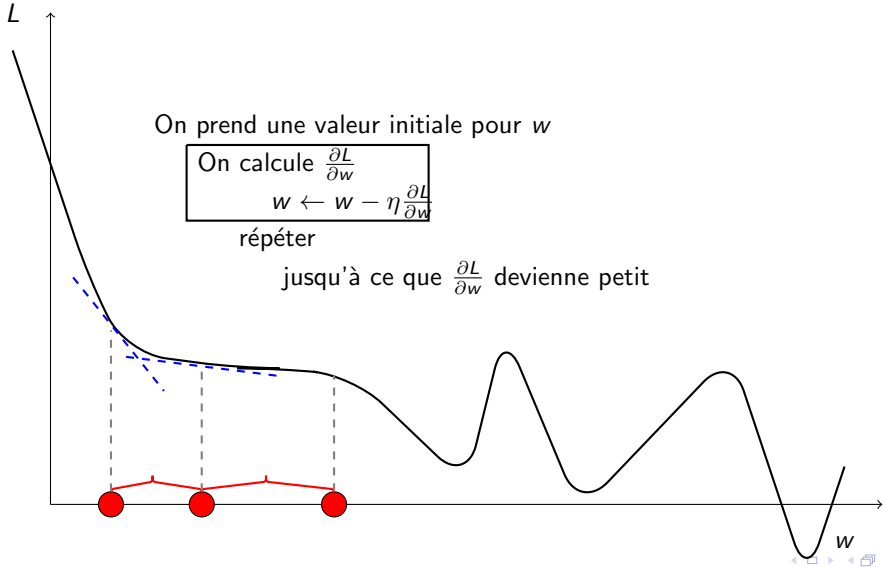
Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.



Descente du gradient

Trouver les paramètres θ du réseau qui minimisent la fonction de perte totale L.

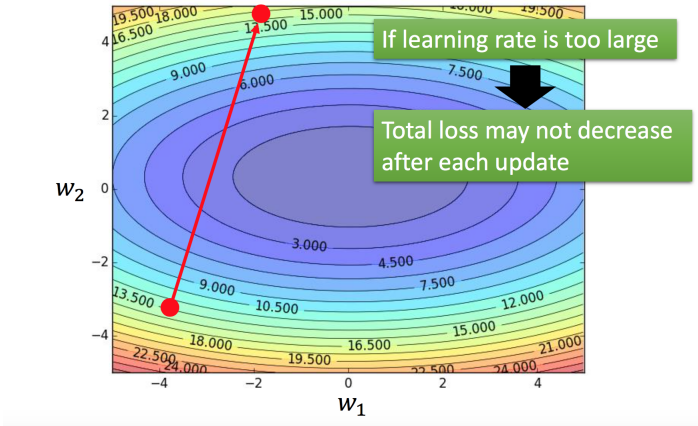


Descente du gradient

de l'importance du choix du taux d'apprentissage η :

Descente du gradient

de l'importance du choix du taux d'apprentissage η :

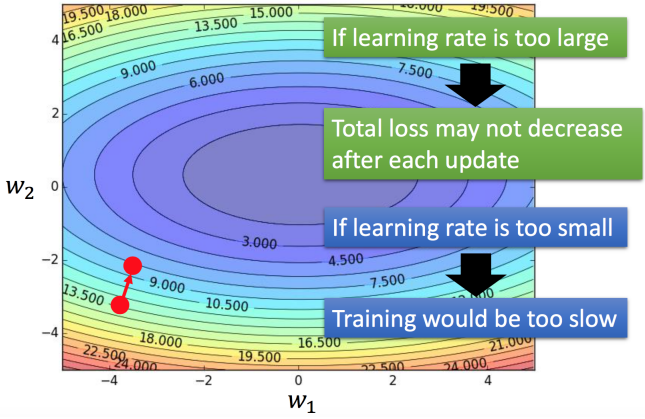


Descente du gradient

de l'importance du choix du taux d'apprentissage η :

Descente du gradient

de l'importance du choix du taux d'apprentissage η :



Descente du gradient

de l'importance du choix du taux d'apprentissage η :

- Un taux trop petit garantit une convergence vers un minimum mais le temps de convergence peut être trop grand ....
- Un taux trop grand peut faire "sauter" le minimum ..

Descente du gradient

de l'importance du choix du taux d'apprentissage η :

Descente du gradient

de l'importance du choix du taux d'apprentissage η :
Une solution simple : réduire le taux par un facteur toutes les K itérations

- Initialement, on est loin de la destination, on utilise donc un pas assez grand,
- Après quelques itérations, on est proche de la destination, on réduit donc le pas.
- En général, on utilise la mise à jour suivante :

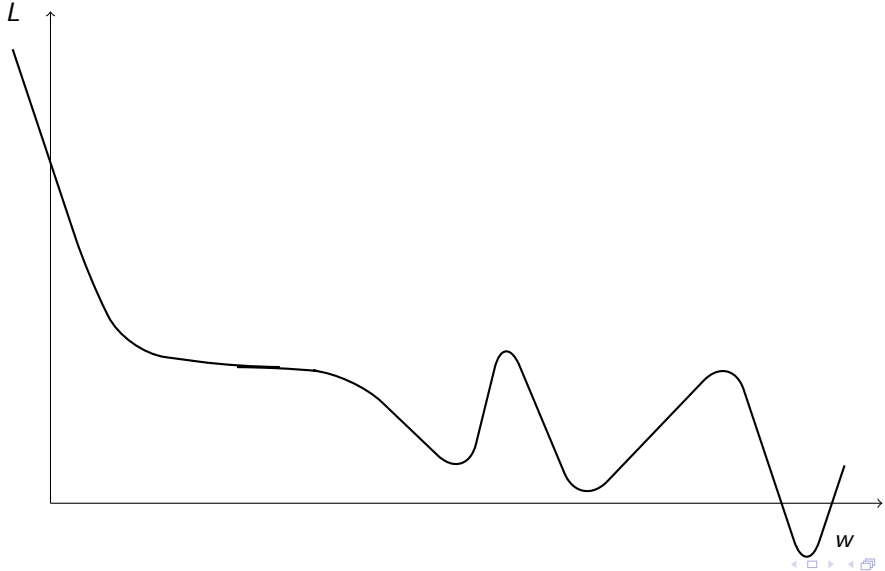
$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

Descente du gradient

- La descente du gradient ne garantie pas de trouver un minimum global,
- Néanmoins, c'est LA méthode d'apprentissage du deep ...

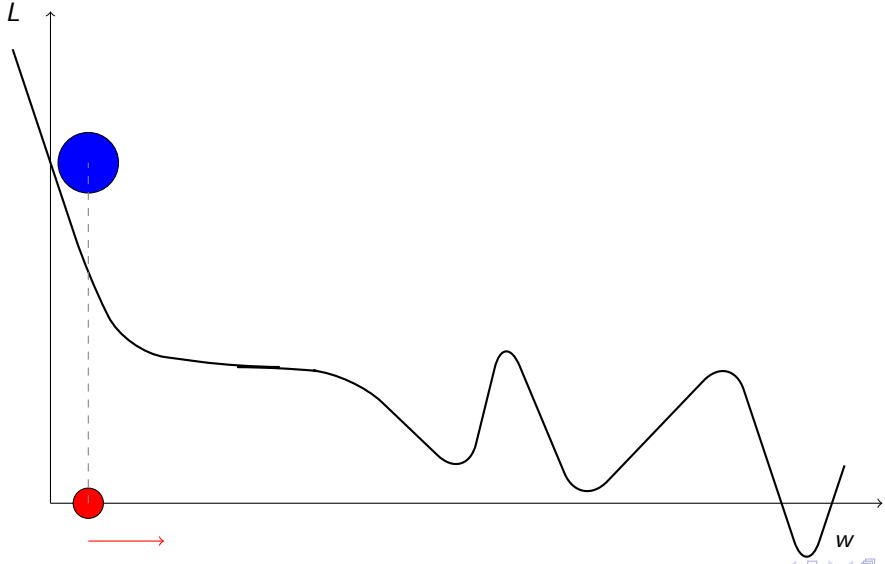
Descente du gradient

Problème : minimum mais local ....



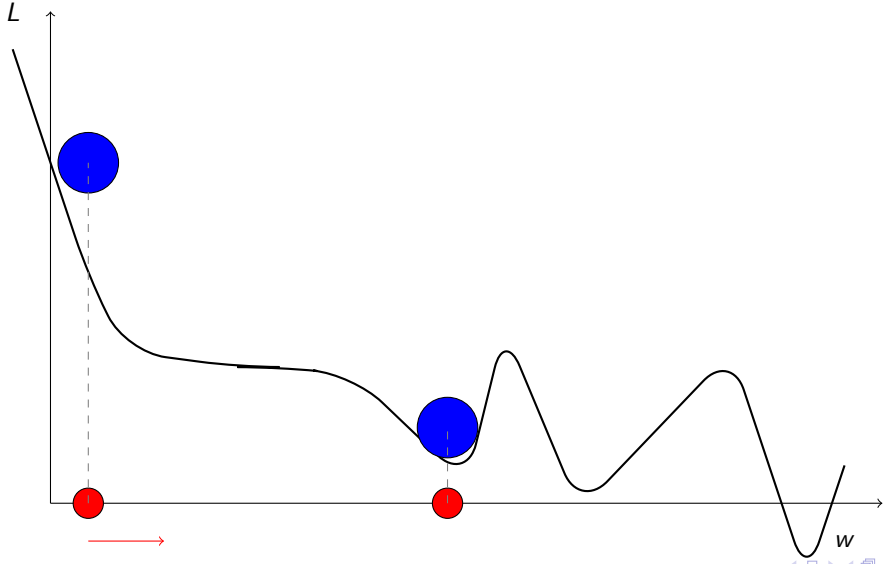
Descente du gradient

Problème : minimum mais local ...



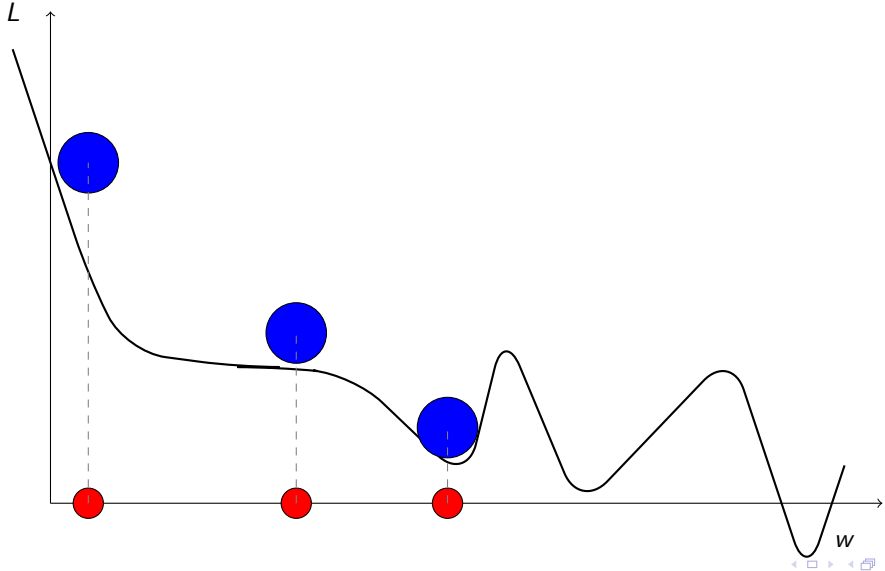
Descente du gradient

Problème : minimum mais local ....



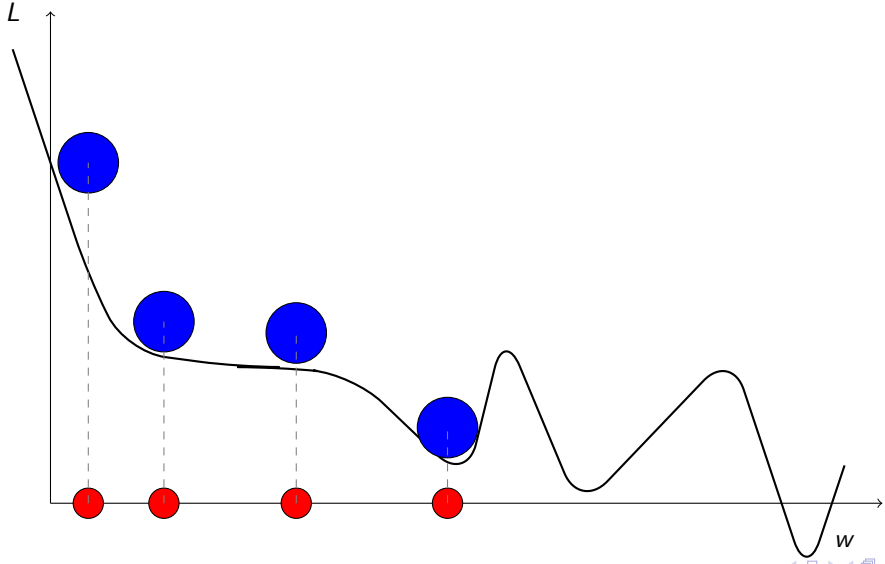
Descente du gradient

Problème : minimum mais local ....



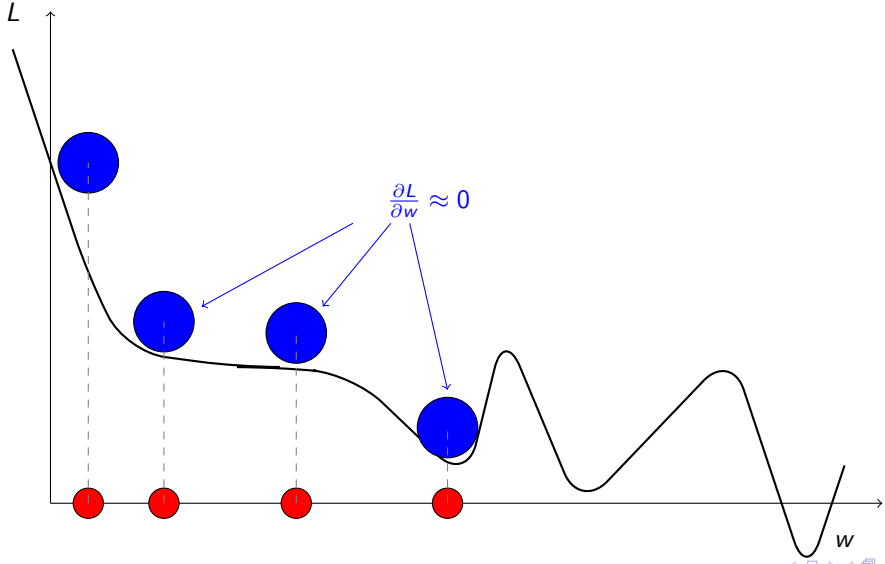
Descente du gradient

Problème : minimum mais local ....



Descente du gradient

Problème : minimum mais local ....

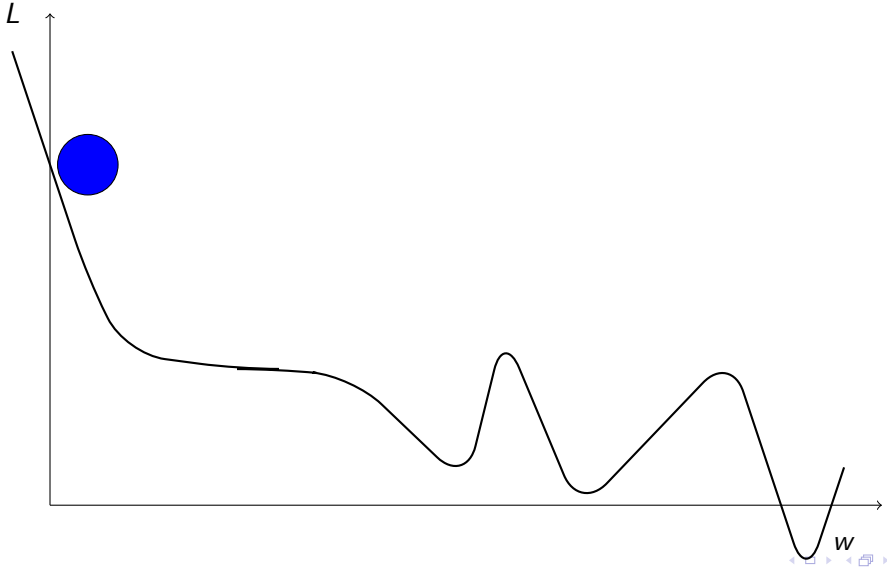


Descente du gradient

Problème : minimum mais local ....
Idée : s'inspirer de la physique et d'une balle qui dévale une pente

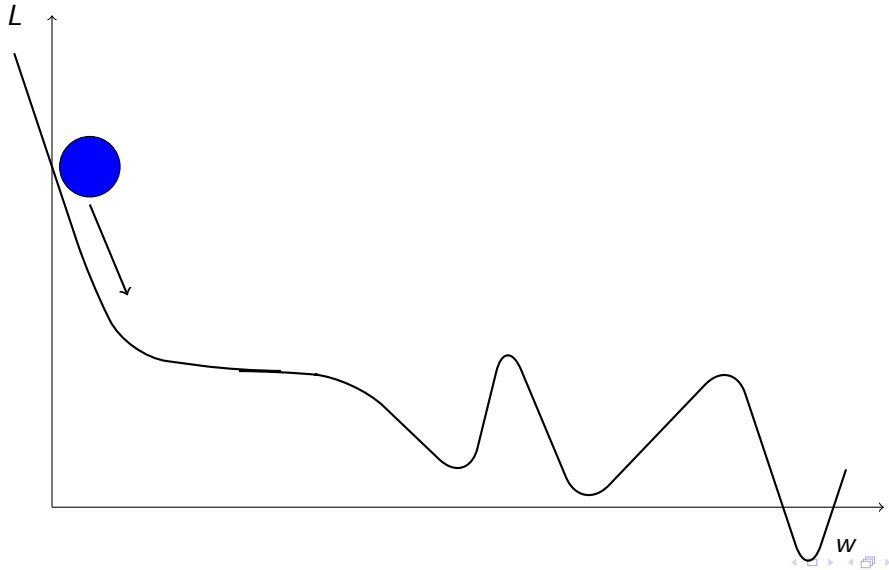
Descente du gradient

Méthode du Momentum :



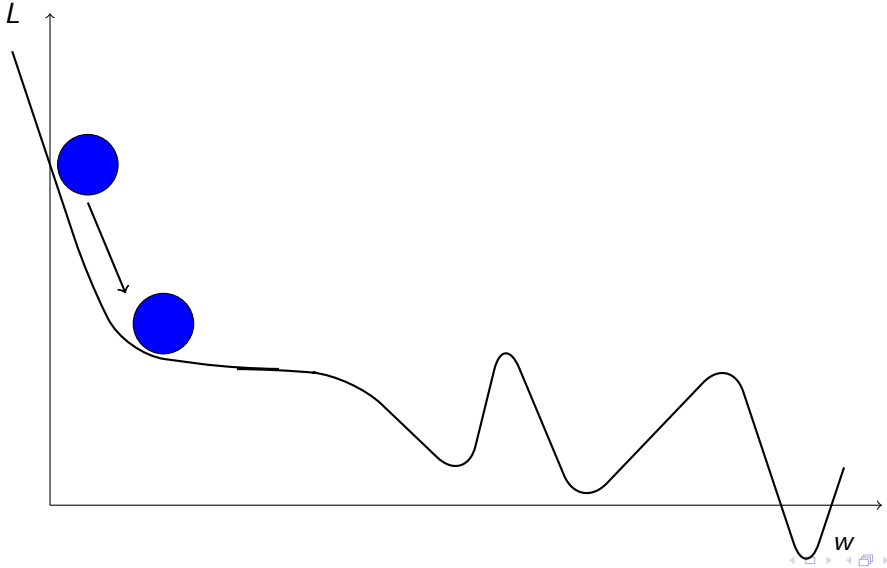
Descente du gradient

Méthode du Momentum :



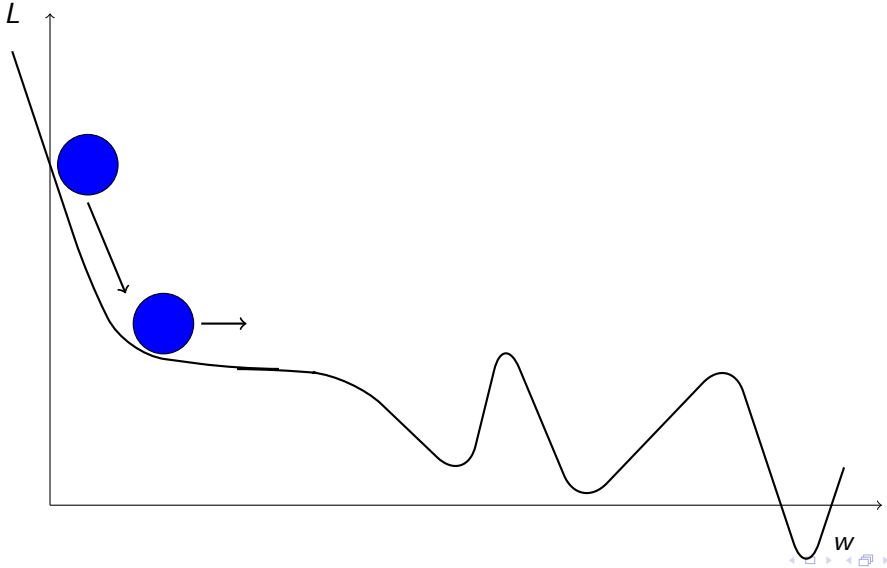
Descente du gradient

Méthode du Momentum :



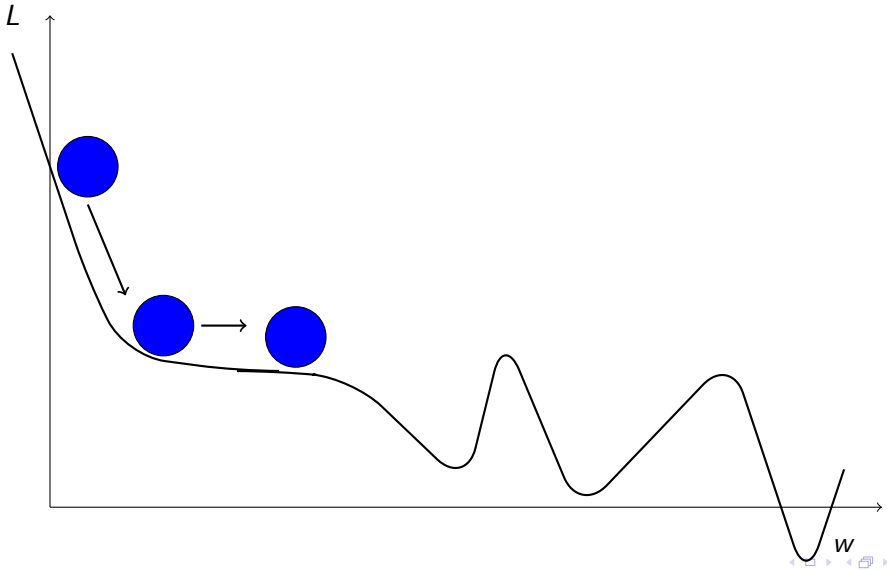
Descente du gradient

Méthode du Momentum :



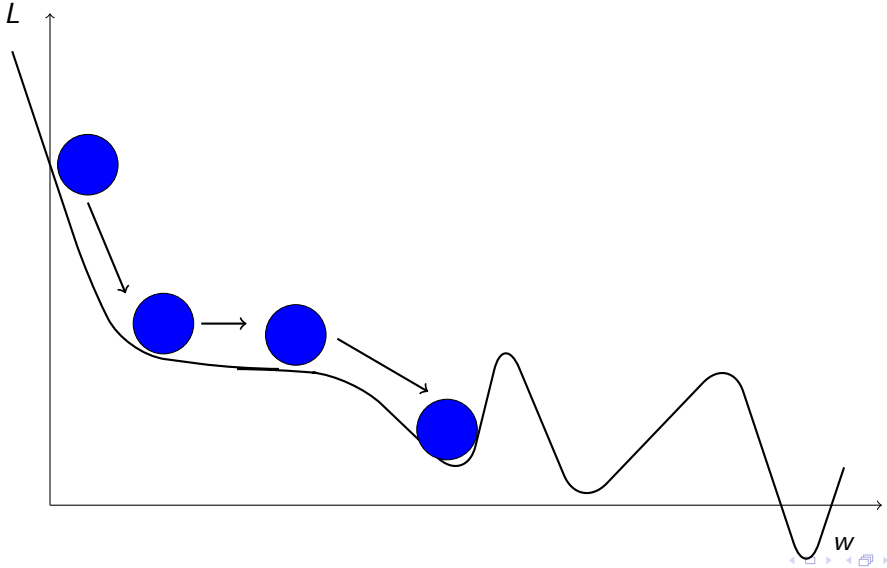
Descente du gradient

Méthode du Momentum :



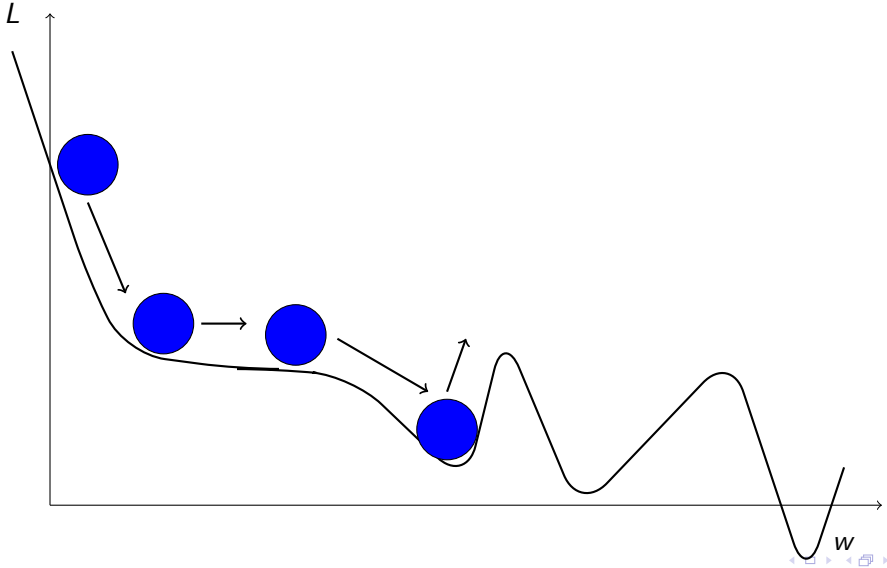
Descente du gradient

Méthode du Momentum :



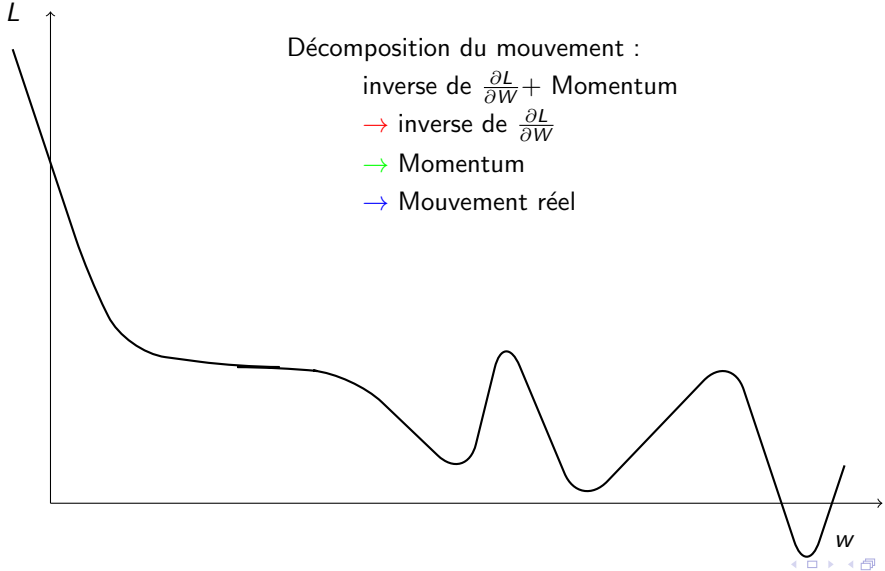
Descente du gradient

Méthode du Momentum :



Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

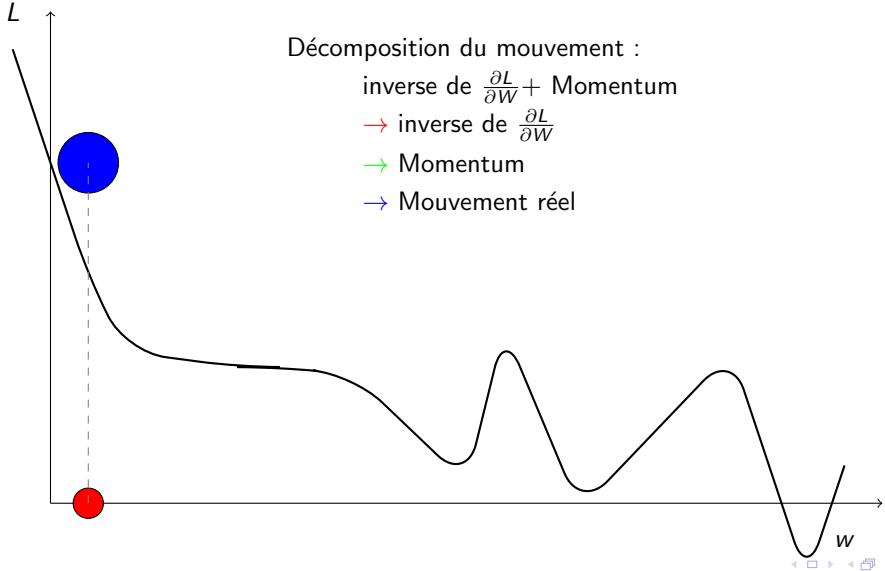
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

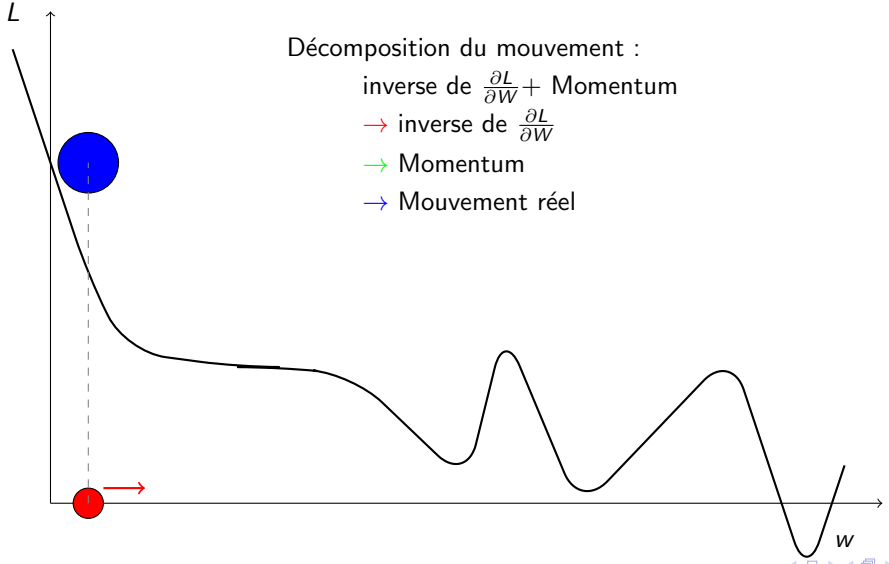
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

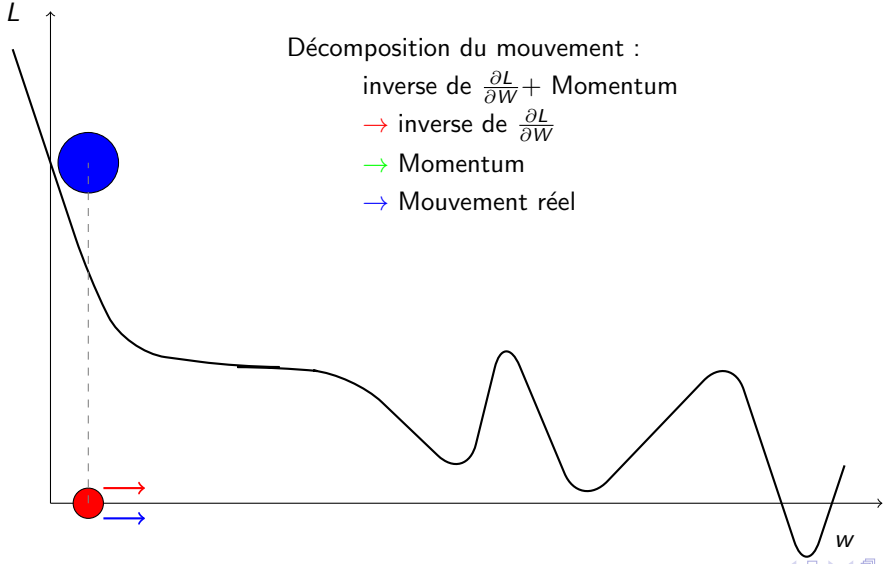
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

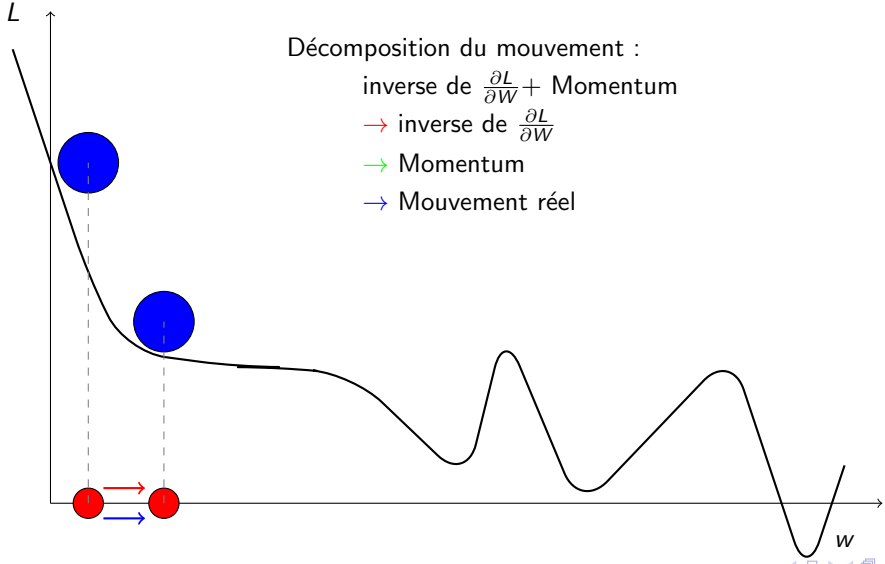
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

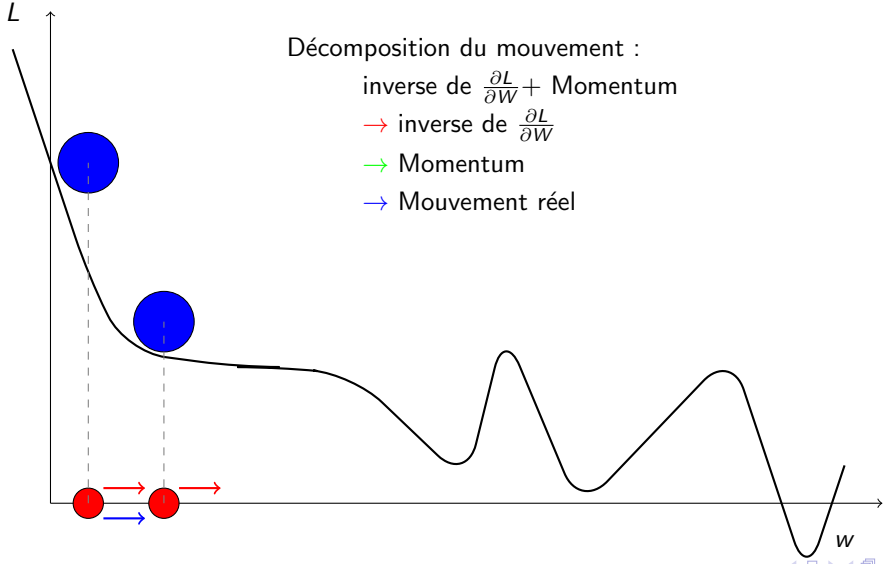
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

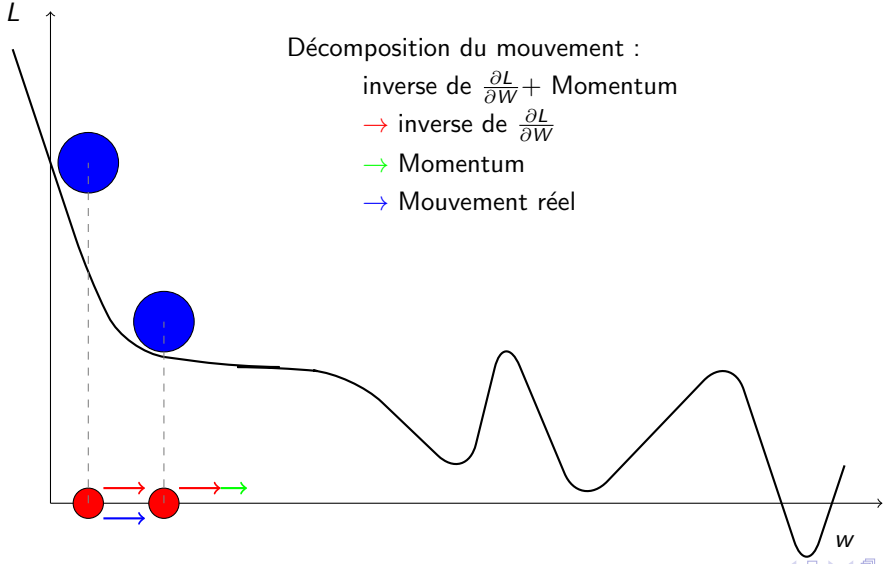
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

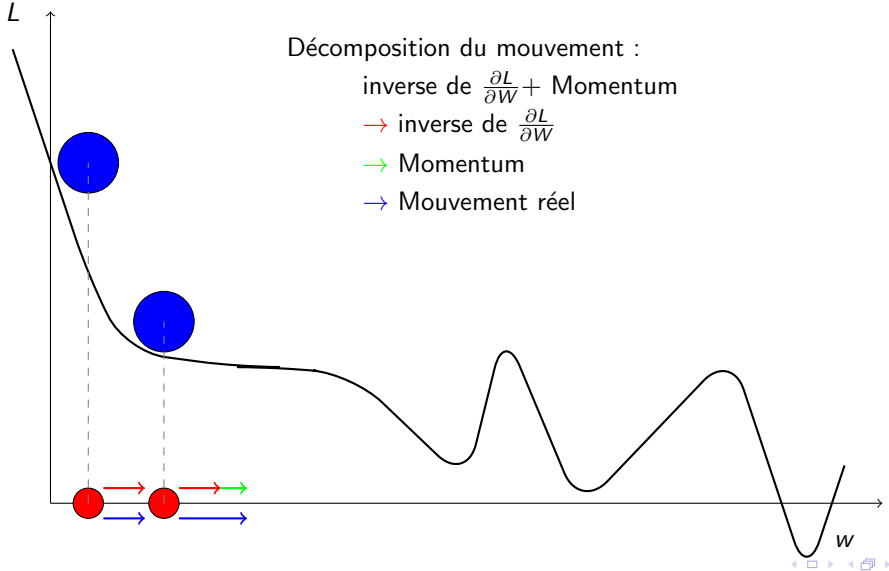
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

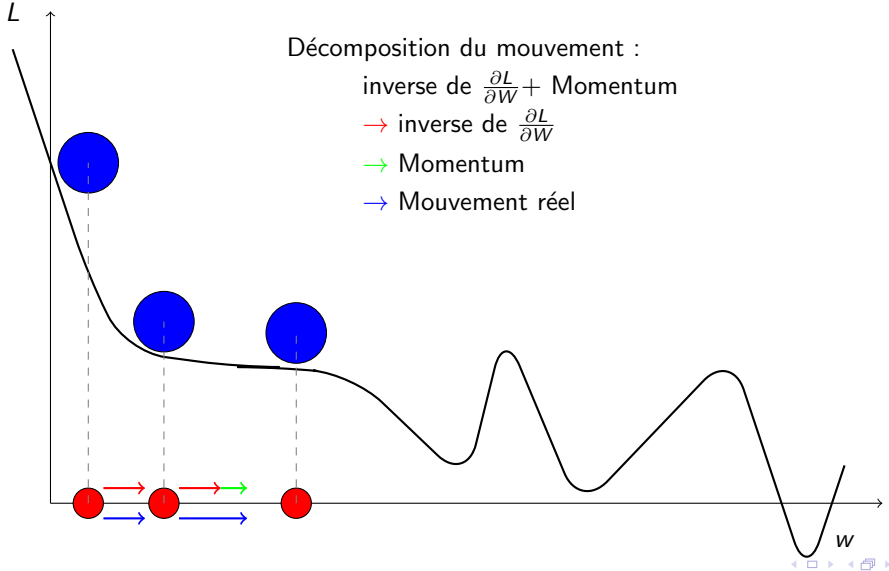
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

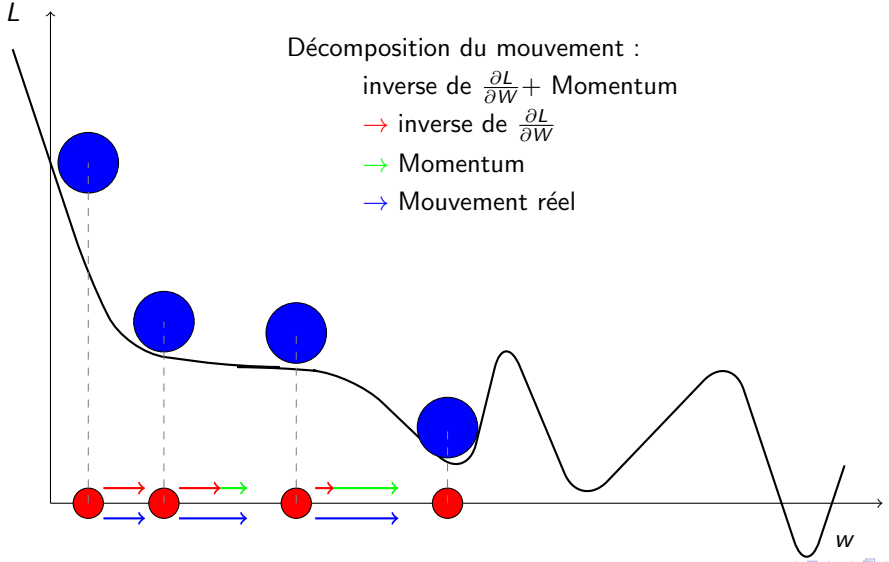
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

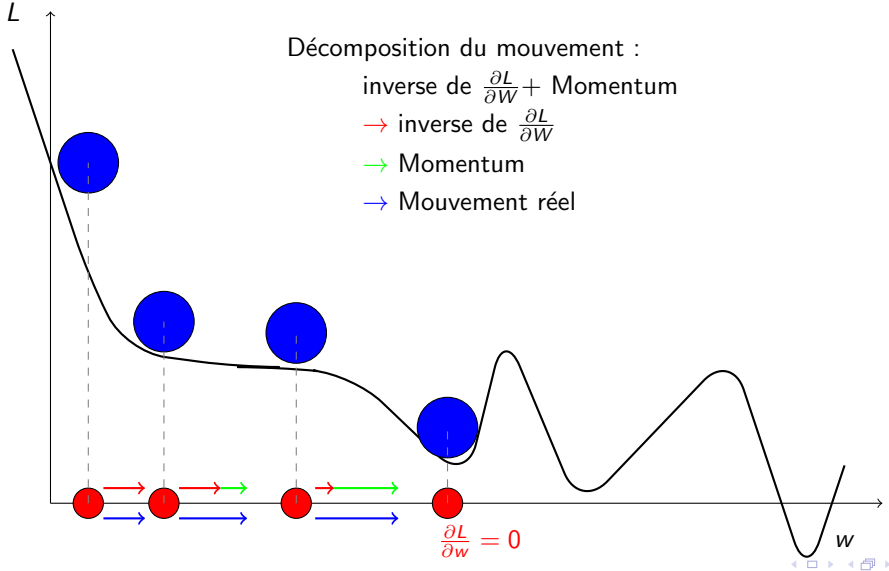
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

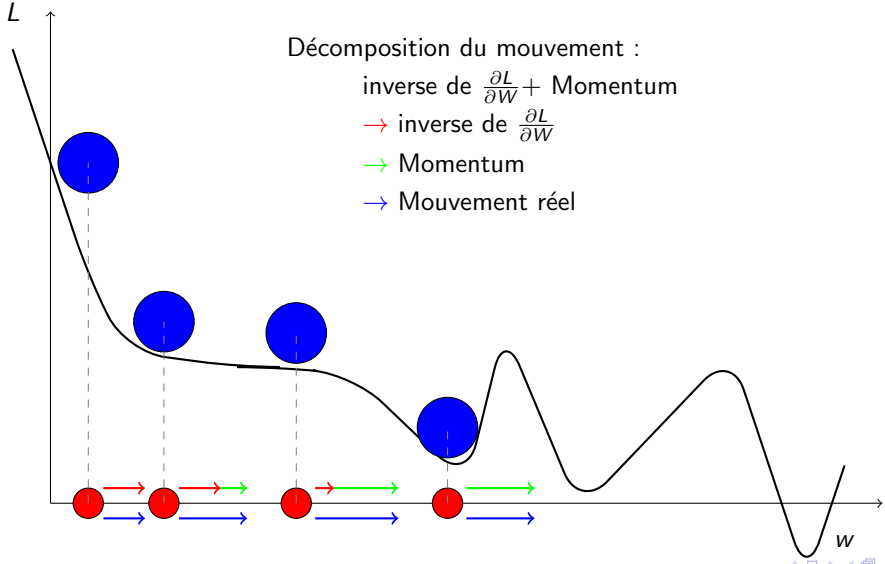
Descente du gradient

Problème : minimum mais local ....



Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

inverse de $\frac{\partial L}{\partial W}$ + Momentum

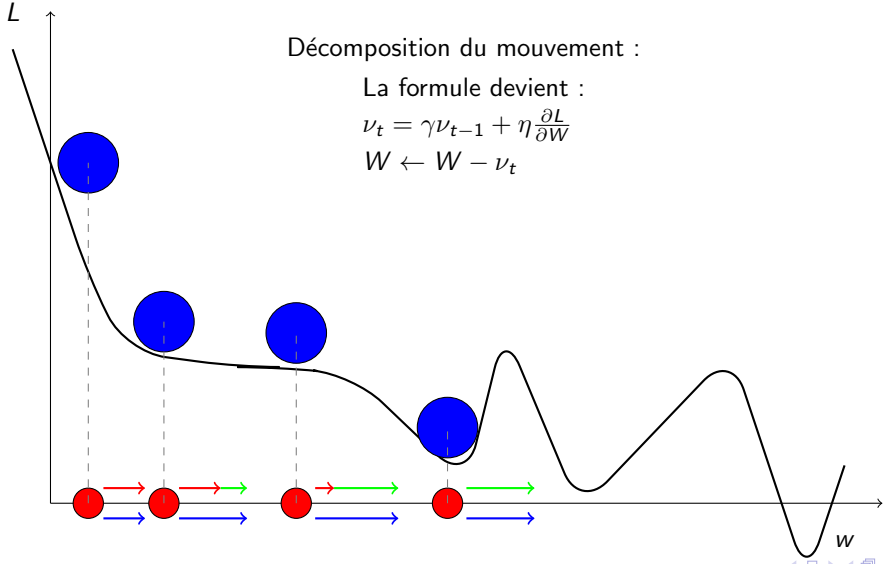
$\rightarrow$ inverse de $\frac{\partial L}{\partial W}$

$\rightarrow$ Momentum

$\rightarrow$ Mouvement réel

Descente du gradient

Problème : minimum mais local ....



Décomposition du mouvement :

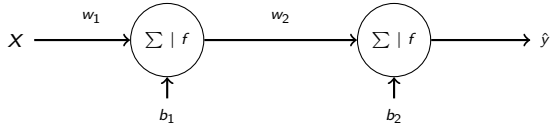
La formule devient :

$\nu_t = \gamma \nu_{t-1} + \eta \frac{\partial L}{\partial W}$

$W \leftarrow W - \nu_t$

Algorithme de Back-propagartion

Considérons un réseau très simple :



Ecrivons les fonctions mathématiques de chaque couche :

$$z_1 = w_1 x + b_1 \quad (1)$$
$$y_1 = f(z_1) \quad (2)$$
$$z_2 = w_2 y_1 + b_2 \quad (3)$$
$$\hat{y} = f(z_2). \quad (4)$$

Algorithme de Back-propagartion

Pour simplifier, on considère que la fonction de perte (Loss function) est la somme des carrés des écarts :

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2 .$$

Et que la fonction d'activation (dans les deux couches) est simplement la sigmoïd :

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Rappelons le but de l'apprentissage :
Si on note $\theta = \{w_1, w_2, b_1, b_2\}$ trouver θ^* tel que :

$$\theta^* = \arg\min_{\theta} L(\theta).$$

Algorithme de Back-propagartion

Et rappelons la formule de la méthode du descente de gradient :

$$\theta^{t+1} = \theta^t - \eta \nabla L\left(\theta^t\right)$$

Ce que l'on peut traduire dans notre cas par :

$$w_1^{(t+1)} = w_1^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial w_1}\left(w_1^{(t)}, w_2^{(t)}, b_1^{(t)}, b_2^{(t)}\right)$$
$$w_2^{(t+1)} = w_2^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial w_2}\left(w_1^{(t)}, w_2^{(t)}, b_1^{(t)}, b_2^{(t)}\right)$$
$$b_1^{(t+1)} = b_1^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial b_1}\left(w_1^{(t)}, w_2^{(t)}, b_1^{(t)}, b_2^{(t)}\right)$$
$$b_2^{(t+1)} = b_2^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial b_2}\left(w_1^{(t)}, w_2^{(t)}, b_1^{(t)}, b_2^{(t)}\right)$$

Algorithme de Back-propagartion

LA question :

Comment calculer les dérivées partielles

$$\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \frac{\partial \mathcal{L}}{\partial b_1} \text{ et } \frac{\partial \mathcal{L}}{\partial b_2} ?$$

Algorithme de Back-propagartion

Commençons par $w = w_2$.
On peut observer que :

$$\begin{aligned} \mathcal{L}(w) &= \mathcal{L}\left(\hat{y}(w)\right) \\ &= \mathcal{L}\left(\hat{y}(z_2(w))\right) \end{aligned}$$

La règle de chaînage nous indique que :

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial z_2} . \frac{\partial z_2}{\partial w} .$$

Algorithme de Back-propagartion

Faisons les calculs :

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{y}} &= \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(y-\hat{y})^2\right) = (y-\hat{y}) \\
\frac{\partial \hat{y}}{\partial z_2} &= \frac{\partial}{\partial z_2}\left(f(z_2)\right) = \frac{\partial}{\partial z_2}\left(\sigma(z_2)\right) = \sigma(z_2)\left(1-\sigma(z_2)\right) \\
&= \hat{y}(1-\hat{y}). \\
\frac{\partial z_2}{\partial w} &= \frac{\partial}{\partial w}\left(w_2 y_1 + b_2\right) = y_1.
\end{aligned}$$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial w_2} = (y-\hat{y})\,\hat{y}(1-\hat{y})y_1$$

Un calcul semblable (exercice) donne :

$$\frac{\partial \mathcal{L}}{\partial b_2} = (y-\hat{y})\,\hat{y}(1-\hat{y})$$

Algorithme de Back-propagartion

Remarques.
- Observer que la quantité $y - \hat{y}$ n'est autre que l'erreur commise.
- On voit donc que l'on dispose de tous les ingrédients pour calculer les deux dérivées nécessaires pour la dernière couche.

Qu'en est-il de la couche cachée ?

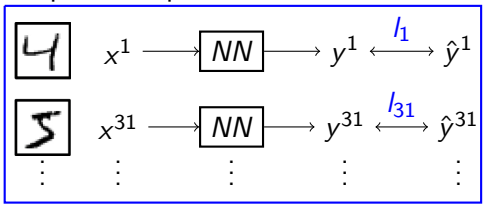
Algorithme de Back-propagartion

En fait tous calculs faits (exercice), on obtient :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_0} &= (\hat{y} - y)\hat{y}(1-\hat{y})y_1(1-y_1)w_1 x \\ \frac{\partial \mathcal{L}}{\partial b_1} &= (\hat{y} - y)\hat{y}(1-\hat{y})y_1(1-y_1)w_1. \end{aligned}$$

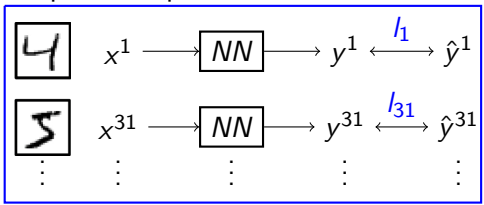
A la recherche de la meilleure fonction

On prend un premier mini-batch :



A la recherche de la meilleure fonction

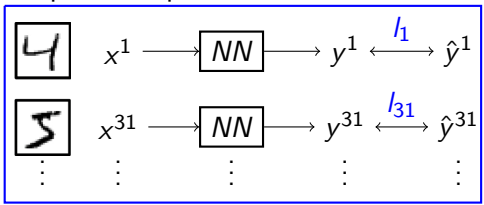
On prend un premier mini-batch :



$$L' = l^1 + l^{31} + \cdots$$

A la recherche de la meilleure fonction

On prend un premier mini-batch :

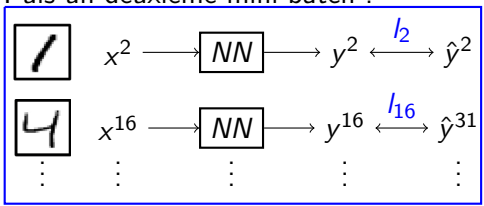


$L' = l^1 + l^{31} + \cdots$

Mettre à jour les paramètres

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :



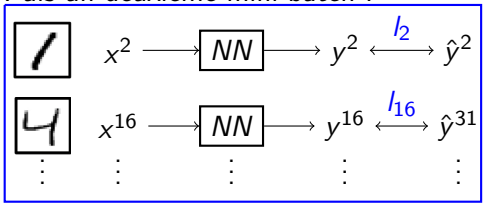
Et ainsi de suite

...

Jusqu'à ce que tous les mini-batchs soient lus.
$\Rightarrow$ Cele correspond à un(e) epoch.
Et l'on répète le processus un nombre d'epochs ...

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :



$$L'' = l^2 + l^{16} + \dots$$

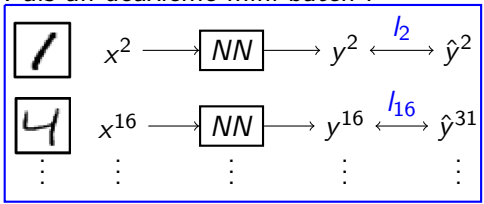
Et ainsi de suite

$$\dots$$

Jusqu'à ce que tous les mini-batchs soient lus.
$\Rightarrow$ Cele correspond à un(e) epoch.
Et l'on répète le processus un nombre d'epochs ...

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :

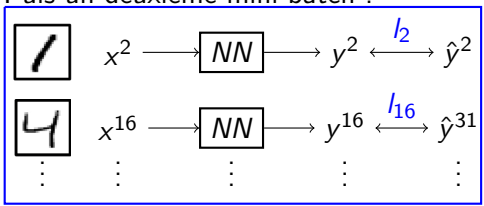


$L'' = l^2 + l^{16} + \dots$

Mettre à jour les paramètres

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :



$L'' = l^2 + l^{16} + \dots$

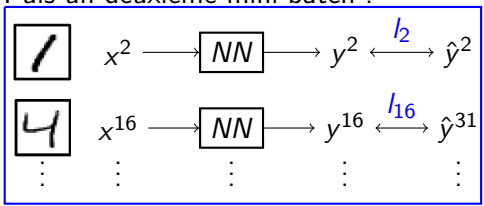
Mettre à jour les paramètres

Et ainsi de suite

...

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :



$L'' = l^2 + l^{16} + \dots$

Mettre à jour les paramètres

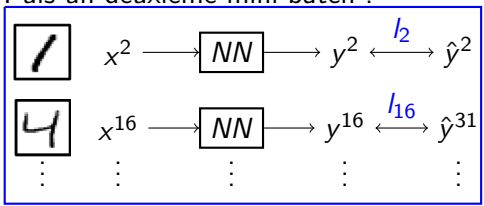
Et ainsi de suite

...

Jusqu'à ce que tous les mini-batchs soient lus.

A la recherche de la meilleure fonction

Puis un deuxième mini-batch :



$L'' = l^2 + l^{16} + \dots$

Mettre à jour les paramètres

Et ainsi de suite

...

Jusqu'à ce que tous les mini-batchs soient lus.

$\Rightarrow$ Cele correspond à un(e) epoch.

Et l'on répète le processus un nombre d'epochs ...

A la recherche de la meilleure fonction

Dans la majorité des outils de deep learning, ces deux paramètres sont à indiquer :

- batch_size correspond au nombre d'exemples utilisé pour estimer le gradient de la fonction de coût.
- epochs est le nombre d'époques (i.e. passages sur l'ensemble des exemples de la base d ?apprentissage) lors de la descente de gradient.

Problème de la disparition du gradient

Intuition :
La mise à jour des paramètres se fait si la dérivée n'est pas nulle.
Problème : avec la sigmoid (et d'autres fonctions) comme fonction d'activation, au bout d'un certain nombre d'itérations, $\frac{\partial L}{\partial W}$ devient nulle ...
Solution :
Utiliser, comme fonction d'activation, la fonction ReLU (Rectified Linear Unit) :

