

# Les réseaux de neurones

Pascal Poncelet  
LIRMM  
Pascal.Poncelet@lirmm.fr  
<http://www.lirmm.fr/~poncelet>

---



---



---



---



---



---



---



---

## Qu'est ce que c'est qu'un neurone ?

WIKIPÉDIA L'encyclopédie libre

Accueil Portails thématiques Article au hasard Discuter Contribuer Discuter sur Wikipedia Aide Communauté Mises à jour récentes Faire un don Quitter

Wiki Loves Monuments : photographiez un monument historique, aidez Wikipédia et gagnez ! En apprendre plus

Non connecté Discussion Contributions Créez un compte Se connecter

Lire Modifier Modifier le code Voir l'historique Rechercher dans Wikipédia

**Neurone**

Un neurone, ou une **cellule nerveuse**, est une **cellule excitabile** constituant l'**unité fonctionnelle de base du système nerveux**. Les neurones assurent la transmission d'un **signal biologique** appelé **flux nerveux**. Ils ont deux propriétés physiologiques : l'**excitabilité**, c'est-à-dire la capacité de répondre aux stimulations et de convertir celles-ci en impulsions nerveuses, et la **conductivité**, c'est-à-dire la capacité de transmettre les impulsions.

Sommaire [masquer]

- 1 Statistiques
- 2 Structure

2

---



---



---



---



---



---



---



---

## Un neurone

Les dendrites reçoivent l'influx nerveux d'autres neurones. Le neurone évalue alors l'ensemble de la stimulation reçue. Si celle-ci est suffisante, il est excité : il transmet un signal (0/1) le long de l'axone et l'excitation est propagée jusqu'aux autres neurones qui y sont connectés via les synapses.

3

---



---



---



---



---



---



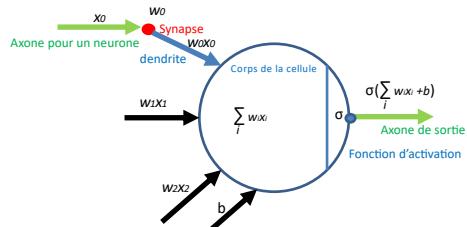
---



---

## Un neurone artificiel

"Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques" (Wikipedia)



4

---

---

---

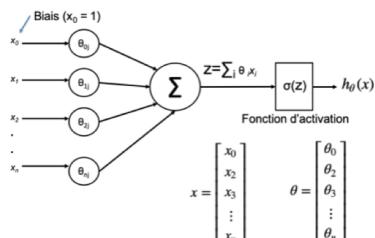
---

---

---

## La régression logistique

- Rappel : utilisation d'une Sigmoid



5

---

---

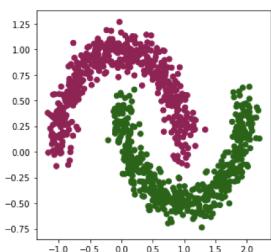
---

---

---

---

## Appliquons la sur ce jeu de données



Quelle est la frontière de prédiction ?

6

---

---

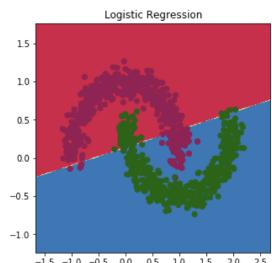
---

---

---

---

## Appliquons la sur ce jeu de données



Une droite

7

---

---

---

---

---

---

---

---

## Les réseaux de neurones

- Les réseaux de neurones se composent des éléments suivants :
  - Une couche d'entrée qui reçoit l'ensemble des caractéristiques (features), i.e. les variables prédictives
  - Un nombre arbitraire de couches cachées
  - Une couche de sortie,  $\hat{y}$ , qui contient la variable à prédire
  - Un ensemble de poids  $W$  qui vont être ajoutés aux valeurs des features et de biais  $b$  entre chaque couche
  - Un choix de fonction d'activation pour chaque couche cachée,  $\sigma$



8

---

---

---

---

---

---

---

---

## Couche de sortie

- Elle doit avoir autant de neurones qu'il y a de sorties au problème de classification :
- régression* : 1 seul neurone (C.f. notebook descente de gradient)
- classification binaire* : 1 seul neurone avec une fonction d'activation qui sépare les deux classes
- classification multi-classe* : 1 neurone par classe et une fonction d'activation Softmax pour avoir la classe appropriée en fonction des probabilités de l'entrée appartenant à chaque classe



9

---

---

---

---

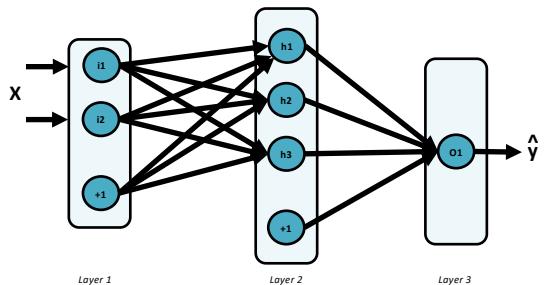
---

---

---

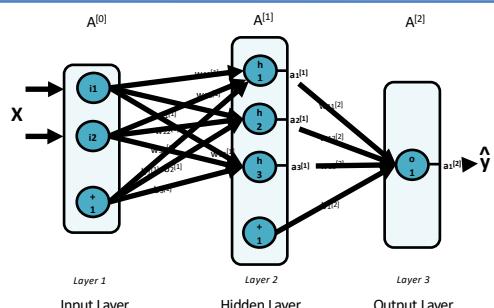
---

## Un exemple de réseau



10

## Un exemple de réseau



11

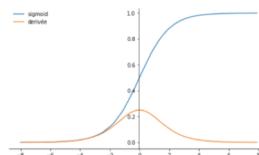
## Choix de la fonction d'activation

Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ 1 & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Softplus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

12

## Attention aux propriétés

- les réseaux de neurones utilisent la descente de gradient
  - le comportement de la dérivée des fonctions est important
- sigmoid transforme de grandes valeurs d'entrée dans des valeurs comprises entre 0 et 1
  - modification importante de l'entrée entraîne une modification mineure de la sortie
  - la dérivée est encore plus petite



13

---

---

---

---

---

---

---

## Disparition de gradient

- Vanishing gradient*
- Généralement des réseaux avec beaucoup de couches
- De trop petites petites valeurs de gradient (le gradient de la fonction de perte approche 0) indiquent que les poids des premiers layers ne seront pas mis à jour efficacement à chaque étape
- Imprécision globale du réseau
  - Exemple : réseau composé de nombreuses couches avec une sigmoid*



14

---

---

---

---

---

---

---

## Mort d'un neurone

- Dead neuron*
- C'est un neurone qui, lors de l'apprentissage, ne s'active plus
- Lié au fait que les dérivées sont très petites ou nulles. Le neurone ne peut donc pas mettre à jour les poids
- Les erreurs ne se propageant plus, ce neurone peut affecter les autres neurones du réseau.
  - Exemple : ReLu qui renvoie 0 quand l'entrée est inférieure ou égale à 0. Si chaque exemple donne une valeur négative, le neurone ne s'active pas et après la descente de gradient le neurone devient 0 donc ne sera plus utilisé. Le Leaky Relu permet de résoudre ce problème.*



15

---

---

---

---

---

---

---

## Explosion de gradient

- *Exploding gradient*
- le problème se pose lorsque des gradients d'erreur important s'accumulent et entraînent des mises à jour importantes des poids. Cela amène un réseau instable : les valeurs de mises à jour des poids peuvent être trop grandes et être remplacées par des NaN donc non utilisables
- Le problème est lié au type de descente de gradient utilisé (Batch vs mini-batch), au fait qu'il y a peut être trop de couches dans le réseau et bien sûr à certaines fonctions d'activation qui favorisent ce problème



16

---

---

---

---

---

---

---

---

## Saturation de neurones

- *Saturated neurons*
- le problème est lié au fait que les grandes valeurs (resp. petites) atteignent un plafond et qu'elles ne changent pas lors de la propagation dans le réseau
- Principalement lié aux fonctions sigmoid et tanh. sigmoid, pour toutes les valeurs supérieures à 1 va arriver sur un plateau et retournera toujours 1. Pour cela, ces deux fonctions d'activations sont assez déconseillées en deep learning (préférer Relu ou Leaky Relu)



17

---

---

---

---

---

---

---

---

## Connaître les propriétés

<https://dashee87.github.io/deep%20learning/visualizing-activation-functions-in-neural-networks/>



18

---

---

---

---

---

---

---

---

## Deux étapes

- Forward propagation
- Backward propagation



19

---

---

---

---

---

---

---

---

---

## Forward propagation

Nous avons vu que :  $\mathbf{z}_i^{[l]} = \mathbf{w}_i^T \cdot \mathbf{a}^{[l-1]} + b_i$      $\mathbf{a}_i^{[l]} = \sigma^{[l]}(\mathbf{z}_i^{[l]})$

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]}$$

En prenant la notation matricielle :

$$\mathbf{A}^{[l]} = \sigma^{[l]}(\mathbf{Z}^{[l]})$$

Nous savons que :  $\mathbf{A}^{[0]} = \mathbf{X}$

Avec Relu et sigmoid comme fonctions d'activation :

$$\mathbf{Z}^{[1]} = \mathbf{W}^{[1]} \cdot \mathbf{A}^{[0]} + \mathbf{b}^{[1]}$$

$$\mathbf{A}^{[1]} = \text{ReLU}^{[1]}(\mathbf{Z}^{[1]})$$

Résultat  $\hat{\mathbf{y}} = \mathbf{A}^{[2]}$

$$\mathbf{Z}^{[2]} = \mathbf{W}^{[2]} \cdot \mathbf{A}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{A}^{[2]} = \text{Sigmoid}^{[2]}(\mathbf{Z}^{[2]})$$

20

---

---

---

---

---

---

---

---

---

## démonstration

- Voir notebook réseaux de neurones



21

---

---

---

---

---

---

---

---

---

## Backward propagation

- L'objectif de la Backward propagation est tout d'abord d'évaluer la différence entre la valeur prédite et la valeur réelle : calcul du coût/perte
- Cross entropy

$$\text{Cost}(\hat{y}, y) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y})$$

- Propager l'erreur dans tous le réseau pour mettre à jour les différents poids : Backward propagation



22

## Backward propagation

Rappel Forward propagation

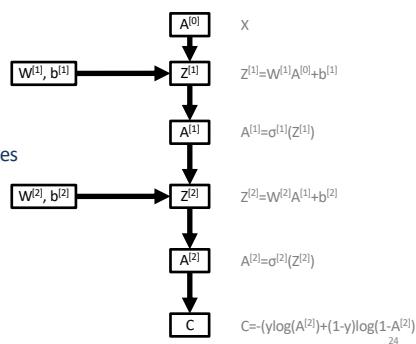
$$\begin{aligned}
 A^{[0]} &= X \\
 Z^{[1]} &= W^{[1]} \cdot A^{[0]} + b^{[1]} \\
 A^{[1]} &= \sigma^{[1]}(Z^{[1]}) \\
 Z^{[2]} &= W^{[2]} \cdot A^{[1]} + b^{[2]} \\
 A^{[2]} &= \sigma^{[2]}(Z^{[2]}) \\
 &\vdots \\
 Z^{[L]} &= W^{[L]} \cdot A^{[L-1]} + b^{[L]} \\
 A^{[L]} &= \sigma^{[L]}(Z^{[L]}) = \hat{y}
 \end{aligned}$$



23

## Backward propagation

Opérations effectuées dans le réseau



## Backward propagation

- Reporter sur le réseau l'ensemble des modifications à apporter à partir du coût obtenu
- Repartir en sens inverse en calculant à chaque fois les dérivées du coût par rapport aux fonctions associées jusqu'au dernier niveau ( $A^{[1]}$ )



25

---

---

---

---

---

---

---

## Backward propagation

- Chaîne de dérivation (*chain rule*)  $\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$
- C** dépend de  $A^{[2]}, A^{[2]}$  dépend lui-même de  $Z^{[2]}, Z^{[2]}$  qui dépend lui-même de  $W^{[2]}$  et de  $b^{[2]}$

$$\frac{\partial C}{\partial W^{[2]}} = \frac{\partial C}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial C}{\partial b^{[2]}} = \frac{\partial C}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial b^{[2]}}$$



26

---

---

---

---

---

---

---

## Backward propagation

- De la même manière
- Pour avoir la dérivée partielle de **C** par rapport à  $W^{[1]}$  et  $b^{[1]}$ ,  $Z^{[2]}$  dépend de  $A^{[1]}$ , qui elle-même dépend de  $Z^{[1]}$  et que finalement  $Z^{[1]}$  dépend de  $W^{[1]}$  et  $b^{[1]}$

$$\frac{\partial C}{\partial W^{[1]}} = \frac{\partial C}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial C}{\partial b^{[1]}} = \frac{\partial C}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial b^{[1]}}$$



27

---

---

---

---

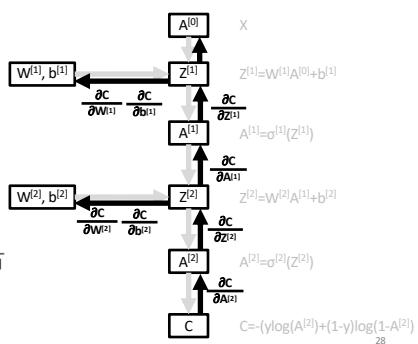
---

---

---

## Backward propagation

dérivées partielles de la fonction de coût en fonction des poids et des biais d'une couche  $l$



28

## Backward propagation

- Pour obtenir les dérivées partielles de  $C$  par rapport à  $W^{[l]}$  et  $b^{[l]}$  il faut calculer les dérivées partielles :

$$\frac{\partial C}{\partial A^{[L]}}, \frac{\partial C}{\partial Z^{[L]}}, \frac{\partial C}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial W^{[l]}}, \frac{\partial Z^{[l]}}{\partial b^{[l]}}$$

29



$$\frac{\partial C}{\partial A^{[L]}}$$

$$\frac{\partial C}{\partial A^{[L]}} = \frac{\partial(-y \log(A^{[L]}) - (1-y) \log(1 - A^{[L]}))}{\partial A^{[L]}}$$

$$\text{La dérivée de } \log(x) \text{ est : } \frac{\partial \log(x)}{\partial x} = \frac{1}{x}$$

$$\text{Pour la partie gauche } -y \log(A^{[L]}) \text{ nous avons : } \frac{-y}{A^{[L]}}$$

Pour la partie droite  $-(1 - y) \log(1 - A^{[L]})$  en appliquant la dérivée d'une fonction

$$\frac{\partial \log(g(x))}{\partial x} = \frac{1}{g(x)} g'(x)$$

30

$$\frac{\partial C}{\partial A^{[L]}}$$

comme la dérivée de  $\mathbf{1} - \mathbf{A}^{[L]}$  est  $-\mathbf{1}$  nous avons au final :

$$\begin{aligned}\frac{\partial C}{\partial A^{[L]}} &= \frac{-y}{A^{[L]}} - (-) \frac{(1-y)}{(1-A^{[L]})} \\ &= \left( \frac{-y}{A^{[L]}} + \frac{(1-y)}{(1-A^{[L]})} \right)\end{aligned}$$

$$\frac{\partial C}{\partial A^{[L]}} = \left( \frac{-y}{A^{[L]}} + \frac{(1-y)}{(1-A^{[L]})} \right)$$

31

---

---

---

---

---

---

$$\frac{\partial C}{\partial Z^{[L]}}$$

$$\frac{\partial C}{\partial Z^{[L]}} = \frac{\partial C}{\partial A^{[L]}} \cdot \frac{\partial A^{[L]}}{\partial Z^{[L]}} = \frac{\partial C}{\partial A^{[L]}} * \sigma'^{[L]}(Z^{[L]})$$

$\sigma'^{[L]}(Z^{[L]})$  : dérivée de la sigmoid (cf notebook descente de gradient)

$$\frac{\partial A^{[L]}}{\partial Z^{[L]}} = \text{sigmoid}(Z^{[L]})(1 - \text{sigmoid}(Z^{[L]})) = A^{[L]}(1 - A^{[L]})$$

$$\frac{\partial C}{\partial A^{[L]}} \cdot \frac{\partial A^{[L]}}{\partial Z^{[L]}} = \left( \frac{-y}{A^{[L]}} + \frac{(1-y)}{(1-A^{[L]})} \right) A^{[L]}(1 - A^{[L]})$$

32

---

---

---

---

---

---

$$\frac{\partial C}{\partial Z^{[L]}}$$

En multipliant par  $(1 - A^{[L]})$  et  $(A^{[L]})$  pour simplifier :

$$\begin{aligned}&= \left( \frac{-y(1-A^{[L]})}{A^{[L]}(1-A^{[L]})} + \frac{A^{[L]}(1-y)}{A^{[L]}(1-A^{[L]})} \right) A^{[L]}(1 - A^{[L]}) \\ &= \left( \frac{-y(1-A^{[L]}) + A^{[L]}(1-y)}{A^{[L]}(1-A^{[L]})} \right) A^{[L]}(1 - A^{[L]})\end{aligned}$$

En supprimant  $A^{[L]}(1 - A^{[L]})$

$$\begin{aligned}&= (-y(1-A^{[L]}) + A^{[L]}(1-y)) \\ &= -y + yA^{[L]} + A^{[L]} - A^{[L]}y \\ &= -y + A^{[L]}\end{aligned}$$

$$\frac{\partial C}{\partial Z^{[L]}} = A^{[L]} - y$$

33

---

---

---

---

---

---

$$\frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l]}}$$

Pour un niveau  $l$  dérivée partielle de  $\mathbf{C}$  par rapport à  $\mathbf{Z}^{[l]}$

Si on connaît  $\mathbf{Z}^{[l]}$  on peut calculer  $\mathbf{Z}^{[L-1]}, \mathbf{Z}^{[L-2]},$  etc

$$\frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l]}} = \frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l+1]}} \cdot \frac{\partial \mathbf{Z}^{[l+1]}}{\partial \mathbf{A}^{[l]}} \cdot \frac{\partial \mathbf{A}^{[l]}}{\partial \mathbf{Z}^{[l]}}$$

$$\begin{aligned} \mathbf{Z}^{[l+1]} &= \mathbf{W}^{[l+1]} \cdot \mathbf{A}^{[l]} + \mathbf{b}^{[l+1]} & \frac{\partial \mathbf{A}^{[l]}}{\partial \mathbf{Z}^{[l]}} &= \sigma'^{[l]}(\mathbf{Z}^{[l]}) \\ \frac{\partial \mathbf{Z}^{[l+1]}}{\partial \mathbf{A}^{[l]}} &= \frac{\partial(\mathbf{W}^{[l+1]} \cdot \mathbf{A}^{[l]} + \mathbf{b}^{[l+1]})}{\partial \mathbf{A}^{[l]}} & \\ &= \mathbf{W}^{[l+1]} \end{aligned}$$

$$\boxed{\frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l]}} = (\mathbf{W}^{[l+1]T} \cdot \frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l+1]}}) * \sigma'^{[l]}(\mathbf{Z}^{[l]})}$$

34

---

---

---

---

---

---

$$\frac{\partial \mathbf{C}}{\partial \mathbf{W}^{[l]}}$$

Dérivée partielle de  $\mathbf{Z}^{[l]}$  par rapport à  $\mathbf{W}^{[l]}$

$$\begin{aligned} \mathbf{Z}^{[l]} &= \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \\ \frac{\partial \mathbf{Z}^{[l]}}{\partial \mathbf{W}^{[l]}} &= \frac{\partial(\mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]})}{\partial \mathbf{W}^{[l]}} \\ &= \mathbf{A}^{[l-1]} \end{aligned}$$

Dérivée partielle de  $\mathbf{C}$  par rapport à  $\mathbf{W}^{[l]}$

$$\boxed{\frac{\partial \mathbf{C}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l]}} \cdot \mathbf{A}^{[l-1]T}}$$

35

---

---

---

---

---

---

$$\frac{\partial \mathbf{C}}{\partial \mathbf{b}^{[l]}}$$

Dérivée partielle de  $\mathbf{Z}^{[l]}$  par rapport à  $\mathbf{b}^{[l]}$

$$\begin{aligned} \mathbf{Z}^{[l]} &= \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \\ \frac{\partial \mathbf{Z}^{[l]}}{\partial \mathbf{b}^{[l]}} &= \frac{\partial(\mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]})}{\partial \mathbf{b}^{[l]}} \\ &= 1 \end{aligned}$$

Dérivée partielle de  $\mathbf{C}$  par rapport à  $\mathbf{b}^{[l]}$

$$\boxed{\frac{\partial \mathbf{C}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathbf{C}}{\partial \mathbf{Z}^{[l]}}}$$

36

---

---

---

---

---

---

## Pour résumer

Pour le layer L

$\frac{\partial C}{\partial A^{[L]}}$	$\left( \frac{-y}{A^{[L]}} + \frac{(1-y)}{1-A^{[L]}} \right)$
$\frac{\partial C}{\partial Z^{[L]}}$	$(A^{[L]} - y)$
$\frac{\partial C}{\partial W^{[L]}}$	$\frac{\partial C}{\partial Z^{[L]}} \cdot (A^{[L-1]})^T$
$\frac{\partial C}{\partial b^{[L]}}$	$\frac{\partial C}{\partial Z^{[L]}}$

Pour un layer l

$\frac{\partial C}{\partial Z^{[l]}}$	$(W^{[l+1]})^T \cdot \frac{\partial C}{\partial Z^{[l+1]}} * \sigma'([l])(Z^{[l]})$
$\frac{\partial C}{\partial W^{[l]}}$	$\frac{\partial C}{\partial Z^{[l]}} \cdot A^{[l-1] T}$
$\frac{\partial C}{\partial b^{[l]}}$	$\frac{\partial C}{\partial Z^{[l]}}$

37

## La descente de gradient

Il suffit d'utiliser les dérivées calculées précédemment et de reporter les modifications

Pour l du dernier layer au layer 1 {

$$\begin{aligned} W^{[l]} &= W^{[l]} - \eta \frac{\partial C}{\partial W^{[l]}} \\ b^{[l]} &= b^{[l]} - \eta \frac{\partial C}{\partial b^{[l]}} \end{aligned}$$

}

38

## Demo

- Notebook réseau de neurones (fonctions, classification binaire)



39

## Classification multi-classes

- Jusqu'à présent : classification binaire
- Pour faire de la classification multi-classes, fonction d'activation : softmax
- Attribution des probabilités à chaque classe d'un problème à plusieurs classes et la somme de ces probabilités doit être égale à 1



40

---



---



---



---



---



---



---



---

## Softmax

Formellement :

Entrée : vecteur  $\mathbf{z}$  de  $C$ -dimensions (le nombre de classes possibles)  
Sortie : vecteur  $\mathbf{a}$  de  $C$ -dimensions de valeurs réelles comprises entre 0 et 1

$$\mathbf{a}_i = \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}$$

Pour  $i = 1 \dots C$   
avec  $\sum_{i=1}^C = 1$

où  $C$  est le nombre de classes

41

---



---



---



---



---



---



---



---

## Softmax

Fonction instable

```
1  nums = np.array([4000, 5000, 6000])
2  print(softmax(nums))
[nan nan nan]
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/ipykernel_launcher.py:2: RuntimeWarning:
  nting: overflow encountered in exp
```

Multiplication par une constante au numérateur et au dénominateur

$$\mathbf{a}_i = \frac{e^{z_i - \max(z)}}{\sum_{k=1}^C e^{z_k - \max(z)}}$$

```
1  def softmax(z):
2      expz = np.exp(z - np.max(z))
3      return expz / expz.sum(axis=0, keepdims=True)
4
5  nums = np.array([4, 5, 6])
6  print(softmax(nums))
7  print ("la somme des probabilités donne 1")
8  nums = np.array([4000, 5000, 6000])
9  print(softmax(nums))
```

```
[0.09003057 0.24472847 0.66524096]
la somme des probabilités donne 1
[0. 0. 1.]
```

42

---



---



---



---



---



---



---



---

## Dérivée de softmax

Considérer que  $\mathbf{g}(\mathbf{x}) = \mathbf{e}^{\mathbf{z}}$        $\mathbf{h}(\mathbf{x}) = \sum_{k=1}^C \mathbf{e}^{z_k}$

La dérivée d'une fonction  $\mathbf{f}(\mathbf{x}) = \frac{\mathbf{g}(\mathbf{x})}{\mathbf{h}(\mathbf{x})}$  est

$$\mathbf{f}'(\mathbf{x}) = \frac{\mathbf{g}'(\mathbf{x})\mathbf{h}(\mathbf{x}) - \mathbf{h}'(\mathbf{x})\mathbf{g}(\mathbf{x})}{\mathbf{h}(\mathbf{x})^2}$$

Simplification de notation     $\sum_C = \sum_{k=1}^C \mathbf{e}^{z_k}$

Pour  $i = 1..C$  nous avons     $\mathbf{a}_i = \frac{\mathbf{e}^{z_i}}{\sum_C}$

43

## Dérivée de softmax

La dérivée  $\frac{\partial \mathbf{a}_i}{\partial z_j}$  de la sortie de softmax  $\mathbf{a}$  par rapport à  $z$  :

Si  $i=j$

$$\frac{\partial a_i}{\partial z_i} = \frac{\partial \left( \frac{e^{z_i}}{\sum_C} \right)}{\partial z_i} = \frac{e^{z_i} \sum_C -e^{z_i} e^{z_i}}{\sum_C^2} = \frac{e^{z_i}}{\sum_C} \frac{\sum_C -e^{z_i}}{\sum_C} = \frac{e^{z_i}}{\sum_C} \left( 1 - \frac{e^{z_i}}{\sum_C} \right) = a_i \left( 1 - a_i \right)$$

Si  $i \neq j$

$$\frac{\partial \mathbf{a}_i}{\partial z_j} = \frac{\partial \left( \frac{e^{z_i}}{\sum_C} \right)}{\partial z_j} = \frac{0 - e^{z_i} e^{z_j}}{\sum_C^2} = \frac{e^{z_i}}{\sum_C} \frac{e^{z_j}}{\sum_C} = -a_i a_j$$

44

## Dérivée de softmax

Softmax avec la cross entropy (même principe que précédemment)

$$\boxed{\frac{\partial C}{\partial Z^{[L]}} = A^{[L]} - y}$$

Toutes les dérivées précédentes sont donc similaires

45

## Demo

- Notebook réseau de neurones (classification multi-classes)



46

---

---

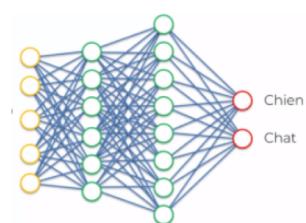
---

---

---

---

## Que se passe-t'il dedans ?



47

---

---

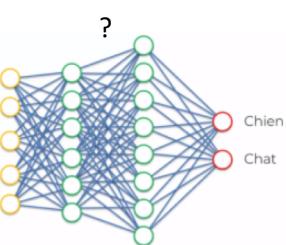
---

---

---

---

## Que se passe-t'il dedans ?



48

---

---

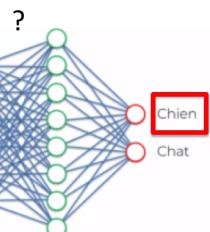
---

---

---

---

## Que se passe-t'il dedans ?



49

---

---

---

---

---

---

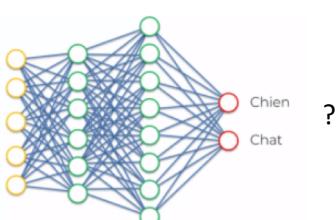
---

---

---

---

## Que se passe-t'il dedans ?



50

---

---

---

---

---

---

---

---

---

---

## Rechercher des patterns

- Un pattern = une signature = chemin suivi par un objet dans le réseau
- Principe :
  - Apprendre le réseau
  - Récupérer pour chaque layer les fonctions d'activations
  - Regrouper, par layer, celles qui ont même valeur (clustering)
  - Afficher les résultats, appliquer un algorithme de recherche de patterns (séquences, trajectoires, etc.)

51

---

---

---

---

---

---

---

---

---

---

## Demo

- <http://www.lirmm.fr/~poncelet/RN>



52

---

---

---

---

---

---

---

- Des questions ?



53

---

---

---

---

---

---

---