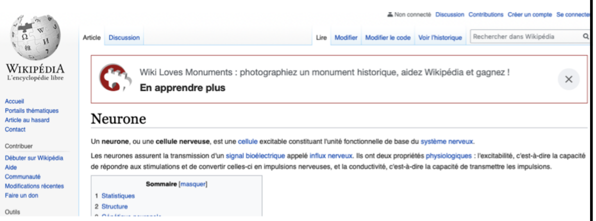
Les réseaux de neurones

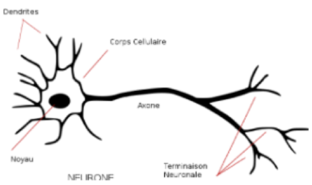
Pascal Poncelet
LIRMM
Pascal.Poncelet@lirmm.fr
http://www.lirmm.fr/~poncelet

Qu’est ce que c’est qu’un neurone ?



2

Un neurone

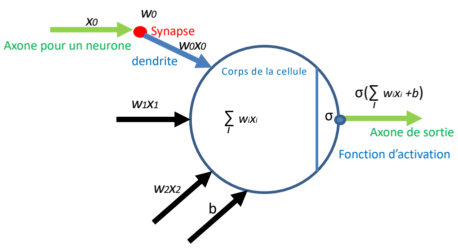


Les dendrites reçoivent l’influx nerveux d’autres neurones. Le neurone évalue alors l’ensemble de la stimulation reçue. Si celle-ci est suffisante, il est excité : il transmet un signal (0/1) le long de l ’axone et l’excitation est propagée jusqu’aux autres neurones qui y sont connectés via les synapses.

3

Un neurone artificiel

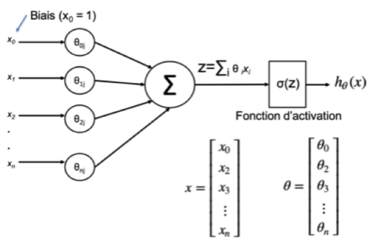
"Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques" (Wikipedia)



4

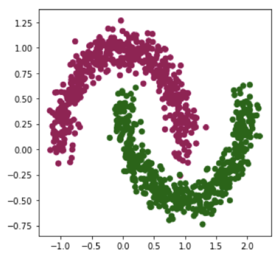
La régression logistique

- Rappel : utilisation d'une sigmoid



5

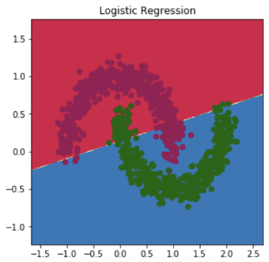
Appliquons la sur ce jeu de données



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, ŷ, 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, σ

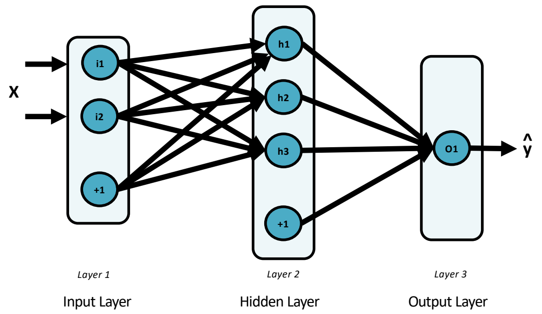
8

Couche de sortie

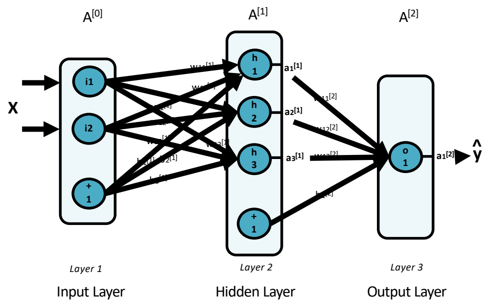
- Elle doit avoir autant de neurones qu'il y a de sorties au problème de classification :
- *régression* : 1 seul neurone (C.f. notebook descente de gradient)
- *classification binaire* : 1 seul neurone avec une fonction d'activation qui sépare les deux classes
- *classification multi-classe* : 1 neurone 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



Un exemple de réseau

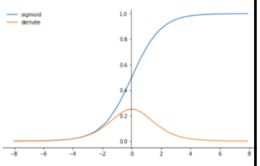


Choix de la fonction d'activation

Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
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)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

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/visualising-activation-functions-in-neural-networks/

18

Deux étapes

- Forward propagation
- Backward propagation

19

Forward propagation

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

En prenant la notation matricielle :

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

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

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

Avec Relu et sigmoid comme fonctions d'activation :

$$\mathbf{Z}^{[1]} = \mathbf{W}^{[1]} . \mathbf{A}^{[0]} + \mathbf{b}^{[1]}$$
$$\mathbf{A}^{[1]} = ReLu^{[1]}(\mathbf{Z}^{[1]})$$

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

$$\mathbf{Z}^{[2]} = \mathbf{W}^{[2]} . \mathbf{A}^{[1]} + \mathbf{b}^{[2]}$$
$$\mathbf{A}^{[2]} = Sigmoid^{[2]}(\mathbf{Z}^{[2]})$$

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

  Cost(ŷ,y)=−ylog(ŷ)−(1−y)log(1−ŷ)

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

22

Backward propagation

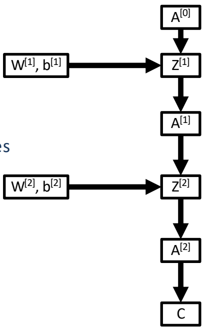
Rappel Forward propagation

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

23

Backward propagation

Opérations effectuées dans le réseau



X

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

$A^{[1]}=\sigma^{[1]}(Z^{[1]})$

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

$A^{[2]}=\sigma^{[2]}(Z^{[2]})$

$C=-(y\log(A^{[2]})+(1-y)\log(1-A^{[2]})$

24

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

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

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

26

Backward propagation

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

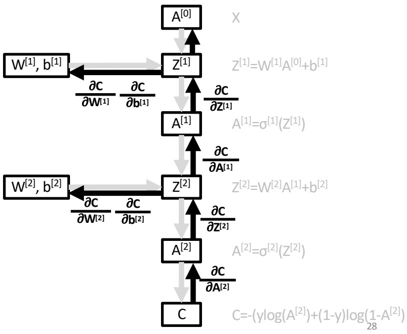
$$\frac{\partial \mathbf{C}}{\partial \mathbf{W}^{[1]}} = \frac{\partial \mathbf{C}}{\partial \mathbf{A}^{[2]}} \cdot \frac{\partial \mathbf{A}^{[2]}}{\partial \mathbf{Z}^{[2]}} \cdot \frac{\partial \mathbf{Z}^{[2]}}{\partial \mathbf{A}^{[1]}} \cdot \frac{\partial \mathbf{A}^{[1]}}{\partial \mathbf{Z}^{[1]}}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

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

27

Backward propagation

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



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

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

Backward propagation

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

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

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

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

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

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

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

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

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

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

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

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

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

31

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

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

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

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

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

32

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

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

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

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

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

$$= (-y(1-A^{[L]}) + A^{[L]}(1-y))$$

$$= -y + yA^{[L]} + A^{[L]} - A^{[L]}y$$

$$= -y + A^{[L]}$$

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

33

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

Pour un niveau $\mathbf{l}$ 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

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

$$Z^{[l+1]} = W^{[l+1]} \bullet A[l] + b[l+1]$$

$$\frac{\partial A^{[l]}}{\partial Z^{[l]}} = \sigma'^{[l]}(Z^{[l]})$$

$$\frac{\partial Z^{[l+1]}}{\partial A^{[l]}} = \frac{\partial (W^{[l+1]} \bullet A[l] + b[l+1])}{\partial A^{[l]}}$$

$$= W^{[l+1]}$$

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

34

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

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

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

$$\frac{\partial Z^{[l]}}{\partial W^{[l]}} = \frac{\partial (W^{[l]} \bullet A[l-1] + b[l])}{\partial W^{[l]}}$$

$$= A^{[l-1]}$$

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

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

35

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

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

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

$$\frac{\partial Z^{[l]}}{\partial b^{[l]}} = \frac{\partial (W^{[l]} \bullet A[l-1] + b[l])}{\partial b^{[l]}}$$

$$= 1$$

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

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

36

Pour résumer

Pour le layer L

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

Pour un layer l

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

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 {

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

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

}

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 **z** de C-dimensions (le nombre de classes possibles)
Sortie : vecteur **a** de C-dimensions de valeurs réelles comprises entre 0 et 1

Pour i = 1 … C

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

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

où *C* est le nombre de classes

41

Softmax

Fonction instable

```
nums = np.array([4000, 5000, 6000])
print(softmax(nums))
```

```
[nan nan nan]
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/ipykernel_launcher.py:2: RuntimeWar
ning: overflow encountered in exp
```

Multiplication par une constante au numérateur et au dénominateur

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

```
def softmax(z):
    expz = np.exp(z - np.max(z))
    return expz / expz.sum(axis=0, keepdims=True)

nums = np.array([4, 5, 6])
print(softmax(nums))
print ("la somme des probabilités donne 1")
nums = np.array([4000, 5000, 6000])
print(softmax(nums))
```

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

42

Dérivée de softmax

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

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

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

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

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

43

Dérivée de softmax

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

Si i=j

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

Si i≠ j

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

44

Dérivée de softmax

Softmax avec la cross entropy (même principe que précédemment)

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

Toutes les dérivées précédentes sont donc similaires

45

Demo

- Notebook réseau de neurones (classification multi-classes)

46

- Des questions ?

47