Chapitre 6.2 - Le réseau de neurones

LES CONCEPTS DE BASE **2**

Le réseau de neurones 2
Le signal d'un neurone 2
Les poids d'un neurone 3
Le biais d'un neurone 3

LES FONCTIONS **4**

La fonction d'agrégation 4
La différentielle de la fonction d'agrégation *4*
La fonction de la somme pondérée *5*
La fonction de la somme quadratique *6*
La fonction d'activation 7
La différentielle de la fonction d'activation *7*
La fonction sigmoïde *8*
La fonction tanh *9*
La fonction ReLU *10*
La fonction Leaky ReLU *11*
La fonction Randomized Leaky ReLU (2015) *11*
La fonction ELU *11*
La fonction swish *12*
La fonction softmax *15*
La fonction softmax numériquement stable *17*
La fonction d'erreur 18
La différentielle de la fonction d'erreur *18*
La fonction d'erreur linéaire *19*
La fonction d'erreur quadratique *19*
La fonction d'erreur cross entropy *19*

L'APPRENTISSAGE **20**

Le gradient de la fonction d'erreur 20
L'algorithme de l'apprentissage 21
La propagation du gradient 22

LES ÉQUATIONS DE LA PROPAGATION DU GRADIENT **25**

L'équation de la propagation de l'erreur de la fonction d'erreur 25
L'équation de la propagation de l'erreur de la fonction d'activation 25
L'équation de la propagation de l'erreur de la fonction d'activation à plusieurs neurones 26
L'équation de la propagation de l'erreur de la fonction d'agrégation 27
Le tableau récapitulatif des équations de la propagation du gradient 29

APPENDICE **30**

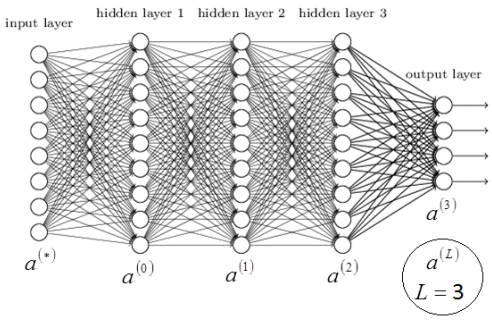
Les notations 30

Les concepts de base

Le réseau de neurones

Un réseau de neurones est une structure mathématique ordonnée en séquence de couches de donnée permettant de générer à partir d'un vecteur d'entrée $a^{(*)}$ un vecteur de sortie $a^{(L)}$. Le but du réseau sera de faire cheminer un vecteur $a^{(*)}$ par l'action de l'activation au travers l'ensemble des couches du réseau afin d'extraire une caractéristique au vecteur d'entrée initial sous la forme d'un vecteur de sortie $a^{(L)}$. Le réseau de neurone joue ainsi le rôle de fonction mathématique.

L'activation d'un réseau se fait en séquence de couche. Dans la notation retenue, nous utiliserons la séquence suivante pour un réseau contenant M couches de neurones :



Exemple de réseau pleinement connecté (*full connected network*)

Couche #1		Couche #2		...	Couche #*m*		...	Dernière couche #*M* $M = L+1$	
$k = 0$		$k = 1$			$k = m-1$			$k = L$	
Entrée	Sortie	Entrée	Sortie		Entrée	Sortie		Entrée	Sortie
$a^{(*)}$	$a^{(0)}$	$a^{(0)}$	$a^{(1)}$		$a^{(k-1)}$	$a^{(k)}$		$a^{(L-1)}$	$a^{(L)}$

Le signal d'un neurone

Dans un réseau de neurone, on utiliser l'expression « signal » pour représenter les composantes d'un vecteur. On distingue deux types de signaux pour une couche *k* : le signal d'entrée $a^{(k-1)}$ et le signal de sortie $a^{(k)}$ (l'activation d'une couche) . Idéalement, les composantes $a_i^{(k)}$ doivent être situées entre -1 et 1 pour augmenter la précision du réseau :

https://www.cgtrader.com/3d-models/character/anatomy/nerve-cell-anatomy-in-details

Le signal d'un neurone d'un réseau artificiel a été modélisé sur le concept biologie du neurone que l'on retrouve dans le cerveau.

- $a_i^{(*)}$: Entrée (*input*) du neurone i du réseau de neurone où $i \in [0,\ N^{(*)} - 1]$. Il appartient à une donnée initiale (vecteur d'entrée) ou à la sortie i d'un autre réseau de neurones.
- $a_i^{(0)}$: Sortie (*output*) du neurone i de la 1re couche de neurones du réseau ($k = 0$) avec $i \in [0,\ N^{(0)} - 1]$. Il sera l'entrée pour la 2e couche de neurones du réseau ($k = 1$)
- $a_i^{(k)}$: Sortie (output) du neurone i de la *k*e couche de neurones du réseau où $i \in [0,\ N^{(0)} - 1]$. Il sera l'entrée pour la *k*+1e couche du réseau.
- $a_i^{(L)}$: Sortie (*output*) du neurone i du réseau de neurones où $i \in [0,\ N^{(L)} - 1]$. Il appartient à la dernière couche du réseau ($k = L$).

Les poids d'un neurone

Dans un réseau de neurones, les poids $w^{(k)}$ sont utilisés pour représenter une connectivité entre les signaux d'entrée $a^{(k-1)}$ à une couche k afin de générer un signal de sortie $a^{(k)}$ à la couche k. Cette connectivité permet d'extraire des caractéristiques entre les données $a^{(k-1)}$ et les valeurs des poids $w^{(k)}$ aura pour but de les identifier.

Le poids $w_{ij}^{(k)}$ établira la connexion entre le neurone d'entrée $a_j^{(k-1)}$ et le neurone de sortie $a_i^{(k)}$. Il déterminera l'importance du neurone $a_j^{(k-1)}$ dans l'émission du signal $a_i^{(k)}$. La valeur de $w_{ij}^{(k)}$ est un nombre réel pouvant être positif ou négatif.

Dans une représentation en algèbre linéaire

$$y = mx + b,$$

les poids $w_{ij}^{(k)}$ correspondent à la variable m dans l'équation, le neurone d'entrée $a_j^{(k-1)}$ représente la variable *x* et le neurone de sortie $a_i^{(k)}$ représente la variable *y* Le poids $w^{(*)}$ n'existe pas.

On distinguera deux notations pour les poids d'un réseau :

Notation matricielle (notation ligne-colonne)	Notation tensorielle (notation *in-out*)
$w_{uv}^{(k)}$ (Composant de matrice en ligne *u* et colonne *v*)	$\widetilde{w}_{vu}^{(k)}$ (Composant du tenseur en entrée *v* et sortie *u*)

Ainsi, en notation matricielle, la ligne de la matrice désigne l'ensemble des poids qui seront exploités pour évaluer le signal de sortie d'un neurone.

Le biais d'un neurone

Le biais $b^{(k)}$ (*bias*) est l'ensemble des poids constants applicable au calcul des signaux $a^{(k)}$ appartenant à une couche k.

Le biais $b_i^{(k)}$ est une constants lors du calcul d'un signal $a_i^{(k)}$ appartenant à une couche k. La valeur de $b_i^{(k)}$ est un nombre réel pouvant être positif ou négatif.

Dans une représentation en algèbre linéaire

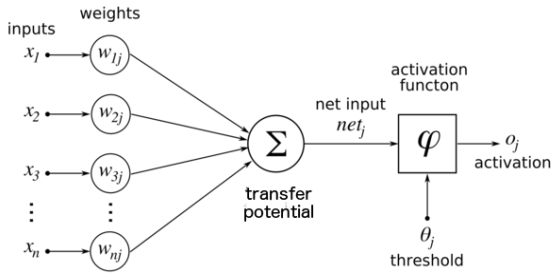
$$y = mx + b,$$

les biais $b^{(k)}$ correspondent à la variable b dans l'équation. Le biais $b^{(*)}$ n'existe pas.

Les fonctions

La fonction d'agrégation

L'agrégation est l'action de regrouper l'ensemble des neurones $a^{(k-1)}$ d'une couche $k-1$ afin de calculer un terme d'agrégation $z_i^{(k)}$ permettant de déterminer la valeur d'un neurone $a_i^{(k)}$ de sortie de la couche k. Il existe plusieurs fonctions d'agrégation. La fonction la plus utilisée est l'agrégation de la somme pondérée. Il est à noter que $z_i^{(*)}$ n'existe pas.



https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089

Schéma de *n* entrées pour déterminer l'agrégation d'un neurone sans l'usage de biais.

Mathématiquement, la fonction d'agrégation prendra la forme de l'équation suivante :

$$z_u^{(k)} = z_u^{(k)}\left(a_v^{(k-1)}, \tilde{w}_{vu}^{(k)}, b_u^{(k)}\right) \qquad \text{avec } v \in \left[0, N^{(k-1)}-1\right] \text{ et } u \in \left[0, N^{(k)}-1\right]$$

où u : Indice de la sortie (*output*) de la fonction.

v : Indice d'entrée (*input*) de la fonction.

$z_u^{(k)}$: Agrégation du neurone u de la couche k.

$a_v^{(k-1)}$: Entrée du neurone v de la couche $k-1$.

$\tilde{w}_{vu}^{(k)}$: Poids du neurone d'entrée v sur le neurone de sortie u appartenant à la couche k.

$b_u^{(k)}$: Biais du neurone de sortie u appartenant à la couche k.

La différentielle de la fonction d'agrégation

L'expression de la différentielle de la fonction d'agrégation

$$z_u^{(k)} = z_u^{(k)}\left(a_v^{(k-1)}, \tilde{w}_{vu}^{(k)}, b_u^{(k)}\right)$$

où

$$v \in \left[0 \ldots N^{(k-1)}-1\right] \qquad \text{et} \qquad u \in \left[0 \ldots N^{(k)}-1\right]$$

est égale à l'expression suivante :

$$\mathrm{d}z_u^{(k)} = \sum_{v=0}^{N^{(k-1)}-1}\left(\frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}}\mathrm{d}a_v^{(k-1)} + \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}}\mathrm{d}\tilde{w}_{vu}^{(k)}\right) + \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}}\mathrm{d}b_u^{(L)}$$

(Différentielle de la fonction d'agrégation)

Preuve :

En construction …

La fonction de la somme pondérée

La somme pondérée (*weighted sum*) est une fonction d'agrégation correspondant à une équation linéaire de type

$$y = m\,x + b$$

où *x* est la variable d'entrée, *y* est la variable de sortie, *m* est la constante linéaire et *b* est une constante.

La somme pondérée (*weighted sum*) $z_i^{(k)}$ d'un neurone i appartenant à la couche k correspond à l'expression suivante qui est déterminée à partir des entrées $a_j^{(k-1)}$ de la couche précédente ou $j \in \left[0, N^{(k-1)}\right]$:

$$z_i^{(k)} = \sum_{j=0}^{N^{(k-1)}-1} w_{ij}^{(k)}\, a_j^{(k-1)} + b_i^{(k)}$$

(Notation des poids en matrice ligne-colonne)

$$z_i^{(k)} = \sum_{i=0}^{N^{(k-1)}-1} \widetilde{w}_{ji}^{(k)}\, a_j^{(k-1)} + b_i^{(k)}$$

(Notation des poids en tenseur *in-out*)

En notation matricielle, la somme pondérée correspond à l'équation suivante :

$$\begin{pmatrix} z_0^{(k)} \\ z_1^{(k)} \\ \dots \\ z_u^{(k)} \\ \dots \\ z_G^{(k)} \end{pmatrix} = \begin{pmatrix} w_{00}^{(k)} & w_{01}^{(k)} & \dots & w_{0v}^{(k)} & \dots & \dots & w_{0F}^{(k)} \\ w_{10}^{(k)} & w_{11}^{(k)} & \dots & w_{1v}^{(k)} & \dots & \dots & w_{1F}^{(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{u0}^{(k)} & w_{u1}^{(k)} & \dots & w_{uv}^{(k)} & \dots & \dots & w_{uF}^{(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{G0}^{(k)} & w_{G1}^{(k)} & \dots & w_{Gv}^{(k)} & \dots & \dots & w_{GF}^{(k)} \end{pmatrix} \begin{pmatrix} a_0^{(k-1)} \\ a_1^{(k-1)} \\ \dots \\ a_v^{(k-1)} \\ \dots \\ a_F^{(k-1)} \end{pmatrix} + \begin{pmatrix} b_0^{(k)} \\ b_1^{(k)} \\ \dots \\ b_u^{(k)} \\ \dots \\ b_G^{(k)} \end{pmatrix}$$

où $F = N^{(k-1)} - 1$: Indice du dernier neurone en entrée (indice de colonne de la matrice).

$G = N^{(k)} - 1$: Indice du dernier neurone en sortie (indice de ligne de la matrice).

Les dérivées partielles de la somme pondérée donnent les résultats suivants :

$$\frac{\partial z_i^{(k)}}{\partial w_{ij}^{(k)}} = a_j^{(k-1)}$$

(Dérivée par rapport au poids)

$$\frac{\partial z_i^{(k)}}{\partial a_j^{(k-1)}} = w_{ij}^{(k)}$$

(Dérivée par rapport à l'entrée)

$$\frac{\partial z_i^{(k)}}{\partial b_i^{(k)}} = 1$$

(Dérivée par rapport au biais)

La fonction de la somme quadratique

La somme quadratique (*quadratic sum*) est une fonction d'agrégation correspondant à une équation du 2e degré de type

$$y = a\,x^2 + bx + c$$

où x est la variable d'entrée, y est la variable de sortie, a est la constante quadratique, b est la constante linéaire et c est une constante.

La somme quadratique[1] (*quadratic sum*) $z_i^{(k)}$ d'un neurone i appartenant à la couche k correspond à l'expression suivante qui est déterminée à partir des entrées $a_j^{(k-1)}$ ou $j \in \left[0, N^{(k-1)}\right]$ de la couche précédente :

$$z_i^{(k)} = \sum_{j=0}^{N^{(k-1)}-1} \left(w_{ij}^{1(k)}\, a_j^{(k-1)} + w_{ij}^{2(k)}\, a_j^{(k-1)} a_j^{(k-1)} \right) + b_i^{(k)}$$

(Notation des poids en matrice ligne-colonne)

$$z_i^{(k)} = \sum_{i=0}^{N^{(k-1)}-1} \left(\tilde{w}_{ji}^{1(k)}\, a_j^{(k-1)} + \tilde{w}_{ji}^{2(k)}\, a_j^{(k-1)} a_j^{(k-1)} \right) + b_i^{(k)}$$

(Notation des poids en tenseur *in-out*)

En notation matricielle, la somme pondérée correspond à l'équation suivante :

$$\begin{pmatrix} z_0^{(k)} \\ z_1^{(k)} \\ \dots \\ z_u^{(k)} \\ \dots \\ z_G^{(k)} \end{pmatrix} = \begin{pmatrix} w_{00}^{1(k)} & w_{01}^{1(k)} & \dots & w_{0v}^{1(k)} & \dots & \dots & w_{0F}^{1(k)} \\ w_{10}^{1(k)} & w_{11}^{1(k)} & \dots & w_{1v}^{1(k)} & \dots & \dots & w_{1F}^{1(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{u0}^{1(k)} & w_{u1}^{1(k)} & \dots & w_{uv}^{1(k)} & \dots & \dots & w_{uF}^{1(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{G0}^{1(k)} & w_{G1}^{1(k)} & \dots & w_{Gv}^{1(k)} & \dots & \dots & w_{GF}^{1(k)} \end{pmatrix} \begin{pmatrix} a_0^{(k-1)} \\ a_1^{(k-1)} \\ \dots \\ a_v^{(k-1)} \\ \dots \\ a_F^{(k-1)} \end{pmatrix} + \begin{pmatrix} w_{00}^{2(k)} & w_{01}^{2(k)} & \dots & w_{0v}^{2(k)} & \dots & \dots & w_{0F}^{2(k)} \\ w_{10}^{2(k)} & w_{11}^{2(k)} & \dots & w_{1v}^{2(k)} & \dots & \dots & w_{1F}^{2(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{u0}^{2(k)} & w_{u1}^{2(k)} & \dots & w_{uv}^{2(k)} & \dots & \dots & w_{uF}^{2(k)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w_{G0}^{2(k)} & w_{G1}^{2(k)} & \dots & w_{Gv}^{2(k)} & \dots & \dots & w_{GF}^{2(k)} \end{pmatrix} \begin{pmatrix} a_0^{(k-1)} * a_0^{(k-1)} \\ a_1^{(k-1)} * a_1^{(k-1)} \\ \dots \\ a_v^{(k-1)} * a_v^{(k-1)} \\ \dots \\ a_F^{(k-1)} * a_F^{(k-1)} \end{pmatrix} + \begin{pmatrix} b_0^{(k)} \\ b_1^{(k)} \\ \dots \\ b_u^{(k)} \\ \dots \\ b_G^{(k)} \end{pmatrix}$$

où $F = N^{(k-1)} - 1$: Indice du dernier neurone en entrée (indice de colonne de la matrice).

$G = N^{(k)} - 1$: Indice du dernier neurone en sortie (indice de ligne de la matrice).

Les dérivées partielles de la somme quadratique donnent les résultats suivants :

$$\frac{\partial z_i^{(k)}}{\partial w_{ij}^{1(k)}} = a_j^{(k-1)}$$

(Dérivée par rapport au poids d'ordre 1)

$$\frac{\partial z_i^{(k)}}{\partial w_{ij}^{2(k)}} = a_j^{(k-1)} a_j^{(k-1)}$$

(Dérivée par rapport au poids d'ordre 2)

$$\frac{\partial z_i^{(k)}}{\partial a_j^{(k-1)}} = w_{ij}^{1(k)} + 2 a_j^{(k-1)} w_{ij}^{2(k)}$$

(Dérivée par rapport à l'entrée)

$$\frac{\partial z_i^{(k)}}{\partial b_i^{(k)}} = 1$$

(Dérivée par rapport au biais)

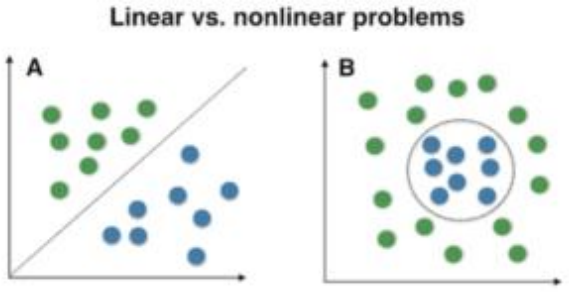
[1] L'efficacité de cette fonction est présentement en étude. Il en existe plusieurs autres formes équivalentes.

La fonction d'activation

L'activation est l'action de transformer le résultat de l'agrégation $z_i^{(k)}$ d'un neurone i appartenant à la couche k en une valeur située idéalement entre -1 et 1. Il existe plusieurs fonctions d'activation avec plusieurs types de valeurs de sortie. Ceci permet de donner un comportement non linéaire au réseau de neurones.

La fonction d'activation $f^{(k)}$ qui génère l'ensemble des neurones $a^{(k)}$ appartenant à la couche k s'applique sur les résultats de l'agrégation $z^{(k)}$ de la façon suivante :

$$a_i^{(k)} = f_i^{(k)}\left(z_i^{(k)}\right) \quad \text{et} \quad i \in \left[0, N^{(k)} - 1\right]$$



https://www.datarobot.com/wiki/neural-network/

Illustration d'une classification à l'aide d'une fonction d'activation linéaire et non linéaire.

où $z_i^{(k)}$ est le paramètre de la fonction $f^{(k)}$. La fonction d'activation $f^{(*)}$ n'existe pas.

Il existe des fonctions d'activation dont l'expression dépend de la valeur de l'agrégation de tous les neurones (*full connected*). Elle peut alors prendre la forme suivante :

$$a_i^{full(k)} = f_i^{full(k)}\left(\vec{z}^{(k)}\right) = f_i^{full(k)}\left(z_0^{(k)}, z_1^{(k)}, \ldots, z_i^{(k)}, \ldots, z_{N^{(k)}-1}^{(k)}\right)$$

La différentielle de la fonction d'activation

La différentielle de la fonction d'activation dépend du nombre de neurones en entrée de celle-ci. Puisque nous avons la fonction d'activation à un neurone

$$a_u^{(k)} = a_u^{(k)}\left(z_u^{(k)}\right)$$

et la fonction d'activation à plusieurs neurones

$$a_u^{full(k)} = a_u^{full(k)}\left(\vec{z}^{(k)}\right) = a_u^{full(k)}\left(z_0^{(k)}, z_1^{(k)}, \ldots, z_p^{(k)}, \ldots, z_{N^{(k)}-1}^{(k)}\right)$$

où

$$u \in \left[0 \ldots N^{(k)} - 1\right],$$

nous appliquons la règle de la dérivée en chaîne et nous obtenons les expression suivante pour la différentielle de nos fonctions d'activations :

Différentielle de la fonction d'activation à un neurone	Différentielle de la fonction d'activation à plusieurs neurones[2] (*full connected*)	Différentielle de la fonction d'activation avec paramètre β [3]
$\mathrm{d}a_u^{(k)} = \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}} \mathrm{d}z_u^{(k)}$	$\mathrm{d}a_u^{full(k)} = \sum_{p=0}^{N^{(k)}-1} \frac{\partial a_u^{(k)}}{\partial z_p^{(k)}} \mathrm{d}z_p^{(k)}$	$\mathrm{d}a_u^{(k)} = \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}} \mathrm{d}z_u^{(k)} + \frac{\partial a_u^{(k)}}{\partial \beta^{(k)}} \mathrm{d}\beta^{(k)}$

Preuve :

En construction …

[2] Voir la fonction *softmax*.
[3] Voir la fonction *swish*.

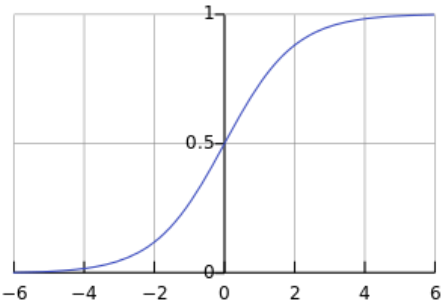
La fonction sigmoïde

La fonction sigmoïde est une fonction d'activation dont le résultat est situé dans l'intervalle $[0,1]$:

Fonction : $\sigma(z)=\dfrac{1}{1+e^{-z}}$

Dérivée : $\dfrac{\mathrm{d}\,\sigma(z)}{\mathrm{d}z}=\sigma(z)(1-\sigma(z))$

De nos jours, la fonction sigmoïde est évitée dans l'activation d'un couche de neurones en raison de la faible valeur de sa dérivée lorsque $|z|>4$. Puisque le taux d'apprentissage dépend de la dérivée de la fonction d'activation, si $|z|$ est trop élevée alors la dérivée chutera vers zéro stoppant l'apprentissage.



https://www.quora.com/Why-is-the-sigmoid-function-rarely-used-in-hidden-layers-recently

Illustration des valeurs de la fonction sigmoïde.

Preuve :

Effectuons la dérivée de la fonction sigmoïde $\sigma(x)$ par rapport à la variable *x* :

$$\sigma'=\frac{\mathrm{d}\sigma}{\mathrm{d}x} \quad \Rightarrow \quad \sigma'=\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{1+e^{-x}}\right) \qquad \left(\sigma(x)=\frac{1}{1+e^{-x}}\right)$$

$$\Rightarrow \quad \sigma'=-\frac{1}{\left(1+e^{-x}\right)^2}\frac{\mathrm{d}}{\mathrm{d}x}\left(1+e^{-x}\right) \qquad \left(\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{f}\right)=-\frac{1}{f^2}\frac{\mathrm{d}f}{\mathrm{d}x}\right)$$

$$\Rightarrow \quad \sigma'=\frac{-1}{\left(1+e^{-x}\right)^2}e^{-x}\frac{\mathrm{d}}{\mathrm{d}x}(-x) \qquad \left(\frac{\mathrm{d}}{\mathrm{d}x}\left(e^{-x}\right)=e^{-x}\frac{\mathrm{d}}{\mathrm{d}x}(-x)\right)$$

$$\Rightarrow \quad \sigma'=\frac{-e^{-x}}{\left(1+e^{-x}\right)^2}\frac{\mathrm{d}}{\mathrm{d}x}(-x)$$

$$\Rightarrow \quad \sigma'=\frac{-e^{-x}}{\left(1+e^{-x}\right)^2}(-1)$$

$$\Rightarrow \quad \sigma'=\frac{e^{-x}}{\left(1+e^{-x}\right)^2}$$

$$\Rightarrow \quad \sigma'=\frac{1}{1+e^{-x}}\left(\frac{e^{-x}}{1+e^{-x}}+\frac{1}{1+e^{-x}}-\frac{1}{1+e^{-x}}\right)$$

$$\Rightarrow \quad \sigma'=\frac{1}{1+e^{-x}}\left(\frac{1+e^{-x}}{1+e^{-x}}-\frac{1}{1+e^{-x}}\right)$$

$$\Rightarrow \quad \sigma'=\sigma(1-\sigma) \quad \blacksquare \qquad \left(\sigma(x)=\frac{1}{1+e^{-x}}\right)$$

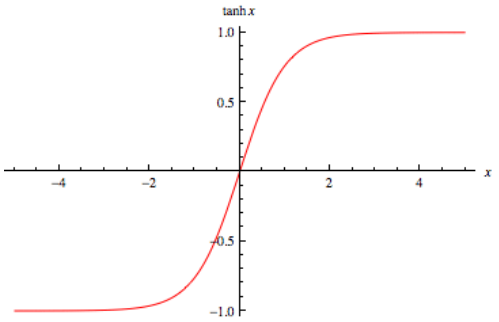
La fonction tanh

La fonction tangente hyperbolique (tanh) est une fonction d'activation dont le résultat est situé dans l'intervalle $[-1,1]$:

Fonction : $\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

Dérivée : $\frac{\mathrm{d}\tanh(z)}{\mathrm{d}z} = 1 - \tanh^{2}(z)$

La fonction tanh est utilisé comme fonction d'activation à l'intérieur du réseau (couche cachée) par ce qu'elle admet des valeurs positives et négatives ce qui augmente la capacité d'apprentissage du réseau. Cependant, on l'évite à la dernière couche si le résultat est le fruit d'une classification (valeur de 0 ou 1).



http://mathworld.wolfram.com/HyperbolicTangent.html

Illustration des valeurs de la fonction tanh.

Preuve :

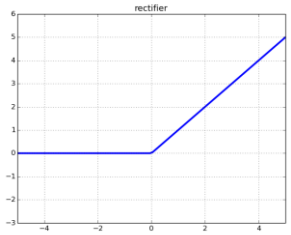
En construction …

La fonction ReLU

La fonction Unité Linéaire Rectifiée

ReLU (rectified linear unit)

En construction …



https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks

Illustration des valeurs de la fonction ReLU.

Fonction : $\mathrm{ReLU}(z) = \begin{cases} 0 \text{ si } z < 0 \\ z \text{ si } z \geq 0 \end{cases}$

ou bien $\mathrm{ReLU}(z) = \max(0, z)$

Dérivée : $\frac{\mathrm{d}\,\mathrm{ReLU}(z)}{\mathrm{d}z} = \begin{cases} 0 \text{ si } z < 0 \\ 1 \text{ si } z \geq 0 \end{cases}$

REMARQUE : (basée sur mon expérience personnelle)

- Puisque la méthode RELU fait bouger de beaucoup les poids et biais des couches, cette fonction d'activation nécessite un gradient avec plusieurs vecteurs (*mini-batch*). Autrement, il n'y aura pas de direction adéquate de prise lors de la descente du gradient et aucune bonne calibration des neurones seront prises.
- L'usage de la méthode ReLU nécessite la méthode SOFTMAX en dernière couche (fonction de classification), car les agrégations générées seront trop élevées et impossible à ajuster avec une fonction d'activation de classification comme SIGMOÏDE, car sa pente est trop faible à haute valeur d'agrégation.
- La fonction ReLU est réputé pour bien propager l'erreur en raison de sa pente de 1. Cependant, si sont activation est toujours négative peu importe le vecteur d'entrée, le neurone sera alors « désactivé » ou « mort ». Ainsi, il y aura moins de neurone pour mettre à jour le réseau. Ainsi, il est souhaitable d'utiliser un grand nombre de neurones dans une couche avec la fonction d'activation ReLU. Pour éviter ce problème, il existe des stratégies de normalisation comme la fonction *batch-normalisation* (voir section 6.3).

La fonction Leaky ReLU

$$\mathrm{LReLU}(z)=\begin{Bmatrix}\varepsilon z \text{ si } z<0\\ z \text{ si } z\geq 0\end{Bmatrix} \quad \text{ou bien} \quad \mathrm{LReLU}(z)=\max(\varepsilon z, z)$$

avec $\varepsilon \in [0,1[$, mais traditionnellement $\varepsilon = 0.1$

$$\frac{\mathrm{d}\,LReLU(z)}{\mathrm{d}z}=\begin{Bmatrix}\varepsilon \text{ si } z<0\\ 1 \text{ si } z\geq 0\end{Bmatrix}$$

La fonction Randomized Leaky ReLU (2015)

On peut également faire une variante de cette fonction où ε peut être déterminé aléatoirement dans l'intervalle $\varepsilon \in [0,\varepsilon_{\max}]$ où $\varepsilon_{\max}$ est inférieur à 1 pour chacun des neurones à activer par la fonction.

La fonction ELU

La fonction unité linéaire exponentiel

Exponential Linear Unit ELU

$$\mathrm{ELU}(z)=\begin{Bmatrix} z & \text{si } z\geq 0\\ \varepsilon\left(e^{z}-1\right) & \text{si } z<0\end{Bmatrix}$$

avec $\varepsilon \in [0,1[$, mais traditionnellement $\varepsilon = 0.1$

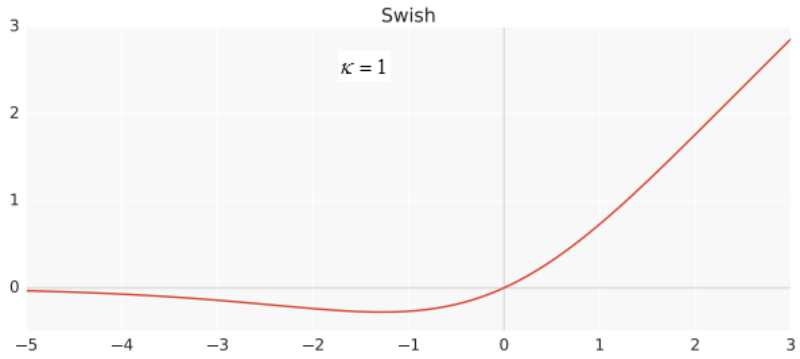
$$\frac{\mathrm{d}\,\mathrm{ELU}(z)}{\mathrm{d}z}=\begin{Bmatrix} 1 & \text{si } z\geq 0\\ \mathrm{ELU}(z)+\varepsilon & \text{si } z<0\end{Bmatrix}$$

La fonction swish

$$\text{swish}(z,\kappa) \equiv \psi(z,\kappa) = \frac{z}{1+e^{-\kappa z}}$$

$$\frac{\partial \psi}{\partial z} = \sigma + \kappa z \sigma(1-\sigma) \quad \text{avec} \quad \sigma = \frac{1}{1+e^{-\kappa z}}$$

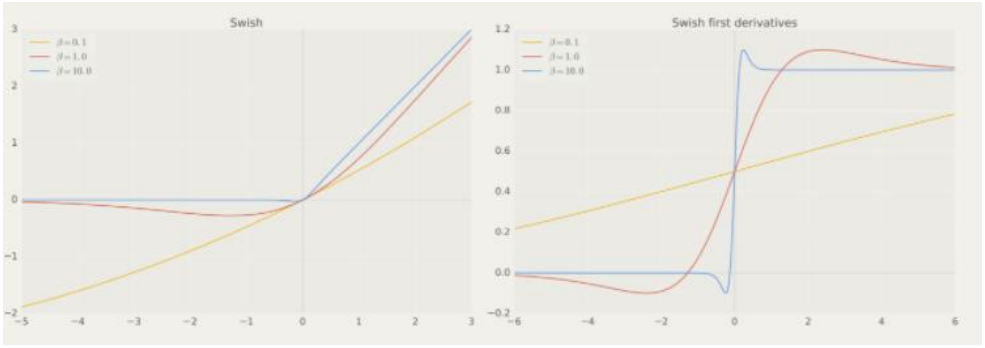
$$\text{et} \qquad \frac{\partial \psi}{\partial \kappa} = z^2 \sigma(1-\sigma)$$



https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820

Le nom « swich » pour *sigmoid and wish* (développée par Google) permet à cette fonction de changer de comportement selon la valeur du paramètre κ qui peut être considéré par un réseau comme une constante ou comme un paramètre pouvant être entraîné (il y aura un paramètre κ pour chaque neurone u lors de l'activation d'une couche de neurones). Ainsi, entraîner le paramètre κ permet au réseau selon le type de donnée de choisir le format idéal de courbure de la fonction pour l'activation de l'agrégation.

- Si $\kappa = 1$, la fonction est équivalente à la fonction $f(z) = z\,\sigma(z)$ (*sigmoid-weighted Linear Unit*, SiL).
- Si $\kappa = 0$, la fonction est équivalent à la fonction linéaire $f(z) = \frac{z}{2}$.
- Si $\kappa \to \infty$, la fonction est équivalent à la fonction ReLU $f(z) = \max(0, z)$.



https://sefiks.com/2018/08/21/swish-as-neural-networks-activation-function/

Preuve : (version où $\beta \equiv \kappa$)

$$\frac{\partial \psi_u}{\partial z_u} = \frac{\partial}{\partial z_u}\left(\frac{z_u}{1+e^{-\beta z_u}}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}}\frac{\partial z_u}{\partial z_u} + z_u \frac{\partial}{\partial z_u}\left(\frac{1}{1+e^{-\beta z_u}}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} + z_u \frac{\partial}{\partial z_u}\left(1+e^{-\beta z_u}\right)^{-1}$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} + z_u (-1)\left(1+e^{-\beta z_u}\right)^{-2}\frac{\partial}{\partial z_u}\left(1+e^{-\beta z_u}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} - \frac{z_u}{\left(1+e^{-\beta z_u}\right)^2}\frac{\partial}{\partial z_u}\left(e^{-\beta z_u}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} - \frac{z_u}{\left(1+e^{-\beta z_u}\right)^2}\left(e^{-\beta z_u}\right)\frac{\partial}{\partial z_u}\left(-\beta z_u\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} - \frac{z_u}{\left(1+e^{-\beta z_u}\right)^2}\left(e^{-\beta z_u}\right)(-\beta)\frac{\partial z_u}{\partial z_u}$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial z_u} = \frac{1}{1+e^{-\beta z_u}} + \beta z_u \frac{e^{-\beta z_u}}{\left(1+e^{-\beta z_u}\right)^2}$$

Si l'on effectue cette restructuration

$$\frac{e^{-\beta z_u}}{\left(1+e^{-\beta z_u}\right)^2} = \frac{e^{-\beta z_u}+1-1}{\left(1+e^{-\beta z_u}\right)^2} = \frac{1}{1+e^{-\beta z_u}}\frac{1+e^{-\beta z_u}-1}{1+e^{-\beta z_u}} = \frac{1}{1+e^{-\beta z_u}}\left(1-\frac{1}{1+e^{-\beta z_u}}\right)$$

et que l'on utilise l'expression

$$\sigma_u = \frac{1}{1+e^{-\beta z_u}} \quad ,$$

nous obtenons

$$\frac{\partial \psi_u}{\partial z_u} = \sigma_u + \beta z_u \sigma_u (1-\sigma_u) \quad \blacksquare$$

$$\frac{\partial \psi_u}{\partial \beta} = \frac{\partial}{\partial \beta}\left(\frac{z_u}{1+e^{-\beta z_u}}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial \beta} = z_u \frac{\partial}{\partial \beta}\left(\left(1+e^{-\beta z_u}\right)^{-1}\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial \beta} = z_u (-1)\left(1+e^{-\beta z_u}\right)^{-2} \frac{\partial}{\partial \beta}\left(1+e^{-\beta z_u}\right)$$

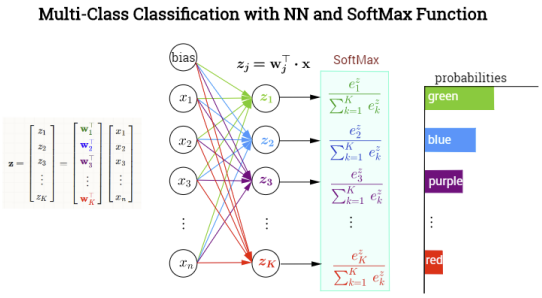
$$\Rightarrow \quad \frac{\partial \psi_u}{\partial \beta} = -\frac{z_u}{\left(1+e^{-\beta z_u}\right)^2}\left(e^{-\beta z_u}\right)\frac{\partial}{\partial \beta}\left(-\beta z_u\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial \beta} = -\frac{z_u e^{-\beta z_u}}{\left(1+e^{-\beta z_u}\right)^2}\left(-z_u\right)$$

$$\Rightarrow \quad \frac{\partial \psi_u}{\partial \beta} = {z_u}^2 \sigma_u (1-\sigma_u) \qquad \frac{e^{-\beta z_u}}{\left(1+e^{-\beta z_u}\right)^2} = \sigma_u (1-\sigma_u) \text{ et } \sigma_u = \frac{1}{1+e^{-\beta z_u}}$$

La fonction softmax

La fonction d'activation *softmax* (fonction exponentielle normalisée) permet de générer une activation strictement positive dont la somme des activations sur l'ensemble des paramètres de la fonction sera égale à 1. Ainsi, la fonction *softmax* aura la forme d'une probabilité où l'activation de la fonction pour chacun de ses paramètres sera une probabilité parmi l'ensemble des probabilités. Cette fonction est régulièrement utilisée comme fonction d'activation à la toute dernière couche d'un réseau (fonction de classification) afin de réaliser une classification parmi plusieurs choix.



http://rinterested.github.io/statistics/softmax.html

Illustration de la distribution des probabilités générées par les activations des agrégations de la couche.

Avec sa nature probabiliste, l'activation correspondra à la probabilité que le réseau accorde à un choix de la classification où l'ensemble des choix possèdent une probabilité de 1.

Fonction	Dérivée partielle
$S_\mu(\vec{z}) = \dfrac{e^{z_\mu}}{\sum_{i=1}^{N} e^{z_i}}$ avec $\vec{z} = (z_1, z_2, ..., z_\mu, ..., z_N)$ et $\mu \in [1, N]$	$\dfrac{\partial S_u(\vec{z})}{\partial z_k} = S_u(\delta_{uk} - S_k)$ et $\delta_{kk} = 1, \delta_{uk} = 0$ (delta de Kronecker)

Preuve :

Effectuons la différentielle de la fonction *softmax* $S_\mu(\vec{x})$ par rapport à la variable $x_\mu \in \vec{x}$:

$$dS_u(\vec{x}) = d\left(\frac{e^{x_u}}{\sum_{i=1}^{N} e^{x_i}}\right)$$

$$\Rightarrow \quad dS_u(\vec{x}) = \sum_{k=1}^{N} \frac{\partial}{\partial x_k}\left(\frac{e^{x_u}}{\sum_{i=1}^{N} e^{x_i}}\right) dx_k$$

$$\Rightarrow \quad dS_u(\vec{x}) = \frac{\partial}{\partial x_u}\left(\frac{e^{x_\mu}}{\sum_{i=1}^{N} e^{x_i}}\right) dx_u + \sum_{\substack{k=1 \\ k \neq u}}^{N} \frac{\partial}{\partial x_k}\left(\frac{e^{x_\mu}}{\sum_{i=1}^{N} e^{x_i}}\right) dx_k$$

$$\Rightarrow \quad dS_u(\vec{x}) = \frac{1}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2}\left[\sum_{i=1}^{N} e^{x_i} \frac{d}{\partial x_u}\left(e^{x_u}\right) - e^{x_u} \frac{\partial}{\partial x_u}\left(\sum_{i=1}^{N} e^{x_i}\right)\right] dx_u + \sum_{\substack{k=1 \\ k \neq u}}^{N} -\frac{e^{x_\mu}}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2} \frac{\partial}{\partial x_k}\left(\sum_{i=1}^{N} e^{x_i}\right) dx_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = \frac{1}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2}\left[\sum_{i=1}^{N} e^{x_i} e^{x_u} - e^{x_u} e^{x_u}\right]\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} \frac{e^{x_\mu}}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2} e^{x_k}\,\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = \frac{e^{x_u}}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2}\left[\sum_{i=1}^{N} e^{x_i} - e^{x_u}\right]\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} \frac{e^{x_\mu} e^{x_k}}{\left(\sum_{i=1}^{N} e^{x_i}\right)^2}\,\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = \frac{e^{x_u}}{\sum_{i=1}^{N} e^{x_i}}\left[1 - \frac{e^{x_u}}{\sum_{i=1}^{N} e^{x_i}}\right]\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} \frac{e^{x_\mu}}{\sum_{i=1}^{N} e^{x_i}} \frac{e^{x_k}}{\sum_{i=1}^{N} e^{x_i}}\,\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = S_u(1 - S_u)\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} S_u S_k\,\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = (S_u - S_u S_u)\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} S_u S_k\,\mathrm{d}x_k$$

La stratégie sera d'unifier nos deux dérivés par l'ajout d'un « rien stratégique ». Introduisons alors le delta de Kronecker[4]

$$\delta_{uk} \qquad \text{où} \qquad \delta_{kk} = 1 \;\text{ et }\; \delta_{uk} = 0 \;\; \forall u, k \in [1, N] \;\; .$$

Ainsi, nous obtenons :

$$\mathrm{d}S_u(\vec{x}) = (S_u - S_u S_u)\mathrm{d}x_u - \sum_{\substack{k=1 \\ k\neq u}}^{N} S_u S_k\,\mathrm{d}x_k \qquad \text{(équation précédent)}$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = (\delta_{uu} S_u - S_u S_u)\mathrm{d}x_u + \sum_{\substack{k=1 \\ k\neq u}}^{N} (\delta_{uk} S_u - S_u S_k)\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = \sum_{k=1}^{N} (\delta_{uk} S_u - S_u S_k)\mathrm{d}x_k$$

$$\Rightarrow \qquad \mathrm{d}S_u(\vec{x}) = S_u \sum_{k=1}^{N} (\delta_{uk} - S_k)\mathrm{d}x_k$$

L'expression de la différentielle prendra la forme

$$\frac{\partial S_u(\vec{x})}{\partial x_k} = S_u(\delta_{uk} - S_k)$$

sous la notation de la dérivée partielle. ■

[4] Référence : https://en.wikipedia.org/wiki/Kronecker_delta

La fonction softmax numériquement stable

En raison de l'expression e^{z_μ} requise dans le calcul de la fonction *softmax*, une erreur numérique peut se propager lors d'une implémentation dans un programme informatique. Par exemple, lorsque $z_u = 40000$, la fonction *softmax* $S_\mu(\vec{z})$ peut engendrer un calcul numérique correspondant à ∞/∞ ce qui donne une valeur indéterminée (NaN, *not a number* en Java).

Afin d'augmenter la stabilité numérique de la fonction *softmax* $S_\mu(\vec{z})$, il est préférable de définir la fonction sous la forme suivante puisque la fonction *softmax* est invariante sous une translation m sur l'ensemble des composante z_μ du vecteur $\vec{z}$:

Fonction	Dérivée partielle
$S_\mu(\vec{z}) = \frac{e^{z_\mu - m}}{\sum_{i=1}^{N} e^{z_i - m}}$ avec $\vec{z} = (z_1, z_2, ..., z_\mu, ..., z_N)$ où $\mu \in [1, N]$ et $m = \max(z_1, z_2, ..., z_\mu, ..., z_N)$	$\frac{\partial S_u(\vec{z})}{\partial z_k} = S_u(\delta_{uk} - S_k)$ et $\delta_{kk} = 1, \delta_{uk} = 0$ (delta de Kronecker)

En réalisant une translation sur la valeur $m = \max(z_1, z_2, ..., z_\mu, ..., z_N)$ correspondant à la composante la plus grande du vecteur $\vec{z}$, nous obtenons la plus grande valeur numérique $e^0 = 1$ et la plus petite valeur numérique possible étant $e^{-\infty} = 0$. Ainsi, $S_\mu(\vec{z})$ ne pourra jamais donner une valeur numérique indéterminée.

Preuve :

Soit la translation c appliquée à la fonction *softmax*, démontrons que le résultat sera équivalent au résultat sans translation :

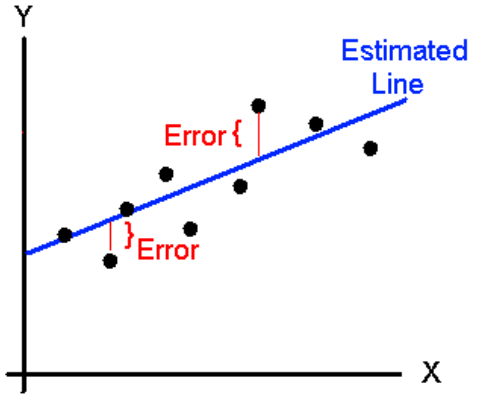
$$S_\mu(\vec{z}) = \frac{e^{z_\mu}}{\sum_{i=1}^{N} e^{z_i}} \quad \Rightarrow \quad S_\mu(\vec{z}) = \frac{e^c}{e^c}\frac{e^{z_\mu}}{\sum_{i=1}^{N} e^{z_i}} \qquad \text{(Multiplier par } \frac{e^c}{e^c}\text{)}$$

$$\Rightarrow \quad S_\mu(\vec{z}) = \frac{e^{z_\mu + c}}{\sum_{i=1}^{N} e^{z_i + c}} \qquad \text{(Propriété des exponentiels : } e^a e^b = e^{a+b}\text{)}$$

$$\Rightarrow \quad S_\mu(\vec{z}) = S_\mu(\vec{z} + \vec{c}) \qquad \vec{c} = (c, c, c, ..., c) \text{ où } c_u = c$$

La fonction d'erreur

La fonction d'erreur[5] C_i évalue la différence entre les valeurs calculées $a^{(L)}$ par le réseau de neurones à partir d'un vecteur d'entrée $a^{(*)}$ avec les valeurs finales attendues y_i associées au vecteur d'entrée. Si le réseau reproduit exactement les valeurs attendues pour un vecteur d'entrée $a^{(*)}$, alors l'erreur sera nulle et aucun apprentissage ne sera possible pour ce vecteur. Autrement, la fonction d'erreur sera utilisée pour mettre à jour les poids $\tilde{w}_{vu}^{(k)}$ et biais $b_u^{(k)}$ du réseau.



http://wiki.fast.ai/images/5/55/Linear_line_w_cost_function.png

L'objectif de la fonction d'erreur est d'évaluer l'écart entre une valeur calculée et une valeur désirée (*Estimated Line*).

Mathématiquement, la fonction d'erreur prendra la forme de l'équation suivante :

$$C = C\left(a_u^{(L)}, y_u\right) \quad \text{et} \quad u \in \left[0, N^{(L)} - 1\right]$$

Il existe plusieurs fonctions d'erreur. Le choix dépendra de la fonction d'activation utilisée à la dernière couche du réseau.

La différentielle de la fonction d'erreur

L'expression de la différentielle de la fonction d'erreur

$$C = C\left(a_u^{(L)}, y_u\right) \qquad \text{où} \qquad u \in \left[0 \ldots N^{(L)} - 1\right]$$

est égale à l'expression suivante :

$$\mathrm{d}C = \sum_{u=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_u^{(L)}} \mathrm{d}a_u^{(L)}$$

(Différentielle de la fonction d'erreur)

Preuve :

Soit la fonction d'erreur

$$C = C\left(a_i^{(L)}, y_i\right)$$

avec $i \in \left[0 \ldots N^{(L)} - 1\right]$ étant l'indice de sortie de l'activation. Appliquons la différentielle sur la fonction d'erreur ce qui nous donnera

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \left(\frac{\partial C}{\partial a_i^{(L)}} \mathrm{d}a_i^{(L)} + \frac{\partial C}{\partial y_i} \mathrm{d}y_i \right) \quad .$$

Puisque $\mathrm{d}y_i = 0 \ \ \forall i \in \left[0 \ldots N^{(L)} - 1\right]$, nous obtenons

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \mathrm{d}a_i^{(L)} \quad . \ \blacksquare$$

[5] Il existe plusieurs synonymes à cette fonction : *Error*, *Cost*, *Loss*.

La fonction d'erreur linéaire

En construction …

Fonction	Dérivée partielle
$C = \sum_{i=0}^{N^{(L)}-1} C_i \quad \text{avec} \quad C_i = a_i^{(L)} - y_i$	$\frac{\partial C}{\partial a_i^{(L)}} = 1$

où y_i : Valeur attendue pour le neurone i.

$a_i^{(L)}$: Valeur obtenue par le réseau pour le neurone i.

La fonction d'erreur quadratique

En construction …

Cette fonction porte également le nom populaire « d'énergie » pour son comportement quadratique tel que l'énergie cinétique $K = 0{,}5\,mv^2$.

Fonction	Dérivée partielle
$C = \sum_{i=0}^{N^{(L)}-1} C_i \quad \text{avec} \quad C_i = \frac{1}{2}\left(a_i^{(L)} - y_i\right)^2$	$\frac{\partial C}{\partial a_i^{(L)}} = a_i^{(L)} - y_i$

où y_i : Valeur attendue pour le neurone i.

$a_i^{(L)}$: Valeur obtenue par le réseau pour le neurone i.

La fonction d'erreur *cross entropy*

En construction …

Cette fonction est utilisée lorsque y_i et $a_i^{(L)}$ correspondent à des distributions de probabilité. On utilise habituellement cette fonction d'erreur lorsque la dernière activation est la fonction *softmax*. Ce choix permet des optimisations dans les calculs de l'apprentissage.

Fonction	Dérivée partielle
$C = \sum_{i=0}^{N^{(L)}-1} C_i \quad \text{avec} \quad C_i = -y_i \ln\left(a_i^{(L)}\right)$	$\frac{\partial C}{\partial a_i^{(L)}} = -\frac{y_i}{\left\lvert a_i^{(L)} \right\rvert}$

où y_i : Valeur probabiliste attendue pour le neurone i.

$a_i^{(L)}$: Valeur probabiliste obtenue par le réseau pour le neurone i.

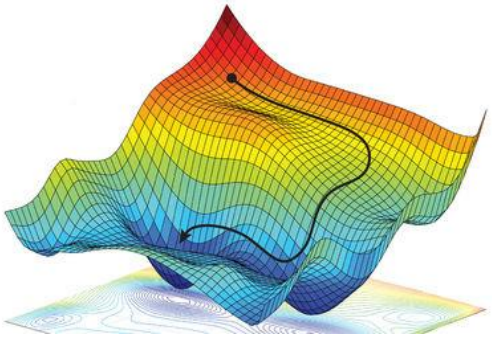
L'apprentissage

Le gradient de la fonction d'erreur

Le gradient de la fonction d'erreur ∇C qui dépend des dérivées partielles

$$\frac{\partial C}{\partial w_{uv}^{(k)}} \text{ et } \frac{\partial C}{\partial b_u^{(k)}}$$

par rapport au poids $w_{uv}^{(k)}$ et biais $b_u^{(k)}$ d'un neurone u d'une couche k permet d'évaluer à quel point ces paramètres du réseau influencent le calcul de l'activation du réseau pour un vecteur d'entrée $a^{(*)}$. Le calcul du gradient dépendra du type de réseau, du choix de la fonction d'erreur et de l'activation du vecteur d'entrée choisi. Sa valeur changera après chaque modification du réseau (modification de $w_{uv}^{(k)}$ et $b_u^{(k)}$) et tendra vers zéro lorsque la fonction d'erreur sera minimisée.



https://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy

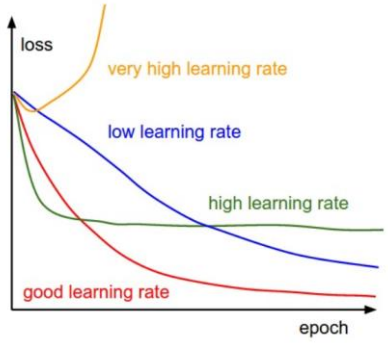
Le gradient de la fonction d'erreur permet de savoir dans quelle direction il faut modifier les paramètres du réseau afin de réduire l'erreur sur l'ensemble des données d'entraînement.

Puisque l'objectif de l'apprentissage sera de réduire la valeur de la fonction d'erreur pour l'ensemble des données d'entraînement, évaluer le gradient permettra de savoir la direction et le dosage du changement à apporter aux paramètres $w_{uv}^{(k)}$ et $b_u^{(k)}$ afin de réduire l'erreur. Puisque le gradient pointe vers une hausse de la fonction d'erreur, il faudra modifier les paramètres $w_{uv}^{(k)}$ et $b_u^{(k)}$ dans le sens contraire du gradient.

Le déplacement dans la fonction d'erreur sera orienté par le gradient de la fonction d'erreur ∇C, mais l'ampleur du déplacement sera pondéré par un facteur portant le nom de taux d'apprentissage (*learning rate*).

Le taux d'apprentissage α influence fortement le rythme de l'apprentissage. Cependant, il influence également la qualité de la convergence de la solution. Il est traditionnellement difficile à déterminer un taux d'apprentissage idéal. Les taux traditionnellement utilisés sont les suivants :

0.001 , 0.003 , 0.01 , 0.03 , 0.1 et 0.3



https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Graphique de l'évolution de la précision du réseau en fonction du nombre de donnée analysé dans l'apprentissage.

L'algorithme de l'apprentissage

Afin que le réseau puisse réaliser un « apprentissage », il faut modifier l'ensemble de ses poids $w_{uv}^{(k)}$ et biais $b_u^{(k)}$ judicieusement afin de minimiser la fonction d'erreur C pour un ensemble de données d'entraînement. Si ces données sont préalablement connues[6], le réseau réaliser un « apprentissage supervisé ». Cette méthode itérative porte le nom de rétropropagation du gradient (b*ackpropagation*). et $b_u^{(k)}$

À partir du gradient de la fonction d'erreur ∇C, nous pouvons modifier les paramètres $w_{uv}^{(k)}$ et $b_u^{(k)}$ du réseau grâce aux équations suivantes :

Au réseau W	Poids $\tilde{w}_{vu}^{(k)}$	Biais $b_u^{(k)}$
$W^{\text{mod}} = W - \alpha \nabla C$	$\tilde{w}_{vu}^{\text{mod}(k)} = \tilde{w}_{vu}^{(k)} - \alpha \frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}}$	$b_u^{\text{mod}(k)} = b_u^{(k)} - \alpha \frac{\partial C}{\partial b_u^{(k)}}$

où

$\tilde{w}_{vu}^{\text{mod}(k)}$, $b_u^{\text{mod}(k)}$: Valeur modifiée d'un poids/biais (après apprentissage).

$\tilde{w}_{vu}^{(k)}$, $b_u^{(k)}$: Valeur d'un poids/biais à l'activation (avant apprentissage).

$\frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}}$, $\frac{\partial C}{\partial b_u^{(k)}}$: Gradient de la fonction d'erreur pour un poids/biais de couche k.

α : Le taux d'apprentissage (*learning rate*) $\alpha > 0$.

[6] Si les données d'entraînement ne sont pas connues, l'entraînement sera « non supervisé ». L'apprentissage par renforcement propose une technique pour générer des données d'entraînement dont la qualité augmente avec la progression du l'entraînement.

La propagation du gradient

Le gradient de la fonction d'erreur C noté ∇C par rapport aux paramètres de poids $\tilde{w}_{vu}^{(k)}$ et biais $b_u^{(k)}$ de toutes les couches du réseau dépend du choix de la fonction d'erreur, mais plus explicitement du choix et de l'ordre des fonctions à l'intérieur du réseau.

Dans un réseau simple[7], il n'y aura que des fonctions d'agrégation

$$z_u^{(k)} = z_u^{(k)}\left(a_v^{(k-1)},\ \tilde{w}_{vu}^{(k)},\ b_u^{(k)}\right)$$

et des fonctions d'activation à un neurone[8]

$$a_u^{(k)} = a_u^{(k)}\left(z_u^{(k)}\right)$$

dans l'ordre

$$a^{(*)} \to z^{(0)} \to a^{(0)} \to z^{(1)} \to a^{(1)} \to \ldots \to z^{(k)} \to a^{(k)} \to z^{(k+1)} \to a^{(k+1)} \to \ldots \to z^{(L)} \to a^{(L)}$$

où

$a^{(*)}$ sont les entrées du réseau et $a^{(L)}$ sont les sorties du réseau (l'activation du réseau).

En exploitant le concept mathématique de la différentielle d'une fonction et de la dérivée en chaîne

$$\frac{\mathrm{d}\,y}{\mathrm{d}x} = \frac{\mathrm{d}\,y}{\mathrm{d}u} \cdot \frac{\mathrm{d}\,u}{\mathrm{d}x}\ ,$$

nous obtenons l'expression suivante pour la différentielle de la fonction d'erreur :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \left(\begin{array}{l} \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial \tilde{w}_{ji}^{(L)}} \mathrm{d}\tilde{w}_{ji}^{(L)} + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} \mathrm{d}b_i^{(L)} \\ + \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}} \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} \left(\begin{array}{l} \displaystyle\sum_{k=0}^{N^{(L-2)}-1} \frac{\partial z_j^{(L-1)}}{\partial \tilde{w}_{kj}^{(L-1)}} \mathrm{d}\tilde{w}_{kj}^{(L-1)} + \frac{\partial z_j^{(L-1)}}{\partial b_j^{(L-1)}} \mathrm{d}b_j^{(L-1)} \\ + \displaystyle\sum_{k=0}^{N^{(L-2)}-1} \frac{\partial z_j^{(L-1)}}{\partial a_k^{(L-2)}} \mathrm{d}a_k^{(L-2)} \end{array} \right) \end{array} \right)$$

(Propagation du gradient de profondeur *n* = 1)

Cette lourde équation permet d'accéder aux équations nécessaires pour évaluer l'ensemble des termes

$$\frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} \quad \text{et} \quad \frac{\partial C}{\partial b_u^{(k)}}$$

permettant de connaître comment on doit modifier tous les paramètres $\tilde{w}_{vu}^{(k)}$ et $b_u^{(k)}$ à l'aide de l'équation de l'apprentissage

$$\tilde{w}_{vu}^{\mathrm{mod}(k)} = \tilde{w}_{vu}^{(k)} - \alpha \frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} \quad \text{et} \quad b_u^{\mathrm{mod}(k)} = b_u^{(k)} - \alpha \frac{\partial C}{\partial b_u^{(k)}}$$

du réseau pour que celui-ci converge vers la fonction que l'on désire reproduire. Pour bien suivre la séquence des modifications, nous procéderons par plusieurs étapes qui seront présentées dans la suite de ce texte.

---

[7] Un réseau avec normalisation aura une fonction supplémentaire à traité dans le calcul de la dérivée partielle de la fonction d'erreur.

[8] Une fonction d'activation à un neurone nécessite seulement une valeur au calcul comme la fonction sigmoïde.

Preuve :

Soit la fonction d'erreur $C = C\left(a_i^{(L)}, y_i\right)$, nous obtenons alors la différentielle suivante :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \mathrm{d}a_i^{(L)}$$

Avec une fonction d'activation $a_i^{(L)} = a_i^{(L)}\left(z_i^{(L)}\right)$ à un neurone telle que la différentielle est égale à

$$\mathrm{d}a_i^{(L)} = \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \mathrm{d}z_i^{(L)} \ ,$$

ceci qui donnera la différentielle suivante :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \mathrm{d}z_i^{(L)}$$

Avec la fonction d'agrégation $z_i^{(L)} = z_i^{(L)}\left(a_j^{(L-1)}, \tilde{w}_{ji}^{(L)}, b_i^{(L)}\right)$ telle que la différentielle est égale à

$$\mathrm{d}z_i^{(L)} = \sum_{j=0}^{N^{(L-1)}-1} \left(\frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}} \mathrm{d}a_j^{(L-1)} + \frac{\partial z_i^{(L)}}{\partial \tilde{w}_{ji}^{(L)}} \mathrm{d}\tilde{w}_{ji}^{(L)} \right) + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} \mathrm{d}b_i^{(L)}$$

ceci nous donnera la différentielle

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \left(\sum_{j=0}^{N^{(L-1)}-1} \left(\frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}} \mathrm{d}a_j^{(L-1)} + \frac{\partial z_i^{(L)}}{\partial \tilde{w}_{ji}^{(L)}} \mathrm{d}\tilde{w}_{ji}^{(L)} \right) + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} \mathrm{d}b_i^{(L)} \right)$$

que nous pouvons réécrire de la façon suivante :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \left(\begin{array}{l} \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial \tilde{w}_{ji}^{(L)}} \mathrm{d}\tilde{w}_{ji}^{(L)} + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} \mathrm{d}b_i^{(L)} \\ + \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}} \mathrm{d}a_j^{(L-1)} \end{array} \right)$$

(Propagation du gradient de niveau $n = 0$)

Si nous continuons à développer la différentielle au niveau $n = 1$, utilisons la fonction d'activation $a_j^{(L-1)} = a_j^{(L-1)}\left(z_j^{(L-1)}\right)$ telle que la différentielle est égale à

$$\mathrm{d}a_j^{(L-1)} = \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} \mathrm{d}z_j^{(L-1)}$$

et ceci nous donne l'expression suivante :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1} \frac{\partial C}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \left(\begin{array}{l} \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial \tilde{w}_{ji}^{(L)}} \mathrm{d}\tilde{w}_{ji}^{(L)} + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} \mathrm{d}b_i^{(L)} \\ + \displaystyle\sum_{j=0}^{N^{(L-1)}-1} \frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}} \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} \mathrm{d}z_j^{(L-1)} \end{array} \right)$$

Avec la fonction d'agrégation $z_j^{(L-1)} = z_j^{(L-1)}\left(a_k^{(L-2)}, \widetilde{w}_{kj}^{(L-1)}, b_j^{(L-1)}\right)$ telle que la différentielle est égale à

$$\mathrm{d}z_j^{(L-1)} = \sum_{k=0}^{N^{(L-2)}-1}\left(\frac{\partial z_j^{(L-1)}}{\partial a_k^{(L-2)}}\mathrm{d}a_k^{(L-2)} + \frac{\partial z_j^{(L-1)}}{\partial \widetilde{w}_{kj}^{(L-1)}}\mathrm{d}\widetilde{w}_{kj}^{(L-1)}\right) + \frac{\partial z_j^{(L-1)}}{\partial b_j^{(L-1)}}\mathrm{d}b_j^{(L-1)}$$

ceci nous donnera la différentielle suivante après réorganisation des termes :

$$\mathrm{d}C = \sum_{i=0}^{N^{(L)}-1}\frac{\partial C}{\partial a_i^{(L)}}\frac{\partial a_i^{(L)}}{\partial z_i^{(L)}}\left(\begin{array}{l}\displaystyle\sum_{j=0}^{N^{(L-1)}-1}\frac{\partial z_i^{(L)}}{\partial \widetilde{w}_{ji}^{(L)}}\mathrm{d}\widetilde{w}_{ji}^{(L)} + \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}}\mathrm{d}b_i^{(L)} \\ \\ \displaystyle+\sum_{j=0}^{N^{(L-1)}-1}\frac{\partial z_i^{(L)}}{\partial a_j^{(L-1)}}\frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}}\left(\begin{array}{l}\displaystyle\sum_{k=0}^{N^{(L-2)}-1}\frac{\partial z_j^{(L-1)}}{\partial \widetilde{w}_{kj}^{(L-1)}}\mathrm{d}\widetilde{w}_{kj}^{(L-1)} + \frac{\partial z_j^{(L-1)}}{\partial b_j^{(L-1)}}\mathrm{d}b_j^{(L-1)} \\ \\ \displaystyle+\sum_{k=0}^{N^{(L-2)}-1}\frac{\partial z_j^{(L-1)}}{\partial a_k^{(L-2)}}\mathrm{d}a_k^{(L-2)}\end{array}\right)\end{array}\right)$$

(Propagation du gradient de niveau n = 1)

En fixant les indices i et j, nous pouvons évaluer les coefficients devant les termes $\mathrm{d}\widetilde{w}_{ji}^{(L)}$ et $\mathrm{d}b_i^{(L)}$ ce qui correspondra aux dérivées partielles de la fonction d'erreur

$$\frac{\partial \mathrm{C}}{\partial \widetilde{w}_{ji}^{(L)}} \qquad \text{et} \qquad \frac{\partial \mathrm{C}}{\partial b_i^{(L)}}$$

par rapport aux paramètres de la couche L du réseau.

En effectuant la sommation sur l'indice i et en fixant l'indice j et k, nous pouvons évaluer les coefficients devant les termes $\mathrm{d}\widetilde{w}_{kj}^{(L-1)}$ et $\mathrm{d}b_j^{(L-1)}$ ce qui correspondra aux dérivées partielles de la fonction d'erreur

$$\frac{\partial \mathrm{C}}{\partial \widetilde{w}_{jk}^{(L-1)}} \qquad \text{et} \qquad \frac{\partial \mathrm{C}}{\partial b_j^{(L-1)}}$$

par rapport aux paramètres de la couche $L-1$ du réseau. ■

Les équations de la propagation du gradient

L'équation de la propagation de l'erreur de la fonction d'erreur

Lorsque l'on évalue le gradient de la fonction d'erreur C, nous allons devoir calculer comment une erreur évaluée sur la fonction d'erreur sera propagée à l'ensemble des paramètres du réseau.

Ainsi, en appliquant la différentielle à notre fonction d'erreur C, nous pouvons évaluer l'expression suivante pour évaluer l'erreur $\Delta_u^{C(L)}$ propagée par la fonction d'erreur C situé à la couche L du réseau pour le neurone de sortie u du réseau :

À la couche L
$\Delta_u^{C(L)} = \dfrac{\partial C}{\partial a_u^{(L)}}$ avec $u \in \left[0, N^{(L)} - 1\right]$

Preuve :

Soit la fonction d'erreur $C = C\left(a_u^{(L)}, y_u\right)$ où y_u est la valeur attendue que le réseau devrait générer pour $a_u^{(L)}$ avec l'activation du vecteur d'entrée $a^{(*)}$. Appliquons la différentielle à cette fonction ce qui nous donnera le terme de propagation de l'erreur de la fonction d'erreur pour le neurone u :

$$\mathrm{d}C = \mathrm{d}C\left(a_u^{(L)}, y_u\right) \quad \Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(L)}-1}\left(\frac{\partial C}{\partial a_u^{(L)}}\mathrm{d}a_u^{(L)} + \frac{\partial C}{\partial y_u}\mathrm{d}y_u\right)$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(L)}-1}\frac{\partial C}{\partial a_u^{(L)}}\mathrm{d}a_u^{(L)} \qquad (\mathrm{d}y_u = 0, \text{ car la valeur attendue ne varie pas})$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(L)}-1}\Delta_u^{C(L)}\mathrm{d}a_u^{(L)} \quad \blacksquare \qquad \left(\Delta_u^{C(L)} = \frac{\partial C}{\partial a_u^{(L)}}\right)$$

L'équation de la propagation de l'erreur de la fonction d'activation

Lorsque l'on évalue le gradient de la fonction d'erreur C, l'erreur propagée par la fonction d'erreur sera propagée à nouveau par la fonction d'activation, car celle-ci devra propager son erreur à la fonction d'agrégation qui elle aura une influence sur les poids $\tilde{w}_{vu}^{(k)}$ et biais $b_u^{(k)}$.

Ainsi, en appliquant la différentielle à notre fonction d'erreur C, nous pouvons évaluer l'expression suivante pour évaluer l'erreur $\Delta_u^{a(k)}$ propagée par la fonction d'activation $a^{(k)}$ situé à la couche k du réseau pour le neurone de sortie u de la couche k du réseau :

À la couche L	À la couche k
$\Delta_u^{a(L)} = \Delta_u^{C(L)}\dfrac{\partial a_u^{(k)}}{\partial z_u^{(k)}}$ avec $u \in \left[0, N^{(k)} - 1\right]$	$\Delta_u^{a(k)} = \Delta_u^{(k)}\dfrac{\partial a_u^{(k)}}{\partial z_u^{(k)}}$ avec $u \in \left[0, N^{(k)} - 1\right]$

Preuve :

Soit la différentielle de la fonction d'erreur C dont l'erreur propagée à la fonction d'activation $a_u^{(k)}$ de niveau de propagation $k = L - n$ est égale à l'expression

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}a_u^{(k)} \qquad \text{avec} \qquad \mathrm{d}a_u^{(k)} = \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}} \mathrm{d}z_u^{(k)}$$

où $\Delta_u^{(k)}$ est la somme des erreurs propagée depuis la fonction d'erreur vers l'intérieur du réseau pour l'ensemble des fonctions entre la fonction d'erreur C et la fonction d'activation $a_u^{(k)}$.

Le paramètre n est le niveau de profondeur dans la propagation de l'erreur dans le réseau. Il est à noter que $n = 0$ lorsque l'erreur est propagée par la dernière couche du réseau et $n = L$ lorsque l'erreur est propagée à la 1re couche du réseau.

Développons l'expression afin d'évaluer l'erreur qui sera propagée par la fonction $a_u^{(L)} = a_u^{(L)}\left(z_u^{(L)}\right)$ pour chaque neurone u de cette fonction à la fonction d'agrégation $z_u^{(k)}$:

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}a_u^{(k)} \quad \Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}} \mathrm{d}z_u^{(k)} \quad \left(\mathrm{d}a_u^{(L)} = \frac{\partial a_u^{(L)}}{\partial z_u^{(L)}} \mathrm{d}z_u^{(L)}\right)$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{a(k)} \mathrm{d}z_u^{(k)} \quad \blacksquare \qquad \left(\Delta_u^{a(k)} = \Delta_u^{(k)} \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}}\right)$$

L'équation de la propagation de l'erreur de la fonction d'activation à plusieurs neurones

Pour évaluer la propagation de l'erreur d'une fonction d'activation à plusieurs neurones $a_u^{full(k)} = f_u^{full(k)}\left(\bar{z}^{(k)}\right)$, il ne suffit que d'effectuer une sommation sur l'ensemble des neurones sous la forme suivante puisque toutes les valeurs d'activation ont été influencées par toutes les agrégations $z_u^{(k)}$:

À la couche L	À la couche k
$\Delta_u^{C-full(k)} = \sum_{p=0}^{N^{(k)}-1} \Delta_p^{C(L)} \frac{\partial a_p^{(k)}}{\partial z_u^{(k)}}$ avec $u, p \in \left[0, N^{(k)} - 1\right]$	$\Delta_u^{a-full(k)} = \sum_{p=0}^{N^{(k)}-1} \Delta_p^{(k)} \frac{\partial a_p^{(k)}}{\partial z_u^{(k)}}$ avec $u, p \in \left[0, N^{(k)} - 1\right]$

Preuve :

Soit la différentielle de la fonction d'erreur C dont l'erreur propagée à la fonction d'activation

$$a_u^{full(k)} = a_u^{full(k)}\left(z_0^{(k)}, z_1^{(k)}, \ldots, z_p^{(k)}, \ldots, z_{N^{(k)}-1}^{(k)}\right)$$

de niveau de propagation $k = L - n$ est égale à l'expression

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}a_u^{full(k)} \quad \text{où} \qquad \mathrm{d}a_u^{full(k)} = \sum_{p=0}^{N^{(k)}-1} \frac{\partial a_u^{full(k)}}{\partial z_p^{(k)}} \mathrm{d}z_p^{(k)}$$

correspond à la différentielle de la fonction d'activation à plusieurs neurones.

Développons l'expression afin d'évaluer l'erreur qui sera propagée par la fonction $a_u^{full(k)}$ pour chaque neurone u de cette fonction à la fonction d'agrégation $z_u^{(k)}$:

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}a_u^{(k)} \quad \Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \sum_{p=0}^{N^{(k)}-1} \frac{\partial a_u^{(k)}}{\partial z_p^{(k)}} \mathrm{d}z_p^{(k)} \quad \left(\mathrm{d}a_u^{full(k)} = \sum_{p=0}^{N^{(k)}-1} \frac{\partial a_u^{full(k)}}{\partial z_p^{(k)}} \mathrm{d}z_p^{(k)}\right)$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{p=0}^{N^{(k)}-1} \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial a_u^{(k)}}{\partial z_p^{(k)}} \mathrm{d}z_p^{(k)} \quad \text{(Factoriser la sommation sur } p\text{)}$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \sum_{p=0}^{N^{(k)}-1} \Delta_p^{(k)} \frac{\partial a_p^{(k)}}{\partial z_u^{(k)}} \mathrm{d}z_u^{(k)} \quad \text{(Inverser indice } p \leftrightarrow u\text{)}$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{a-full(k)} \mathrm{d}z_u^{(k)} \quad \blacksquare \quad \left(\Delta_u^{a-full(k)} = \sum_{p=0}^{N^{(k)}-1} \Delta_p^{(k)} \frac{\partial a_p^{(k)}}{\partial z_u^{(k)}}\right)$$

L'équation de la propagation de l'erreur de la fonction d'agrégation

Pour évaluer la propagation de l'erreur d'une fonction d'agrégation $z_u^{(k)}$ et le gradient de la fonction d'erreur par rapport au poids et biais d'une couche, nous pouvons évaluer les expressions suivantes :

Propagation de l'erreur	Gradient d'un poids	Gradient d'un biais
$\Delta_v^{z(k)} = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}}$	$\frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} = \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}}$	$\frac{\partial C}{\partial b_u^{(k)}} = \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}}$

$$\text{où} \qquad u \in [0, N^{(k)} - 1] \qquad \text{et} \qquad v \in [0, N^{(k-1)} - 1]$$

Preuve :

Soit la différentielle de la fonction d'erreur C dont l'erreur propagée à la fonction d'agrégation $z_u^{(k)}$ de niveau de propagation $k = L - n$ est égale à l'expression

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}z_u^{(k)} \qquad \text{où} \qquad \mathrm{d}z_u^{(k)} = \sum_{v=0}^{N^{(k-1)}-1} \left(\frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}} \mathrm{d}a_v^{(k-1)} + \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} \right) + \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)}$$

Développons l'expression afin d'évaluer l'erreur qui sera propagée par la fonction $z_u^{(k)}$ pour chaque neurone u de cette fonction à la fonction d'activation d'entrée $a_v^{(k-1)}$ ayant un nombre de neurones v différents :

$$\mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \mathrm{d}z_u^{(k)}$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \left(\sum_{v=0}^{N^{(k-1)}-1} \left(\frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}} \mathrm{d}a_v^{(k-1)} + \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} \right) + \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)} \right)$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{u=0}^{N^{(k)}-1} \sum_{v=0}^{N^{(k-1)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} + \sum_{u=0}^{N^{(k)}-1} \sum_{v=0}^{N^{(k-1)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}} \mathrm{d}a_v^{(k-1)} + \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)}$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{v=0}^{N^{(k-1)}-1} \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} + \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)} + \sum_{v=0}^{N^{(k-1)}-1} \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}} \mathrm{d}a_v^{(k-1)}$$

(réorg. et inv. ordre sommation)

$$\Rightarrow \quad \mathrm{d}C = \sum_{v=0}^{N^{(k-1)}-1} \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} + \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)} + \sum_{v=0}^{N^{(k-1)}-1} \Delta_v^{z(k)} \mathrm{d}a_v^{(k-1)}$$

$$\left(\Delta_v^{z(k)} = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}} \right)$$

$$\Rightarrow \quad \mathrm{d}C = \sum_{v=0}^{N^{(k-1)}-1} \sum_{u=0}^{N^{(k)}-1} \frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} \mathrm{d}\tilde{w}_{vu}^{(k)} + \sum_{u=0}^{N^{(k)}-1} \frac{\partial C}{\partial b_u^{(k)}} \mathrm{d}b_u^{(L)} + \sum_{v=0}^{N^{(k-1)}-1} \Delta_v^{z(k)} \mathrm{d}a_v^{(k-1)} \quad \blacksquare$$

$$\left(\frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} = \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}} \text{ et } \frac{\partial C}{\partial b_u^{(k)}} = \Delta_u^{(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}} \right)$$

Le tableau récapitulatif des équations de la propagation du gradient

Afin d'évaluer les dérivées partielles

$$\frac{\partial \mathrm{C}}{\partial \tilde{w}_{vu}^{(k)}} \qquad \text{et} \qquad \frac{\partial \mathrm{C}}{\partial b_u^{(k)}}$$

de la fonction d'erreur par rapport à tous les poids et biais de toutes les couches dans une séquence d'activation tel que

$$a^{(*)} \to z^{(0)} \to a^{(0)} \to z^{(1)} \to a^{(1)} \to \ldots \to z^{(k)} \to a^{(k)} \to z^{(k+1)} \to a^{(k+1)} \to \ldots \to z^{(L)} \to a^{(L)},$$

nous devons calculer efficacement la différentielle de la fonction d'erreur.

En exploitant le concept de propagation de l'erreur $\Delta_u^{f(k)}$ de chacune des fonctions du réseau dans l'ordre inverse de l'activation, nous obtenons la séquence

$$C \to a^{(L)} \to z^{(L)} \to \ldots \to a^{(k+1)} \to z^{(k+1)} \to a^{(k)} \to z^{(k)} \to \ldots \to a^{(1)} \to z^{(1)} \to a^{(0)} \to z^{(0)} \to \Delta^{(*)}$$

les équations suivantes :

Propagation de l'erreur $\Delta_u^{f(k)}$			
Propagation de niveau $n=0$	Propagation : $C^{(L)}$ tel $C^{(L)} \to a^{(L)}$	Propagation : $a^{(L)}$ tel $a^{(L)} \to z^{(L)}$	Indice des neurones
$L \to L-1$ (propagation de couche)	$\Delta_u^{C(L)} = \frac{\partial C}{\partial a_u^{(L)}}$	$\Delta_u^{a(L)} = \Delta_u^{C(L)} \frac{\partial a_u^{(L)}}{\partial z_u^{(L)}}$	$u \in [0, N^{(L)}-1]$
	P.S La fonction d'erreur possède le même nombre de neurones que la couche *L*	$\Delta_u^{a-full(L)} = \sum_{p=0}^{N^{(L)}-1} \Delta_p^{C(L)} \frac{\partial a_p^{(L)}}{\partial z_u^{(L)}}$	
Propagation de niveau n où $k=L-n$	Propagation : $z^{(k)}$ tel $z^{(k)} \to a^{(k-1)}$	Propagation : $a^{(k)}$ tel $a^{(k)} \to z^{(k)}$	Indice des neurones
$k \to k-1$ (propagation de couche)	$\Delta_v^{z(k)} = \sum_{u=0}^{N^{(k)}-1} \Delta_u^{a(k)} \frac{\partial z_u^{(k)}}{\partial a_v^{(k-1)}}$	$\Delta_u^{a(k)} = \Delta_u^{z(k+1)} \frac{\partial a_u^{(k)}}{\partial z_u^{(k)}}$	$u \in [0, N^{(k)}-1]$
	P.S Puisqu'il y a changement de couche, il y a changement du nombre de neurones passant de *u* à *v*.	$\Delta_u^{a-full(k)} = \sum_{p=0}^{N^{(k)}-1} \Delta_p^{z(k+1)} \frac{\partial a_p^{(k)}}{\partial z_u^{(k)}}$	$v \in [0, N^{(k-1)}-1]$

Gradient de la fonction d'erreur par rapport à aux paramètres $\tilde{w}_{vu}^{(k)}$ et $b_u^{(k)}$ du réseau	
$\frac{\partial C}{\partial \tilde{w}_{vu}^{(k)}} = \Delta_u^{a(k)} \frac{\partial z_u^{(k)}}{\partial \tilde{w}_{vu}^{(k)}}$	$\frac{\partial C}{\partial b_u^{(k)}} = \Delta_u^{a(k)} \frac{\partial z_u^{(k)}}{\partial b_u^{(k)}}$

Appendice

Les notations

Voici une liste des notations utilisées lors de la présentation des concepts du réseau de neurone :

$*$: Indice de la couche d'entrée.

k : Indice d'une couche de neurones. La première couche début à $k=0$.

L : Indice de la dernière couche du réseau de neurones ($k \in [0, L]$). Puisque la numérotation en informatique des couches débute à $k=0$, un réseau aura alors $L+1$ couches.

i, j : Indices utilisés pour désigner un neurone particulier d'une couche de neurones.

$a^{(k)}$: Les valeurs de tous les neurones de la couche k.

$a_i^{(k)}$: Valeur du neurone i de la couche k.

$N^{(k)}$: Le nombre de neurones de la couche k. Ainsi $a^{(k)} = \left\{a_0^{(k)}, a_1^{(k)}, \ldots, a_i^{(k)}, \ldots, a_{N^{(k)}-1}^{(k)}\right\}$

$z^{(k)}$: Les agrégation de de tous les neurones de la couche k.

$z_i^{(k)}$: L'agrégation du neurone i de la couche k. Ce calcul nécessite des valeurs de neurones $a_j^{(k-1)}$ d'une couche précédente $k-1$, des poids $w_{ij}^{(k)}$ et d'un biais $b_i^{(k)}$.

$w^{(k)}$: Les poids de tous les neurones de la couche k.

$w_{ij}^{(k)}$: Le poids du neurone $a_j^{(k-1)}$ dans l'agrégation $z_i^{(k)}$ du neurone i de la couche k.

$b^{(k)}$: Les biais de tous les neurones de la couche k.

$b_i^{(k)}$: Le biais dans l'agrégation $z_i^{(k)}$ du neurone i de la couche k.

C : La fonction d'erreur.

n : Indice du niveau de profondeur dans la descente du gradient. On utilisera une notation tel que $a^{(L-n)}$ pour désigner la valeur des neurones en itérant à partir de la fin du réseau.

α : Le taux d'apprentissage $\alpha > 0$.