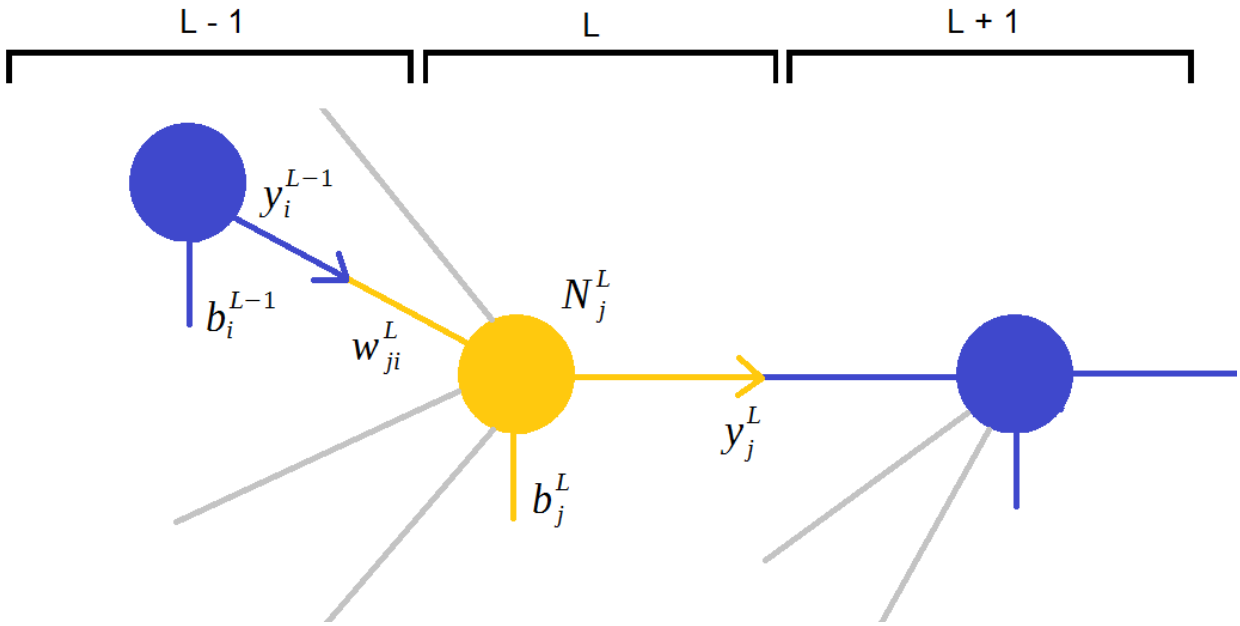
Les réseaux de neurones multicouches

AF. Michael

(v2)

Table des matières

Backpropagation du gradient pour un échantillon



Concentrons nous sur la couche L au niveaux du neurone indexé par j : N_j^L en jaune.

Comme le but est de **minimiser** l'erreur quadratique entre la réponse attendue et la réponse devinée par le réseaux, le gradient à étudier sera donc $\vec{\nabla} E$, ou plutôt son opposé $-\vec{\nabla} E$ dans notre cas.

*** Ainsi pour un exemple ou une donnée de l'ensemble des échantillons ....**

Cette erreur se calcule au niveau de la dernière couche et s'exprime par :

$$E = \sum_j (r_j - y_j^L)^2$$

r_j : réponse correcte attendue pour l'échantillon au neurone j pour l'exemple

y_j^L : activation du neurone j à la dernière couche (L représente l'indexe de la dernière couche)

r_j est constante tandis que y_j^L est variable et est une fonction des **poids** et **biais** de chaque neurone

$$y_j^L = f(\vec{W}_j^L . \vec{Y}^{L-1} + b_j^L) = f(\sum_i w_{ji}^L y_i^{L-1} + b_j^L)$$

On peut réécrire y_i^{L-1} de la même manière ce qui fait que l'erreur est en fonction des variables du réseau en entier (à savoir les poids et les biais en gros, et éventuellement la fonction d'activation f aussi).

E = **en fonction** (Tous les poids, Tous les biais, La fonction d'activation **f**)

Les seuls valeurs variables durant l'expérience restent les poids et les biais puisque la fonction d'activation **f** sera fixée en avance.

Comme « minimiser » c'est calculer l'opposé du gradient :

$$-\vec{\nabla} E\left(w_{00}^{0}, w_{01}^{0}, \ldots., w_{ji}^{L}, \ldots., b_{0}^{0}, \ldots, b_{j}^{L}, \ldots\right)$$

La fonction E est, au final ridiculement complexe, avec des centaines de variables à faire varier, une fonction qui se promène dans un espace de très haute dimension.

Heureusement ce gradient est facile à généraliser, il faut juste bien indexer les variables de manière à ne pas les confondre.

Pour minimiser il faut donc nous mettre dans une position aléatoire sur la graphe de E (ce qui se fait en initialisant au hasard les poids et biais du réseau) puis, pour chaque variable de E, les modifier dans la direction du gradient pour atteindre un **minimum local** ou **global** (si on est **très très chanceux**).

* Petit rappel : la minimisation d'une fonction à variable $\vec{V}$ sachant qu'on se trouve au point $M(\vec{V}_0, f(\vec{V}_0))$ se fait en se déplaçant d'une longueur proportionnelle au vecteur gradient local jusqu'à un nouveau point suivant la direction de ce même gradient. (et de sens opposé comme le but est de minimiser) ce qui se traduit par la suite définie par :

$\vec{V}_0 \leftarrow \vec{V}_0 - \alpha \vec{\nabla} f(\vec{V}_0)$ avec α un nombre réel permettant de régler la longueur du gradient et de contrôler le pas de déplacement.

Même histoire ici, pour simplifier on va casser le gradient en ses composantes au lieu de la notation vectorielle : les composantes seront donc les dérivées partielles par rapport aux poids et aux biais :

Pour tout neurone indexé j à la couche L, on a donc :

$$w_{ji}^{L} \leftarrow w_{ji}^{L} - \alpha \frac{\partial E}{\partial w_{ji}^{L}}$$ où le nombre i indexe les poids du neurone N_j^L

$$b_{j}^{L} \leftarrow b_{j}^{L} - \alpha \frac{\partial E}{\partial b_{j}^{L}}$$

Posons $\Delta w_{ji}^{L} = -\frac{\partial E}{\partial w_{ji}^{L}}$ et $\Delta b_{j}^{L} = -\frac{\partial E}{\partial b_{j}^{L}}$

La mise à jour des valeurs se réécrira donc :

$$w_{ji}^{L} \leftarrow w_{ji}^{L} + \alpha \Delta w_{ji}^{L}$$

$$b_{j}^{L} \leftarrow b_{j}^{L} + \alpha \Delta b_{j}^{L}$$

Il nous faut donc calculer les dérivées partielles $\frac{\partial E}{\partial w_{ji}^L}$ et $\frac{\partial E}{\partial b_j^L}$.

Petites conventions nécessaires aux calculs

- Erreur du réseau pour l'exemple actuel

$$E=\sum_j (r_j-y_j^L)^2$$

- Erreur de la couche de sortie

$$e_j^L=r_j-y_j^L$$

- Entrée du neurone j à la couche L

$$z_j^L=\sum_i w_{ji}^L y_i^{L-1}+b_j^L$$

- Activation du neurone j à la couche L

$$y_j^L=f(z_j^L)$$

Pour la couche de sortie

*** Pour les poids de la couche de sortie :**

1.0- $$\frac{\partial E}{\partial w_{ji}^L}=\frac{\partial E}{\partial e_j^L}\frac{\partial e_j^L}{\partial w_{ji}^L}=\frac{\partial E}{\partial e_j^L}\frac{\partial e_j^L}{\partial z_j^L}\frac{\partial z_j^L}{\partial w_{ji}^L}$$

1.1- $$\frac{\partial E}{\partial e_j^L}=\frac{\partial}{\partial e_j^L}\sum_k (e_k^L)^2=2e_j^L$$

1.2-

$$\frac{\partial e_j^L}{\partial z_j^L}=\frac{\partial}{\partial z_j^L}(r_j-y_j^L)=\frac{\partial}{\partial z_j^L}(r_j-f(z_j^L=\sum_i w_{ji}^L y_i^{L-1}+b_j^L))$$

$$\frac{\partial e_j^L}{\partial z_j^L}=-f'(z_j^L)$$

1.3- $$\frac{\partial z_j^L}{\partial w_{ji}^L}=\frac{\partial}{\partial w_{ji}^L}(\sum_i w_{ji}^L y_i^{L-1}+b_j^L)=y_i^{L-1}$$

1.4- $$\frac{\partial E}{\partial w_{ji}^L} = -2 e_j^L f'(z_j^L) y_i^{L-1}$$

*** Pour les biais de la couche de sortie :**

De même pour les biais, on fait la même procédure...

$$\frac{\partial E}{\partial b_j^L} = \frac{\partial E}{\partial e_j^L}\frac{\partial e_j^L}{\partial b_j^L} = \frac{\partial E}{\partial e_j^L}\frac{\partial e_j^L}{\partial z_j^L}\frac{\partial z_j^L}{\partial b_j^L}$$

Heureusement il ne nous reste qu'à déterminer la dernière dérivée partielle

1.5- $$\frac{\partial z_j^L}{\partial b_{ji}^L} = \frac{\partial}{\partial b_j^L}\left(\sum_i w_{ji}^L y_i^{L-1} + b_j^L\right) = 1$$

1.6- $$\frac{\partial E}{\partial b_j^L} = -2 e_j^L f'(z_j^L)$$

Posons pour la suite

1.7- $$\delta_j^L = e_j^L f'(z_j^L)$$

Ce qui nous permet de réécrire tout plus proprement

1.8- $$\frac{\partial E}{\partial w_{ji}^L} = -2 \delta_j^L y_i^{L-1}$$ pour les poids

1.9- $$\frac{\partial E}{\partial b_j^L} = -2 \delta_j^L$$ pour les biais

INFO : Généralement, on définie f comme étant égale à la fonction sigmoïde.

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

donc

$$f'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{(e^{-z}+1-1)}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} - \frac{1}{(1+e^{-z})^2} = (1-f(z))f(z)$$

ce qui permet faire une redéfinition, d'après l'équation 1.7

$$\delta_j^L = e_j^L (1 - f(z_j^L)) f(z_j^L) \quad \text{or} \quad y_j^L = f(z_j^L)$$

d'où

$$\delta_j^L = e_j^L (1 - y_j^L) y_j^L$$

C'est souvent cette expression que l'on retrouve dans les livres qui abordent le sujet des réseaux de neurones.

Pour être le plus général possible, la fonction f définie initialement comme étant une fonction quelconque continue et dérivable ne changera pas. (on gardera l'équation 1.7 comme base)

Pour les couches cachées

*** Pour les poids**

Rien ne change ! On décompose les dérivées comme d'habitude. Cependant l'objectif actuel va être de trouver un moyen pour que les variations de la couche L soient définies en fonction de la couche suivante L+1 (d'où le terme **backpropagation** comme il faut faire des calculs au niveau de L+1 pour pouvoir déterminer/exprimer les calculs au niveau de L) car sinon on tournera en rond et obtiendra les mêmes résultats que celles de la couche de sortie.

$$\frac{\partial E}{\partial w_{ji}^L} = \frac{\partial E}{\partial e_j^L} \frac{\partial e_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{ji}^L}$$

* La seule chose à remarquer peut-être c'est le changement de la **définition** de l'**erreur locale** e_j^L à la couche L comme l'erreur sur les couches cachées sont différentes par rapport à la couche de sortie, c'est d'ailleurs la seule clé pour exprimer les variations de la couche L en fonction de la couche voisine. Avec un petit tour de passe passe sur le chaînage des dérivées, on peut s'en sortir.

Sachant que $\frac{\partial E}{\partial e_j^L} = \frac{\partial E}{\partial y_j^L} \frac{\partial y_j^L}{\partial e_j^L}$

On peut réécrire la décomposition de départ

$$\frac{\partial E}{\partial w_{ji}^L} = \frac{\partial E}{\partial e_j^L} \frac{\partial e_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{ji}^L} = \left(\frac{\partial E}{\partial y_j^L} \frac{\partial y_j^L}{\partial e_j^L} \right) \frac{\partial e_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{ji}^L}$$

donc

2.0- $$\frac{\partial E}{\partial w_{ji}^L} = \frac{\partial E}{\partial y_j^L} \frac{\partial y_j^L}{\partial e_j^L} \frac{\partial e_j^L}{\partial w_{ji}^L}$$

C'est exactement la même quantité que tout à l'heure le changement d'expression permet juste de **découvrir** quelques expressions qui nous seront utiles pour calculer les gradients au niveau des couches cachées.

$$E=\sum_k (e_k^{L+1})^2$$

Cette expression illustre juste le fait que l'erreur d'une couche cachée L dépend de la couche suivante L+1 : on ne peut calculer l'erreur au niveau de L sans avoir déjà calculé l'erreur au niveau de L+1 (**backpropagation** de l'erreur)

$$\frac{\partial E}{\partial y_j^L}=\frac{\partial}{\partial y_j^L}\sum_k (e_k^{L+1})^2=\sum_k 2\,e_k^{L+1}\frac{\partial e_k^{L+1}}{\partial y_j^L}$$

$$\frac{\partial E}{\partial y_j^L}=\sum_k 2\,e_k^{L+1}\frac{\partial e_k^{L+1}}{\partial y_j^L}=\sum_k 2\,e_k^{L+1}\frac{\partial e_k^{L+1}}{\partial z_k^{L+1}}\frac{\partial z_k^{L+1}}{\partial y_j^L}$$

$$=\sum_k 2\,e_k^{L+1}\left[\frac{\partial}{\partial z_k^{L+1}}\left(r_j-f\left(z_k^{L+1}\right)\right)\right]\left[\frac{\partial}{\partial y_j^L}\sum_q w_{k\,q}^{L+1}\,y_q^L+b_k^{L+1}\right]$$

$$=\sum_k 2\,e_k^{L+1}\left[-f'\left(z_k^{L+1}\right)\right]\left[w_{k\,j}^{L+1}\right]$$

$$=\sum_k \left[-2\,e_k^{L+1}f'\left(z_k^{L+1}\right)\right]\left[w_{k\,j}^{L+1}\right]$$

d'où

2.1- $$\frac{\partial E}{\partial y_j^L}=-2\sum_k \delta_k^{L+1}\,w_{k\,j}^{L+1}$$

Pour uniformiser les choses plus proprement et de manière conforme aux résultats obtenus à la couche de sortie, définissons l'erreur locale par

2.2- $$e_j^L=\sum_k \delta_k^{L+1}\,w_{k\,j}^{L+1}$$

qui est proportionnelle mais de signe opposé au taux de variation de l'erreur de la couche de sortie par rapport à l'activation du neurone j à la couche cachée L.

Le reste se calcule de façon triviale :

2.3- $$\frac{\partial y_j^L}{\partial e_j^L}=\frac{\partial}{\partial e_j^L}\left(-e_j^L+r_j\right)=-1$$

2.4- $$\frac{\partial e_j^L}{\partial w_{ji}^L}=\frac{\partial e_j^L}{\partial z_j^L}\frac{\partial z_j^L}{\partial w_{ji}^L}=-f'\left(z_j^L\right)\frac{\partial}{\partial w_{ji}^L}\left[\sum_i w_{j\,i}^L\,y_i^{L-1}+b_i^L\right]=-f'\left(z_j^L\right)y_i^{L-1}$$

*** Pour les biais c'est pareil... (en zappant les étapes, le principe reste le même)**

2.5- $$\frac{\partial E}{\partial b_j^L}=\frac{\partial E}{\partial y_j^L}\frac{\partial y_j^L}{\partial e_j^L}\frac{\partial e_j^L}{\partial b_j^L}$$

La seule différence réside dans la dernière dérivée partielle comme on connaît déjà les autres

2.6- $$\frac{\partial e_j^L}{\partial b_j^L}=\frac{\partial e_j^L}{\partial z_j^L}\frac{\partial z_j^L}{\partial b_j^L}=-f'(z_j^L)\frac{\partial}{\partial b_j^L}[\sum_i w_{j\,i}^L y_i^{L-1}+b_i^L]=-f'(z_j^L)$$

Ce qui nous donne au final :

2.7- $$\frac{\partial E}{\partial w_{ji}^L}=\frac{\partial E}{\partial y_j^L}\frac{\partial y_j^L}{\partial e_j^L}\frac{\partial e_j^L}{\partial w_{ji}^L}=-2f'(z_j^L)\,y_i^{L-1}\sum_k \delta_k^{L+1} w_{k\,j}^{L+1}$$

c.à.d

$$\frac{\partial E}{\partial w_{ji}^L}=-2\,\delta_j^L\,y_i^{L-1}$$ pour les poids (d'après **2.2, 2.7** et **1.7**)

2.8- $$\frac{\partial E}{\partial b_j^L}=\frac{\partial E}{\partial y_j^L}\frac{\partial y_j^L}{\partial e_j^L}\frac{\partial e_j^L}{\partial b_j^L}=-2f'(z_j^L)\sum_k \delta_k^{L+1} w_{k\,j}^{L+1}$$

c.à.d

$$\frac{\partial E}{\partial b_j^L}=-2\,\delta_j^L$$ pour les biais (d'après **2.2, 2.8** et **1.7**)

<u>Récapitulatif :</u>

Ce récapitulatif est ordonné dans l'ordre de ce qu'il faut calculer en premier.
<u>Rappel</u> :Toutes ces formules sont valides pour l'erreur quadratique uniquement.

(Sachant que l'algorithme commence à l'indexe de la dernière couche (celle de sortie) jusqu'au premier)

Pour une donnée d'exemple on a …

- Différences

Couche de sortie	Couches cachées
$e_j^L = r_j - y_j^L$	$e_j^L = \sum_k \delta_k^{L+1} w_{kj}^{L+1}$

- Points communs

$$\delta_j^L = e_j^L f'(z_j^L)$$

$$\frac{\partial E}{\partial w_{ji}^L} = -2\,\delta_j^L\, y_i^{L-1}$$

$$\frac{\partial E}{\partial b_j^L} = -2\,\delta_j^L$$

$$\Delta w_{ji}^L = -\frac{\partial E}{\partial w_{ji}^L} = 2\,\delta_j^L\, y_i^{L-1}$$

$$w_{ji}^L \leftarrow w_{ji}^L + \alpha\,\Delta w_{ji}^L$$

$$\Delta b_j^L = -\frac{\partial E}{\partial b_j^L} = 2\,\delta_j^L$$

$$b_j^L \leftarrow b_j^L + \alpha\,\Delta b_j^L$$

Remarques :

Comme α agit comme un facteur d'échelle, on peut omettre le nombre 2 et utiliser

$$\Delta w_{ji}^L = \delta_j^L\, y_i^{L-1} \qquad \Delta b_j^L = \delta_j^L$$

Notes sur le codage pratique

1-Conception

Il est idéal de **préparer** des tableaux ou variables pour stocker

- tous les deltas de chaque neurone $\delta_j^L = e_j^L f'(z_j^L)$
- toutes les sorties de chaque neurone y_j^L
- toutes les entrés de chaque neurone pour calculer $z_j^L = \sum_i w_{ji}^L y_i^{L-1} + b_j^L$ ou stocker directement z_j^L : ce qui est utile pour le calcul de $f'(z_j^L)$
- les erreurs e_j^L (facultatif, c'est uniquement utile pour les statistiques et l'affichage de la progression de l'apprentissage)
- les variations Δ (facultatif, la raison est la même que le stockage des erreurs)

2-Développement et algorithme

Pour une étape d'apprentissage, l'algorithme est

*** Pour un exemple X de l'ensemble des échantillons …**

- Calculer la sortie du réseaux pour X en sauvegardant l'état de chaque neurone durant la propagation vers l'avant. (c.a.d sauvegarder y_j^L et z_j^L pour chaque neurone)
- Dans une boucle parcourant les couches de **l'arrière** vers **l'avant** (**backpropagation**), on calcule les e_j^L et les $\delta_j^L = e_j^L f'(z_j^L)$ dans cette boucle tout en les stockant.
- Dans une boucle parcourant les couches de **l'avant** à **l'arrière**, mettre à jour les poids et les biais

*** Faire de même pour un autre exemple**

Poursuivre jusqu'à ce que le nombre d'étape d'apprentissage fixé soit terminé ou que l'on ait atteint l'erreur minimale voulue (ce dernier peut se faire uniquement dans le cas où les erreurs du réseau ont été stockées)