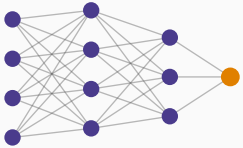
Apprentissage profond

Chapitre 5 : La rétropropagation



Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Le problème à résoudre

Un million de dérivées

Ce que le chapitre 4 a supposé

La descente de gradient exige $\partial L / \partial w$ pour **chaque** paramètre.

Calculer chacune séparément, en perturbant le paramètre et en refaisant une passe complète, coûterait autant de passes que de paramètres. Pour un million de poids, c'est impossible.

La **rétropropagation** obtient toutes les dérivées en **une seule** passe supplémentaire.

Le principe

C'est la **règle de dérivation en chaîne**, appliquée systématiquement, de la sortie vers l'entrée, en réutilisant les calculs intermédiaires.

Application en gestion

La règle de la chaîne y était illustrée par deux engrenages : si y dépend de u qui dépend de x, alors

$$\frac{dy}{dx} = \frac{dy}{du} \times \frac{du}{dx}$$

Un réseau profond est une composition de dizaines de fonctions. La rétropropagation applique cette règle de proche en proche, sans rien calculer deux fois.

Les deux passes

Avant, puis arrière

La passe avant

De l'entrée vers la sortie : calculer $z^{(\ell)}$ et $a^{(\ell)}$ couche par couche, jusqu'à la prédiction, puis la perte.

On conserve toutes les valeurs intermédiaires — elles seront nécessaires au retour.

La passe arrière

De la sortie vers l'entrée : propager l'erreur.

1. Calculer l'erreur en sortie : $\delta^{(L)} = \partial L / \partial z^{(L)}$.
2. Pour chaque couche en remontant :

$$\delta^{(\ell)} = \left(W^{(\ell+1)}\right)^{\mathsf{T}} \delta^{(\ell+1)} \odot g'\left(z^{(\ell)}\right)$$

3. En déduire les gradients :

$$\frac{\partial L}{\partial W^{(\ell)}} = \delta^{(\ell)} \left(a^{(\ell-1)}\right)^{\mathsf{T}} \qquad \frac{\partial L}{\partial b^{(\ell)}} = \delta^{(\ell)}$$

Ce que dit la formule de récurrence

Intuition

Trois opérations, à chaque couche :

- **transposer et multiplier** par les poids — l'erreur remonte par les mêmes connexions qu'à l'aller ;
- **multiplier par** $g'(z)$ — un neurone saturé ne transmet pas d'erreur ;
- le gradient d'un poids est le produit de l'**erreur en aval** par l'**activation en amont**.

Cette dernière règle est la plus parlante : un poids est corrigé proportionnellement à ce qu'il a reçu et à l'erreur qu'il a causée.

La mémoire, coût caché de la profondeur

Application en gestion

Conserver toutes les activations intermédiaires occupe une mémoire proportionnelle à la profondeur **et** à la taille du lot.

C'est ce qui limite en pratique la taille des lots sur une carte graphique — bien plus que le temps de calcul. Réduire le lot est le premier remède à un dépassement de mémoire.

Le gradient qui s’évanouit, expliqué

Pourquoi la profondeur pose problème

Le produit de la récurrence

Remonter L couches multiplie L facteurs, chacun contenant $g'(z)$ et les poids.

Si ces facteurs sont **inférieurs à un**, le produit tend vers zéro : gradient **évanoui**. S'ils sont supérieurs à un, il explose.

Le chiffrage du chapitre 3, replacé

Avec la sigmoïde, $g' \leq 0{,}25$:

Couches	Facteur au mieux
5	$9{,}8 \times 10^{-4}$
10	$9{,}5 \times 10^{-7}$
20	$9{,}1 \times 10^{-13}$

Les premières couches ne reçoivent plus rien : elles cessent d'apprendre.

Définition

Remède	Chapitre
ReLU — dérivée de 1	3
Initialisation adaptée — facteurs proches de 1	7
Normalisation par lot	7
Connexions résiduelles — un chemin direct pour le gradient	8

Le gradient qui explose, et son remède

Intuition

Le problème inverse existe, surtout dans les réseaux récurrents du chapitre 9 : le produit diverge, les poids partent à l'infini, la perte devient indéfinie.

Le remède est brutal et efficace : **l'écrêtage du gradient**, qui plafonne sa norme avant la mise à jour.

La différentiation automatique

Ce que font les bibliothèques

Le graphe de calcul

Chaque opération d'un réseau est un nœud d'un graphe. La passe avant le construit ; la passe arrière le parcourt en sens inverse, en appliquant la règle de la chaîne à chaque nœud.

C'est la **différentiation automatique**, et c'est ce qui rend inutile tout calcul manuel de dérivée.

Pourquoi l'étudier malgré tout

Aucun praticien n'écrit une rétropropagation à la main. Mais on ne diagnostique pas ce qu'on ne comprend pas :

- un gradient nul signale une saturation ou un neurone mort ;
- un gradient explosif signale un taux ou une initialisation inadaptés ;
- une perte indéfinie signale une division par zéro ou un logarithme de zéro.

Chacun de ces symptômes se lit dans les formules de ce chapitre. C'est à cela qu'il sert.

Ce qu’il faut retenir

Le résumé du chapitre

Cinq points

1. La rétropropagation obtient **tous** les gradients en une seule passe arrière.
2. C'est la **règle de la chaîne** appliquée systématiquement, avec réutilisation des calculs.
3. Passe **avant** : conserver les activations. Passe **arrière** : propager l'erreur δ.
4. Le gradient d'un poids est le produit de l'**erreur en aval** par l'**activation en amont**.
5. Le produit des facteurs explique l'**évanouissement** et l'**explosion** du gradient.

La suite

Le réseau sait maintenant apprendre. Le chapitre 6 traite du problème inverse : l'empêcher d'apprendre **trop bien** les données d'entraînement.