Analyse Mathématique Complète du Processus d'Apprentissage

S. Jaubert

20 novembre 2025

Résumé

Ce document fournit une explication mathématique approfondie du processus d'apprentissage d'un réseau de neurones. Il détaille le formalisme de la propagation avant, analyse diverses fonctions d'activation et de perte, explicite les calculs sous-jacents à l'algorithme de rétropropagation du gradient et met ce processus en contexte en abordant la validation et le phénomène de surapprentissage.

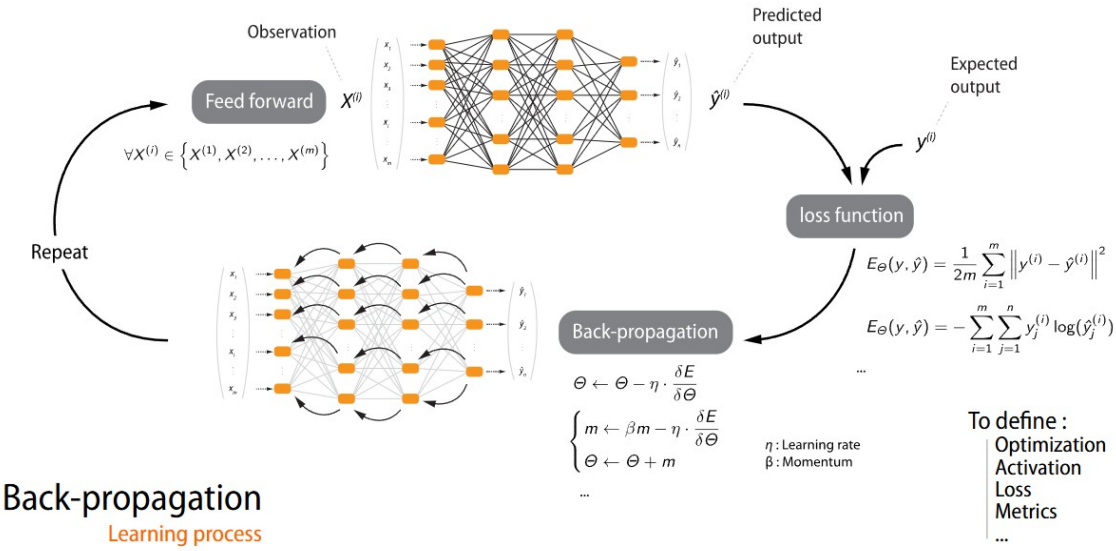

FIGURE 1 – Schéma global du processus d'apprentissage par rétropropagation du gradient.

1 Phase 1 : Propagation Avant (Feedforward)

La propagation avant est le processus par lequel une entrée X est transformée en une sortie prédite $\hat{y}$ en traversant le réseau couche par couche.

1.1 Formalisme Mathématique

Soit un réseau de neurones à L couches. Pour une couche $l \in \{1, ..., L\}$:
— $W^{[l]}$ est la matrice des poids, $b^{[l]}$ est le vecteur des biais.
— $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ est le vecteur des sorties pondérées (pré-activation).
— $A^{[l]} = g^{[l]}(Z^{[l]})$ est le vecteur des activations.
Avec $A^{[0]} = X$, la prédiction finale est $\hat{y} = A^{[L]}$.

1.2 Fonctions d'Activation Possibles ($g(z)$)

1.2.1 Fonction Sigmoïde

- **Formule** : $\sigma(z) = \frac{1}{1+e^{-z}}$
- **Domaine de sortie** : $(0, 1)$.
- **Dérivée** : $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- **Usage** : Principalement en sortie pour la classification binaire.
- **Inconvénient** : Sujette au problème de l'évanouissement du gradient (vanishing gradient).

1.2.2 Fonction Tangente Hyperbolique (Tanh)

- **Formule** : $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- **Domaine de sortie** : $(-1, 1)$. Centrée en 0, ce qui peut accélérer la convergence.
- **Dérivée** : $\tanh'(z) = 1 - \tanh^2(z)$
- **Inconvénient** : Souffre également du problème d'évanouissement du gradient.

1.2.3 Unité de Linéarité Rectifiée (ReLU)

- **Formule** : $g(z) = \max(0, z)$
- **Domaine de sortie** : $[0, +\infty)$.
- **Dérivée** : $g'(z) = \begin{cases} 1 & \text{si } z > 0 \\ 0 & \text{si } z \leq 0 \end{cases}$
- **Avantages** : La plus utilisée dans les couches cachées. Très efficace contre l'évanouissement du gradient.

2 Phase 2 : Fonctions de Perte (Loss Functions)

2.1 Pour les problèmes de Régression

- **Erreur Quadratique Moyenne (MSE)** : $E_{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$
- **Erreur Absolue Moyenne (MAE)** : $E_{MAE} = \frac{1}{m} \sum_{i=1}^{m} |y^{(i)} - \hat{y}^{(i)}|$

2.2 Pour les problèmes de Classification

- **Entropie Croisée Binaire (BCE)** : $E_{BCE} = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$
- **Entropie Croisée Catégorielle (CCE)** : $E_{CCE} = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)})$

3 Phase 3 : Rétropropagation et Optimisation

3.1 Le Calcul du Gradient : Règle de Dérivation en Chaîne

On minimise E en calculant $\frac{\partial E}{\partial W^{[l]}}$ et $\frac{\partial E}{\partial b^{[l]}}$. Pour cela, on définit l'erreur à la pré-activation $\delta^{[l]} = \frac{\partial E}{\partial Z^{[l]}}$.

1. Erreur à la couche de sortie (L) : $\delta^{[L]} = \frac{\partial E}{\partial A^{[L]}} \frac{\partial A^{[L]}}{\partial Z^{[L]}} = \nabla_{A^{[L]}} E \odot g^{[L]'}(Z^{[L]})$, où $\odot$ est le produit de Hadamard.

2. Rétropropagation de l'erreur : L'erreur se propage à l'envers, de la couche $l + 1$ à la couche l :

$$\delta^{[l]} = \left((W^{[l+1]})^T \delta^{[l+1]} \right) \odot g^{[l]'}(Z^{[l]})$$

3. Gradient des paramètres :

$$\frac{\partial E}{\partial W^{[l]}} = \delta^{[l]}(A^{[l-1]})^T \quad \text{et} \quad \frac{\partial E}{\partial b^{[l]}} = \delta^{[l]}$$

3.2 La Boucle d'Apprentissage et les Variantes de la Descente de Gradient

La mise à jour $\Theta \leftarrow \Theta - \eta \nabla_\Theta E$ est effectuée par itérations sur des **mini-batchs**, qui sont le meilleur compromis entre la **descente de gradient stochastique** (un seul exemple) et la **descente par lots** (tout le dataset).

4 Définitions Étendues des Termes Clés

Optimization Processus algorithmique visant à trouver les paramètres qui minimisent la perte. Des optimiseurs adaptatifs comme **Adam** ou **RMSprop** ajustent dynamiquement le taux d'apprentissage.

Activation Function Fonction non-linéaire essentielle qui permet au réseau d'apprendre des relations complexes.

Loss Function Mesure quantitative et différentiable de l'erreur du modèle, servant de signal pour l'optimisation.

Metrics Mesures de performance interprétables (accuracy, F1-score, etc.) pour évaluer la qualité finale du modèle.

Hyperparamètres Paramètres fixés avant l'entraînement (taux d'apprentissage η, nombre de couches, taille des batchs, etc.) qui définissent l'architecture et le comportement de l'apprentissage.

5 Cadre Pratique : Évaluation et Surapprentissage

Le mécanisme de rétropropagation doit être encadré par une méthodologie d'évaluation rigoureuse.

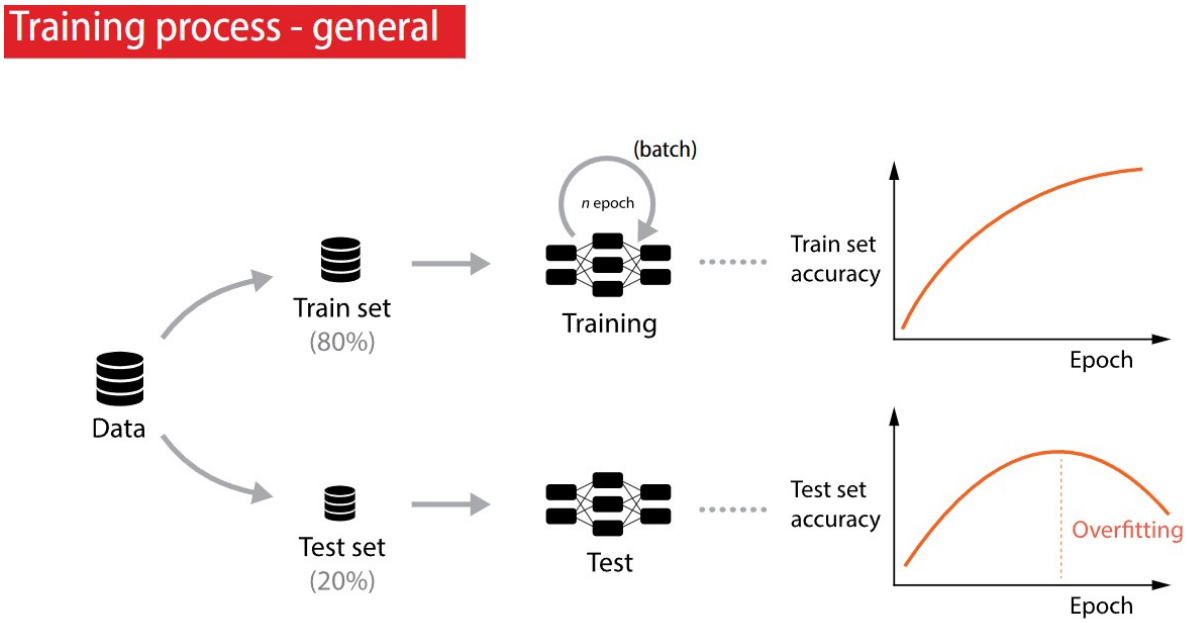

FIGURE 2 – Processus d'entraînement, évaluation et illustration du surapprentissage (overfitting).

5.1 Séparation des Données (Train-Test Split)

- **Ensemble d'entraînement (Train set)** : Utilisé exclusivement pour que le modèle apprenne et ajuste ses paramètres.
- **Ensemble de test (Test set)** : Mis de côté et utilisé uniquement à la fin pour une évaluation impartiale de la capacité de **généralisation**.

Un troisième ensemble de **validation** est souvent utilisé pour ajuster les hyperparamètres et décider quand arrêter l'entraînement (early stopping).

5.2 Interprétation des Courbes d'Apprentissage

- **Performance sur le Train Set** : Devrait augmenter de manière continue.
- **Performance sur le Test Set** : C'est le véritable indicateur.
  1. **Phase d'apprentissage** : La performance sur le test set augmente.
  2. **Surapprentissage (Overfitting)** : La performance sur le test set stagne puis chute, tandis que celle sur le train set continue de s'améliorer. Le modèle "mémorise" au lieu de généraliser.

L'objectif est de stopper l'entraînement au point optimal pour maximiser la performance sur des données inconnues.