Introduction à l'Intelligence Artificielle

https://sleek-think.ovh/enseignement

Dr. Jehan-Antoine Vayssade

Backpropagation

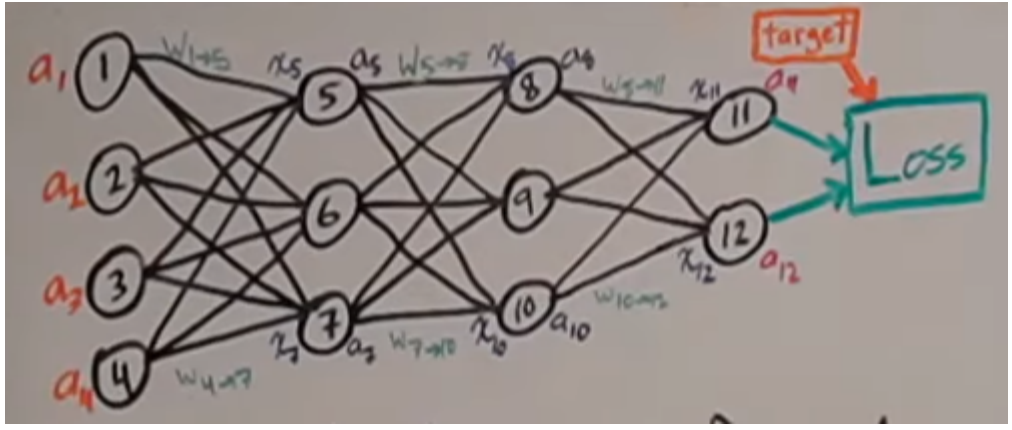
Étapes de l’algorithme

L’algorithme pour cette mise à jour se déroule en trois étapes :

1. Utiliser une passe avant (forward) pour faire une prédiction sur un point de données.
2. Effectuer une passe arrière (backward) pour calculer les dérivées partielles de la perte.
3. Modifier les paramètres du réseau en utilisant les dérivées partielles.

Forward pass

On stock chacune des activations que nous avons calculées car elles contribuent aux calculs des dérivées partielles plus tard.



Fonction de perte / loss

On défini une fonction de perte pour un réseau de neurones où, pour chaque point de données j et chacune des sorties du réseau o, nous avons calculé une erreur quadratique :

$$L = \frac{1}{N} \sum_{j=1}^{N} \sum_{o=1}^{O} (y_{j,o} - \hat{y}_{j,o})^2$$

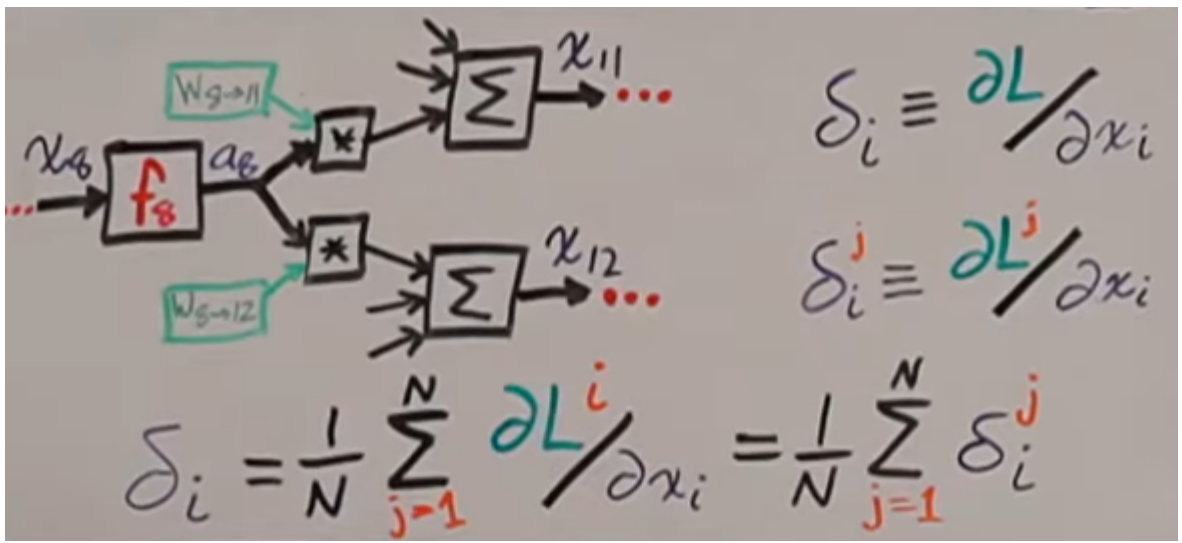
Où

- O pour output : nombre de sorties du réseau
- Dans le cas d'une classification binaire, $O = 1$
- y_o : valeur réelle de la sortie o
- $\hat{y}_o$: valeur prédite de la sortie o
- $y_{j,i}$: vecteur attendu
- $\hat{y}_{j,i}$: vecteur prédit

Backward

La tache de descente de gradient est de déterminer comment les poids du réseau influencent notre erreur. On peu calculer des dérivées partielles, l'étape clé est de calculer une quantité que nous appellerons δ pour chacun des nœuds des couches de sortie et cachées du réseau.

Pour tout neurone, nous définissons δ comme $\delta_i = \frac{\partial L}{\partial x_i}$.
Où x_i est la somme pondérée des entrées à ce neurone.



Calcul de δ pour la couche de sortie

Pour un neurone de la couche de sortie, le δ pour le point de données j est donné par :

$$\delta_o^j = -2(y_o^j - \hat{y}_o^j) \cdot \sigma'(y_o^j - \hat{y}_o^j)$$

$$\delta_o^j = -2(y_o^j - \hat{y}_o^j) \cdot \frac{\partial a}{\partial x_o^j}(y_o^j - \hat{y}_o^j) = -2(y_o^j - \hat{y}_o^j) \cdot \frac{\partial a_o^j}{\partial x_o^j} = -2e_o^j \cdot \sigma'(a_o^j)$$

Pour un neurone de la couche de sortie, le δ pour l'enssemble du dataset est donné par :

$$\delta = \frac{1}{N}\sum_{j=1}^{N} \delta_j = \frac{1}{N}\sum_{j=1}^{N}\sum_{o=1}^{O} \delta_o^j$$

Où

- σ' est la dérivée de la fonction d'activation.
- $a_{j,i}$ est l'activation du neuronne pour une observation j et une sortie i

Calcul de δ pour les couches cachées

Pour un neurone de la couche cachée, le delta est calculé comme :

$$\delta_i^j = \sigma'(a_k^j) \sum_l \delta_l^j \cdot w_{k,l}$$

Exemple pour le neurone 8, le delta est calculé comme :

$$\delta_8^j = \sum_l \delta_n^l \frac{\partial x_n^j}{\partial x_8^j}$$

$$\frac{\partial x_n^j}{\partial x_8^j} = \sigma'(a_8^j) \cdot w8 \to n$$

$$\sum_l \delta_n^j \sigma'(a_8^j) \cdot w8 \to n$$

Dérivées partielles pour les poids et les biais

La dérivée partielle de la perte par rapport à un poids $w_{k,l}$ est :

$$\frac{\partial L^j}{\partial w_l} = \delta_l^j \cdot a_k^j \rightarrow \frac{\partial L}{\partial w_{k,l}} = \frac{1}{N} \sum_{j=0} N \delta_l^j \cdot a_k^j$$

Où a_k^j est l'activation du neurone k pour le point de données j, donc.

La dérivée partielle de la perte par rapport à un biais b_l est :

$$\frac{\partial L}{\partial b_l} = \delta_{j,l}$$

C'est a dire l'erreur, donc

Descente de gradient stochastique

Au lieu d'effectuer une descente de gradient exacte, nous utilisons souvent une descente de gradient stochastique en échantillonnant aléatoirement un sous-ensemble des points de données. Nous calculons la perte moyenne sur ce sous-ensemble et utilisons le gradient de l'erreur moyenne sur ce sous-ensemble comme approximation du gradient de la perte sur l'ensemble du jeu de données.

La passe de mise à jour à travers le réseau changera chacun des paramètres par η fois la dérivée partielle moyenne calculée sur l'échantillon :

$$w_{k,l} = w_{k,l} - \eta \frac{\partial L}{\partial w_{k,l}}$$

$$b_l = b_l - \eta \frac{\partial L}{\partial b_l}$$

Où η est le taux d'apprentissage. Jusqu'a present on a utilisé un sous-enssemble de taille 1.