

Solutions TD N°3: Backpropagation et descente de gradient

Youcef SKLAB

November 4, 2024

Exercice : Backpropagation

Considérons un réseau de neurones à trois couches : une couche d'entrée, une couche cachée et une couche de sortie. Le réseau est défini comme suit :

- La couche d'entrée comporte deux neurones (x_1 et x_2).
- La couche cachée comporte deux neurones (h_1 et h_2) avec une fonction d'activation sigmoïde.
- La couche de sortie comporte un neurone ($\hat{y}$) avec une fonction d'activation sigmoïde.

Les poids sont initialisés comme suit :

- w_1, w_2, w_3, w_4 pour les connexions entre les entrées et la couche cachée.
- w_5, w_6 pour les connexions entre la couche cachée et la couche de sortie.

Le biais de chaque neurone est noté respectivement b_1, b_2 pour la couche cachée, et b_3 pour la couche de sortie.

On vous donne les valeurs suivantes :

$$\begin{aligned}x_1 &= 0.5, & x_2 &= 0.1 \\w_1 &= 0.4, & w_2 &= 0.3, & w_3 &= 0.2, & w_4 &= 0.8 \\w_5 &= 0.6, & w_6 &= 0.9 \\b_1 &= 0.1, & b_2 &= -0.2, & b_3 &= 0.05 \\y &= 1 \quad (\text{valeur cible})\end{aligned}$$

Question : Utilisez l'algorithme de rétropropagation pour mettre à jour les poids w_5 et w_6 . Utilisez une fonction d'activation sigmoïde pour tous les neurones, définie par :

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

Le taux d'apprentissage est $\eta = 0.01$.

Solution 1

Pour résoudre cet exercice, nous suivrons les étapes suivantes :

Étape 1 : Propagation Avant (Forward Propagation)

Calculons d'abord les valeurs de sortie de chaque couche.

1. **Couche cachée :**

$$z_1 = w_1x_1 + w_2x_2 + b_1 = (0.4 \times 0.5) + (0.3 \times 0.1) + 0.1 = 0.33$$

$$h_1 = \sigma(z_1) = \frac{1}{1 + e^{-0.33}} \approx 0.5818$$

$$z_2 = w_3x_1 + w_4x_2 + b_2 = (0.2 \times 0.5) + (0.8 \times 0.1) - 0.2 = -0.02$$

$$h_2 = \sigma(z_2) = \frac{1}{1 + e^{0.02}} \approx 0.4950$$

2. **Couche de sortie :**

$$z_3 = w_5h_1 + w_6h_2 + b_3 = (0.6 \times 0.5818) + (0.9 \times 0.4950) + 0.05 = 0.8446$$

$$\hat{y} = \sigma(z_3) = \frac{1}{1 + e^{-0.8446}} \approx 0.6994$$

Étape 2 : Calcul de l'Erreur

L'erreur est définie par la fonction de coût d'entropie croisée (Cross-Entropy Loss) :

$$E = -[y \cdot \ln(\hat{y}) + (1 - y) \cdot \ln(1 - \hat{y})]$$

En substituant les valeurs :

$$E = -[1 \cdot \ln(0.6994) + (1 - 1) \cdot \ln(1 - 0.6994)] = -\ln(0.6994) \approx 0.3575$$

Étape 3 : Rétropropagation de l'Erreur

Pour mettre à jour les poids w_5 et w_6 , nous devons calculer les gradients de l'erreur par rapport à ces poids.

1. **Gradient par rapport à la sortie ($\hat{y}$) :**

$$\delta_3 = \hat{y} - y = 0.6994 - 1 = -0.3006$$

2. **Gradients pour les poids w_5 et w_6 :**

$$\frac{\partial E}{\partial w_5} = \delta_3 \cdot h_1 = (-0.3006) \times 0.5818 \approx -0.1749$$

$$\frac{\partial E}{\partial w_6} = \delta_3 \cdot h_2 = (-0.3006) \times 0.4950 \approx -0.1488$$

Étape 4 : Mise à Jour des Poids

Nous utilisons la règle de mise à jour des poids :

$$w_i \leftarrow w_i - \eta \cdot \frac{\partial E}{\partial w_i}$$

avec $\eta = 0.01$.

$$w_5 \leftarrow 0.6 - 0.01 \times (-0.1749) = 0.6 + 0.0017 \approx 0.6017$$

$$w_6 \leftarrow 0.9 - 0.01 \times (-0.1488) = 0.9 + 0.0015 \approx 0.9015$$

Conclusion

Après une itération de rétropropagation, les nouveaux poids sont :

$$w_5 \approx 0.6017$$

$$w_6 \approx 0.9015$$

Cette mise à jour des poids réduit l'erreur de sortie en rapprochant la prédiction de la valeur cible. En répétant ce processus pour plusieurs itérations (jusqu'à convergence), le réseau apprendra les relations entre les entrées et les sorties de manière optimale.

Solution 2

Calcul du Gradient

1. La dérivée de $L(\theta) = (\theta - 3)^2$ par rapport à θ est donnée par :

$$\frac{dL}{d\theta} = 2(\theta - 3)$$

2. En évaluant le gradient pour les valeurs de $\theta = 0$, $\theta = 2$, et $\theta = 4$:

- Pour $\theta = 0$:

$$\frac{dL}{d\theta} = 2(0 - 3) = -6$$

- Pour $\theta = 2$:

$$\frac{dL}{d\theta} = 2(2 - 3) = -2$$

- Pour $\theta = 4$:

$$\frac{dL}{d\theta} = 2(4 - 3) = 2$$

Interprétation : Le signe du gradient indique la direction dans laquelle θ doit être mis à jour pour réduire la fonction de perte. Un gradient négatif indique que θ doit augmenter, tandis qu'un gradient positif indique que θ doit diminuer.

Descente de Gradient - Mise à Jour du Paramètre

1. Avec un taux d'apprentissage $\eta = 0.1$ et en partant de $\theta = 0$, la mise à jour est :

$$\theta_{\text{new}} = \theta - \eta \cdot \frac{dL}{d\theta} = 0 - 0.1 \times (-6) = 0.6$$

2. En répétant la mise à jour pour 3 itérations supplémentaires :

- Itération 2 : $\theta = 0.6$

$$\begin{aligned}\frac{dL}{d\theta} &= 2(0.6 - 3) = -4.8 \\ \theta_{\text{new}} &= 0.6 + 0.1 \times 4.8 = 1.08\end{aligned}$$

- Itération 3 : $\theta = 1.08$

$$\begin{aligned}\frac{dL}{d\theta} &= 2(1.08 - 3) = -3.84 \\ \theta_{\text{new}} &= 1.08 + 0.1 \times 3.84 = 1.464\end{aligned}$$

- Itération 4 : $\theta = 1.464$

$$\begin{aligned}\frac{dL}{d\theta} &= 2(1.464 - 3) = -3.072 \\ \theta_{\text{new}} &= 1.464 + 0.1 \times 3.072 = 1.7712\end{aligned}$$

Interprétation : À chaque itération, θ se rapproche progressivement de la valeur optimale $\theta = 3$.

Effet du Taux d'Apprentissage

1. Avec un taux d'apprentissage $\eta = 0.5$, reprenons les mises à jour en partant de $\theta = 0$:

- Itération 1 :

$$\theta_{\text{new}} = 0 - 0.5 \times (-6) = 3$$

En une seule itération, θ atteint le minimum de la fonction de perte. Interprétation : Un taux d'apprentissage plus élevé permet une convergence plus rapide, mais peut également causer des oscillations si le pas est trop grand.

2. Avec $\eta = 1.5$, la mise à jour de $\theta = 0$ devient :

$$\theta_{\text{new}} = 0 - 1.5 \times (-6) = 9$$

Interprétation : θ dépasse largement le minimum (qui est 3). Un taux d'apprentissage trop élevé peut causer une divergence au lieu de la convergence vers le minimum.

Application de la Descente de Gradient avec Deux Paramètres

Pour $L(\theta_1, \theta_2) = (\theta_1 - 2)^2 + (\theta_2 + 3)^2$:

1. Les gradients sont :

$$\frac{\partial L}{\partial \theta_1} = 2(\theta_1 - 2), \quad \frac{\partial L}{\partial \theta_2} = 2(\theta_2 + 3)$$

2. En partant de $\theta_1 = 0$ et $\theta_2 = 0$, et avec $\eta = 0.1$:

$$\frac{\partial L}{\partial \theta_1} = 2(0 - 2) = -4, \quad \frac{\partial L}{\partial \theta_2} = 2(0 + 3) = 6$$

$$\theta_1^{\text{new}} = 0 + 0.1 \times 4 = 0.4, \quad \theta_2^{\text{new}} = 0 - 0.1 \times 6 = -0.6$$

3. Répétitions pour 2 itérations supplémentaires :

- Itération 2 : $\theta_1 = 0.4, \theta_2 = -0.6$

$$\frac{\partial L}{\partial \theta_1} = 2(0.4 - 2) = -3.2, \quad \frac{\partial L}{\partial \theta_2} = 2(-0.6 + 3) = 4.8$$

$$\theta_1^{\text{new}} = 0.4 + 0.1 \times 3.2 = 0.72, \quad \theta_2^{\text{new}} = -0.6 - 0.1 \times 4.8 = -1.08$$

- Itération 3 : $\theta_1 = 0.72, \theta_2 = -1.08$

$$\frac{\partial L}{\partial \theta_1} = 2(0.72 - 2) = -2.56, \quad \frac{\partial L}{\partial \theta_2} = 2(-1.08 + 3) = 3.84$$

$$\theta_1^{\text{new}} = 0.72 + 0.1 \times 2.56 = 0.976, \quad \theta_2^{\text{new}} = -1.08 - 0.1 \times 3.84 = -1.464$$

Trajectoire des paramètres : Les valeurs de (θ_1, θ_2) se rapprochent progressivement des valeurs optimales $(2, -3)$.

Impact de la Convergence et du Choix des Hyperparamètres

1. Effet du Taux d'Apprentissage : Un taux d'apprentissage trop élevé risque de faire diverger le modèle, car les mises à jour sautent autour du minimum sans y converger. Un taux trop faible ralentit la convergence.
2. Taux d'apprentissage dynamique : En réduisant progressivement le taux d'apprentissage au cours des itérations, le modèle effectue de grands pas au début pour converger rapidement, puis des ajustements plus petits lorsqu'il approche du minimum. Cette approche aide à éviter les oscillations autour du minimum et augmente la probabilité de convergence vers le minimum global, surtout dans des fonctions de perte complexes.

Par exemple, on peut définir un taux d'apprentissage dynamique sous la forme suivante :

$$\eta = \frac{\eta_0}{1 + k \cdot t}$$

où η_0 est le taux d'apprentissage initial, k est un paramètre de décroissance, et t représente le numéro d'itération. Cette méthode permet de stabiliser la descente de gradient au fur et à mesure que le modèle se rapproche de l'optimum.