Descente de gradient - Variantes

October 29, 2024

1 Descente de gradient

La descente de gradient (gradient descent, GD) est une méthode d'optimisation couramment utilisée pour minimiser une fonction objectif, dans le contexte de l'apprentissage machine. Elle consiste à mettre à jour les paramètres du modèle dans la direction opposée au gradient de la fonction de coût, afin de trouver le minimum local ou global.

1.0.1 Formulation de la Descente de Gradient

Soit $f(\theta)$ une fonction de coût (ou fonction objectif) que l'on souhaite minimiser, où θ représente les paramètres du modèle. La mise à jour des paramètres est donnée par :

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

où : - θ_t est le vecteur des paramètres à l'itération t. - η est le taux d'apprentissage, un hyperparamètre qui contrôle la taille du pas de mise à jour. - $\nabla f(\theta_t)$ est le **gradient** de la fonction de coût par rapport aux paramètres θ, évalué à l'itération t.

L'idée de base est que le gradient de la fonction de coût indique la direction de la plus forte augmentation de la fonction. En soustrayant ce gradient, nous nous déplaçons dans la direction opposée, ce qui diminue la valeur de la fonction de coût.

1.0.2 Variantes de la Descente de Gradient

Il existe plusieurs variantes de la descente de gradient, chacune ayant ses avantages et ses inconvénients. Les principales variantes sont :

1.0.3 Descente de Gradient Standard (Batch Gradient Descent - BGD)

- Formulation : Dans la descente de gradient standard, le gradient est calculé en utilisant l'ensemble complet des données d'entraînement. La mise à jour des paramètres se fait après avoir parcouru l'ensemble des données.

- **Avantages**:

- Converge généralement vers le minimum global pour les fonctions convexes.
- Mise à jour stable des paramètres.

- **Inconvénients**:

- Peut être très coûteux en termes de calcul pour de grands ensembles de données, car il faut parcourir toutes les données avant chaque mise à jour.
- Convergence lente dans les grands ensembles de données.

1.1 Descente de Gradient Stochastique (Stochastic Gradient Descent - SGD)

- **Formulation** : Dans SGD, les paramètres sont mis à jour après chaque échantillon de données. La mise à jour des paramètres est donnée par :

$$\theta_{t+1} = \theta_t - \eta \nabla f_i(\theta_t)$$

où $f_i(\theta_t)$ représente la fonction de coût pour l'échantillon i.

- **Avantages** :

- Mise à jour plus fréquente des paramètres, ce qui peut conduire à une convergence plus rapide.
- Plus efficace pour les grands ensembles de données.
- Peut échapper aux minima locaux en raison de la nature stochastique des mises à jour.

- **Inconvénients** :

- Les mises à jour sont plus bruyantes, ce qui peut rendre la convergence moins stable.
- Le taux d'apprentissage doit être soigneusement ajusté pour éviter les oscillations.

1.1.1 Descente de Gradient par Mini-lots (Mini-Batch Gradient Descent

- **Formulation** : La descente de gradient par mini-lots combine les approches de BGD et SGD. Les paramètres sont mis à jour après chaque mini-lot de m échantillons de données. La mise à jour des paramètres est donnée par :

$$\theta_{t+1} = \theta_t - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(\theta_t)$$

où m est la taille du mini-lot.

- **Avantages** :

- Offre un compromis entre la stabilité de BGD et la rapidité de SGD.
- Réduit le bruit des mises à jour par rapport à SGD tout en restant plus rapide que BGD.

- **Inconvénients** : - La performance dépend du choix de la taille du mini-lot.

2 Variantes Avancées de la Descente de Gradient

Plusieurs variantes avancées de la descente de gradient ont été développées pour améliorer la convergence et la performance. Voici quelques-unes des plus courantes :

2.1 Descente de Gradient avec Momentum

Formulation : Ajoute un terme de vitesse aux mises à jour, basé sur les gradients passés. La mise à jour est donnée par :

$$v_{t+1} = \gamma v_t + \eta \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t - v_{t+1}$$

où γ est le facteur de momentum (typiquement entre 0,5 et 0,9).

- **Avantages** :

- Aide à accélérer la convergence et à réduire les oscillations.
- Plus efficace pour les problèmes avec des vallées étroites.

2.2 Nesterov Accelerated Gradient (NAG)

- **Formulation** : Améliore le momentum en anticipant la mise à jour future. La mise à jour est donnée par :

$$v_{t+1} = \gamma v_t + \eta \nabla f(\theta_t - \gamma v_t)$$
$$\theta_{t+1} = \theta_t - v_{t+1}$$

- **Avantages**:
- Permet d'ajuster la vitesse de convergence en fonction de la courbure locale.
- Prévient les erreurs de mise à jour lorsqu'on approche d'un minimum.

2.3 Descente de Gradient Adaptative (Adagrad)

- **Formulation** : Ajuste le taux d'apprentissage pour chaque paramètre en fonction de la fréquence de mise à jour. La mise à jour est donnée par :

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla f(\theta_t)$$

où G_t est la somme des carrés des gradients passés et ϵ est un terme de régularisation pour éviter la division par zéro.

- **Avantages**:
- Adaptation automatique du taux d'apprentissage.
- Convient aux problèmes où les gradients sont rares ou d'amplitudes très différentes.

2.4 RMSprop

- **Formulation** : Variante d'Adagrad qui utilise une moyenne mobile exponentielle des gradients passés pour éviter que le taux d'apprentissage ne diminue trop rapidement. La mise à jour est donnée par :

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)(\nabla f(\theta_t))^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla f(\theta_t)$$

- **Avantages** : - Améliore la performance d'Adagrad en évitant les diminutions excessives du taux d'apprentissage.

2.5 Adam (Adaptive Moment Estimation)

- **Formulation** : Combine les idées de RMSprop et du momentum. La mise à jour est donnée par :

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla f(\theta_t))^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

où m_t et v_t représentent les estimations du premier et du second moment du gradient, et β_1 et β_2 sont les coefficients de lissage.

- **Avantages** :

- Combine les forces de RMSprop et du momentum.
- Convient aux problèmes avec de grands jeux de données et de nombreux paramètres.

2.6 Conclusion

Les différentes variantes de la descente de gradient permettent de s'adapter à différents types de problèmes et de données. Le choix de la variante à utiliser dépend souvent de la nature de la fonction de coût, de la taille des données, et des ressources de calcul disponibles.