UNIVERSITÉ SORBONNE PARIS NORD | Institut GALILÉE Université Sorbonne Paris Nord

Master EID²

Exploration Informatique des Données et Décisionnel

UE Science des Données Numériques

Fiche TP N° : 3

Les TP sont notés. Les étudiants doivent sauvegarder le code de chaque exercice et doivent présenter le résultat au chargé du TP.

Ce TP présente ce qu'est une Descente de Gradient (DG). La DG s'applique lorsque l'on cherche le minimum d'une fonction dont on connaît l'expression analytique, qui est dérivable, mais dont le calcul direct du minimum est difficile. C'est un algorithme fondamental à connaître car utilisé partout sous des formes dérivées. Nous n'étudions ici que la version de base.

Descente de gradient

Ce paragraphe présente la DG sur la minimisation de la fonction $E(x)$ quelconque. Le problème est de trouver la valeur de x qui minimise $E(x)$. Pour trouver analytiquement le minimum de la fonction E, il faut trouver les racines de l'équation $E'(x) = 0$, donc trouver ici les racines d'un polynôme de degré 3, ce qui est des fois "difficile". Donc on va utiliser la DG. La DG consiste à construire une suite de valeurs x_i (avec x_0 fixé au hasard) de manière itérative :

$$x_{i+1} = x_i - \eta E'(x_i).$$

On peut donner un critère de fin à la DG par exemple si $x_{i+1} - x_i < \epsilon$ ou si $i > nombre_{max}$. Pour ce problème, nous utilisons $\epsilon = 0.01$ et $nombre_{max} = 1000$.

1. Calculez l'expression analytique de la fonction $E(x) = (x-1)(x-2)(x-3)(x-5)$ et sa dérivée.
2. Implémentez l'algorithme DG sous Python pour la fonction $E(x)$.
3. Pour comprendre ce que fait effectivement la DG, testez l'algorithme implémenté en utilisant des exemples d'exécutions avec des valeurs initiales de x_0 et η suivantes :
   (a) $x_0 = 5$ et $\eta = 0.001$
   (b) $x_0 = 5$ et $\eta = 0.01$
   (c) $x_0 = 5$ et $\eta = 0.1$
   (d) $x_0 = 5$ et $\eta = 0.17$
   (e) $x_0 = 5$ et $\eta = 1$
   (f) $x_0 = 0$ et $\eta = 0.001$
4. Affichez le minimum trouvé, ainsi que $E(x_{min})$ et le nombre d'itérations. Que constatez-vous ?
5. Visualisez l'évolution des minimums de la fonction $E(x)$ trouvés au cours des itérations.
6. Testez votre algorithme avec d'autres valeurs de ϵ et $nombre_{max}$.

Descente de gradient pour la régression linéaire

Maintenant considérons le modèle de la régression linéaire

$$\hat{y}_i = ax_i + b,$$

où $(a; b)$ sont des coefficients de régressions inconnus à estimer, x_i les données de régression, y le vecteur des données des sorties y_i.

Dans ce cas, la fonction de coût à minimiser est la suivante :

$$E(a, b) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2.$$

1. Calculez les dérivées partielles de la fonction $E(a, b)$ selon a et b.
2. Implémentez l'algorithme DG sous Python pour la fonction $E(a, b)$. NB : Pour le cas avec deux dimensions, les règles de mis à jour de la DG deviennent :

$$a_{i+1} = a_i - \eta \frac{\partial E(a, b)}{\partial a}(a_i),$$

$$b_{i+1} = b_i - \eta \frac{\partial E(a, b)}{\partial b}(b_i).$$

3. Importez la fonction *datasets.make_ regression* et utilisez la pour générez un problème de régression aléatoire de 100 example avec une seule variable. Appliquez l'algorithme implémenté au jeu de données généré avec les paramètres suivants :
   (a) $\eta = 0.001$, $nombre_{max} = 100$
   (b) $\eta = 0.001$, $nombre_{max} = 500$
   (c) $\eta = 0.001$, $nombre_{max} = 1000$
   (d) $\eta = 0.01$, $nombre_{max} = 1000$
   (e) $\eta = 1$, $nombre_{max} = 1000$
4. Affichez les coefficients trouvés, ainsi que la valeur de $E(a_{min}, b_{min})$ et le nombre d'itérations. Que constatez-vous ?
5. Importez la fonction stats.linregress de scipy. Utilisez cette fonction pour résoudre le même problème. Comparez les résultats obtenus. Que constatez-vous ?
6. Visualisez le jeu de données généré avec les fonctions approximatives obtenues en utilisant les deux méthodes.