Travail pratique - Optimisation

Mathématiques en technologie de l'information

1 Objectif

— Réaliser un programme permettant de réaliser une régression linéaire à l'aide de la méthode de la descente de gradient.
— Tester ce programme sur des données synthétiques (générées aléatoirement) afin de valider votre implémentation.

2 Travail à réaliser

2.1 La régression linéaire à une seule variable

2.1.1 Solution analytique

Afin de *valider* votre implémentation, il faut d'abord étudier un cas simplifié où trouver la solution analytique est aisé.

On va chercher "la meilleure droite" passant par un ensemble de points $\{(x_j, y_j)\}_{j=1}^{N}$ (Ex pour $N = 3$: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}_{j=1}^{3}$). Comme on l'a vu en cours, on cherche à minimiser la fonction de coût (erreur quadratique)

$$E(a, b) = \sum_{j=1}^{N} (a \cdot x_j + b - y_j)^2. \tag{1}$$

En résolvant, le système de deux équations à deux inconnues

$$\vec{\nabla} E(a, b) = \vec{0}, \tag{2}$$

on peut trouver la valeur de a et b pour n'importe quel ensemble de points $\{(x_j, y_j)\}_{j=1}^{N}$.

Votre premier exercice sera de trouver l'expression de a et b en fonction de $\{(x_j, y_j)\}_{j=1}^{N}$ analytiquement (avec un papier et un crayon). En d'autres termes, on cherche une formule pour a et une pour b ne dépendant que des valeurs des points (x_j, y_j).

2.1.2 Solution numérique

En prenant comme référence la solution ci-dessus, il faut à présent implémenter la méthode de la descente de gradient pour minimiser $E(a, b)$.

En partant d'une pente a_0 et d'une ordonnée à l'origine b_0 (choisies aléatoirement), il faut itérativement construire de meilleures approximations

$$\begin{pmatrix} a_{i+1} \\ b_{i+1} \end{pmatrix} = \begin{pmatrix} a_i \\ b_i \end{pmatrix} - \lambda \cdot \vec{\nabla} E(a_i, b_i), \tag{3}$$

avec $i \geq 0$ et $\lambda \in [0, 1)$. On arrêtera les itérations lorsque

$$\left|\left| \begin{pmatrix} a_{i+1} \\ b_{i+1} \end{pmatrix} - \begin{pmatrix} a_i \\ b_i \end{pmatrix} \right|\right| < \varepsilon, \tag{4}$$

où $\varepsilon > 0$ est la précision souhaitée.

2.1.3 Test

Afin de tester votre programme, vous devez générer un nuage de points $\{(x_j, y_j)\}_{j=1}^{N}$ aléatoirement. Pour contrôler au mieux ce qu'il se passe, il est recommandé de générer ces points aléatoirement le long d'une droite de pente c et une ordonnée à l'origine d que vous choisirez, et de bruiter un peu le résultat. Pour générer aléatoirement un point (x_j, y_j), vous choisissez x_j entre deux bornes de votre choix (p.ex. 0 et 1) puis, à partir de là vous construisez y_j comme

$$y_j = c \cdot x_j + d + r_j, \tag{5}$$

où $|r_j|$ est un "petit" nombre aléatoire devant $(c \cdot x_j + d)$.

Il faut vous assurer que la solution analytique et la solution numérique soient très proches (à ε près) et qu'elles soient également assez proches du c et du d que vous avez choisis.

Tester votre code sur différentes valeurs de c et d. Est-ce que vos résultats sont toujours cohérents? Quelle est la valeur de l'erreur moyenne? Qu'est-ce que l'erreur signifie? Faites également varier la valeur maximale de $|r_j|$. Que se passe-t-il quand $|r_j|$ devient trop grand? N'hésitez pas à représenter graphiquement vos résultats.

Important : Pour des raisons de pertes de précision obtenus après des calculs itératifs sur des nombres à virgule flottante. Vous devrez choisir des valeurs avec les contraintes suivantes :

$$\begin{aligned} x_j &\in [0, 1] \\ c, d &\in]0, 1] \end{aligned} \tag{6}$$

(Note : En pratique le domaine de nos données n'est pas restraint. On effectue donc une normalisation de nos données avant et après le calcul des paramètres de notre modèle.)

2.2 Validation du modèle de régression

Lorsqu'on réalise une régression, on *modélise* notre nuage de points. Ici, on dit que le phénomène qui a généré les points suit une droite plutôt qu'une parabole ou une exponentielle ou n'importe quelle autre fonction. Afin de s'assurer que notre modèle correspond relativement bien à notre jeu de donnée, on peut faire ce qu'on appelle une *validation croisée* (ou *cross validation* en bon français).

Cette technique est très utilisée en apprentissage automatique. Il en existe un grand nombre de variantes, ici nous n'en verrons qu'une.

Il s'agit ici de vérifier si le a et le b que nous avons déterminés sont des valeurs qui continueraient à être correctes si on ajoutait de nouveaux points à notre ensemble $\{(x_j, y_j)\}_{j=1}^{N}$ (issues du même phénomène). Il est souvent peu pratique de générer de nouveaux points, on se contente donc de diviser notre jeu de données en plusieurs partie. Une partie des points sera utilisée pour *entraîner* notre modèle (déterminer un a et un b) l'autre partie sera utilisée pour *tester* le modèle, on calculera l'erreur effective $E(a, b)$ par rapport à cette seconde partie des points.

Ici, pour simplifier on va séparer notre ensemble de N points en trois groupes de taille égale et en répartissant les points aléatoirement dans les groupes. Nommons les groupes G_1, G_2, et G_3. Pour effectuer la validation croisée, il faut réaliser les étapes suivantes:

— entraîner le modèle avec les groupes $G_1 \cup G_2$ et tester sur G_3,
— entraîner le modèle avec les groupes $G_1 \cup G_3$ et tester sur G_2,
— entraîner le modèle avec les groupes $G_2 \cup G_3$ et tester sur G_1.

Pour les nuages de point générés à la section précédente, quelle est la valeur de l'erreur pour chacun des groupes de tests? (Donner les valeurs sous forme de tableau peut être une bonne idée.) Comment interprétez vous ces résultats? N'hésitez pas à représenter graphiquement vos résultats.

3 Rendu

Il faut rendre un rapport de quelques pages (quelques: **plus petit** que 6). Ce rapport doit être relativement bref et expliquer votre travail. Il doit être composé de quatre parties principales:

1. Une introduction générale qui décrit le cadre général du travail (ce que vous essayez de réaliser, par quels moyens, etc) et donner la structure de votre rapport (que contient chaque autre partie).
2. Une partie "théorique" décrire les concepts et méthodes que vous utilisez si cela est nécessaire afin de permettre une bonne compréhension du reste de votre travail par le lecteur. Ceci est nécessaire pour comprendre comment vous arrivez aux résultats que vous présentez dans la partie suivante.
3. Une partie résultats, où vous donnez les résultats que vous avez obtenus. Par exemple, répondre aux différentes questions posées dans cet énoncé, mais n'hésitez pas à développer.
4. Une conclusion où vous résumez les résultats principaux de votre travail et éventuellement ouvrez sur comment vous pourriez améliorer votre travail ou l'étendre.

Le code doit être réalisé en C (afin de vous entraîner). La visualisation peut être faite avec l'outil de votre choix. Python avec la librairie *matplotlib* peut-être un bon choix.

Le rapport doit être écrit en **Markdown** ou **LaTeX** (ou une combinaison des deux avec Pandoc).

Vous **devez** faire ce travail par groupe de 2 et aucune exception ne sera faite. Vous devez rendre le rapport sur `cyberlearn`. Le code doit être dans un repo git public dont vous mettrez l'url sur `cyberlearn`. N'oubliez pas de bien spécifier le nom des deux membres du groupe dans le rapport et dans le code. Je devrais pouvoir compiler et exécuter votre projet (pensez évidemment à créer un `Makefile`). Le travail doit être rendu **au plus tard le 13.12.2021 à 23h59**.

La note est une combinaison de la note du code et du rapport.

4 Conseils et remarques

Ce travail est loin d'être simple à réaliser. Il demande de combiner beaucoup de concept vu ou pas en détail en cours. Utilisez le temps à disposition pendant les séance pour poser des questions et n'attendez pas le dernier moment.

La rédaction du rapport est également une tâche complexe et il s'agit de ne pas bâcler sa réalisation. C'est un exercice qui vous sera utile lorsque vous devrez écrire votre mémoire pour votre travail de bachelor.