examen_python_2023

November 8, 2023

0.1 La librairie pytorch

PyTorch est un framework d'apprentissage profond (deep learning) open source principalement développé par le groupe de recherche en IA de Facebook. Pour installer *PyTorch* dans votre environnement conda, rendez-vous sur https://pytorch.org et sélectionnez votre système d'exploitation, conda, Python, CPU (ceux qui possèdent une GPU peuvent installer la version CUDA mais ça peut déclencher plein de bugs à gérer pour installer la bonne version de cuda). Copiez le texte de la boîte "Run this command :". Ouvrez maintenant un terminal. Collez le texte et exécutez. Une fois l'installation terminée, vous devez redémarrer Jupyter.

0.2 Jeu de données MNIST

Pour ce travail, nous allons faire appel au données MNIST. C'est un jeu de données contient des images numériques de chiffres manuscrits de 0 à 9. Chaque image a au format $28x28$ pixels qui prennent chacun 256 valeurs dans une gamme allant du blanc (= 0) au noir (= 1). Les étiquettes appartenant aux images sont également incluses. Heureusement, *PyTorch* est équipé d'un chargeur de données MNIST. La première fois que vous exécutez le code de la cellule ci-dessous, elle télécharge automatiquement le jeu de données MNIST. Cela peut prendre quelques minutes. Les principaux types de données dans *PyTorch* sont les tenseurs. Dans un premier temps, nous allons convertir ces tenseurs en tableaux *numpy*. Ensuite, nous allons utiliser le module torch pour travailler directement avec les tenseurs *PyTorch*.

[5]:
```
import torch
from torchvision import datasets, transforms

train_dataset = datasets.MNIST('../data', train=True, download=True,
                    transform=transforms.Compose([
                        transforms.ToTensor(),
                        transforms.Normalize((0.1307,), (0.3081,))
                    ]))

train_labels = train_dataset.train_labels.numpy()
train_data = train_dataset.train_data.numpy()
# Nous allons implémenter l'algorithme EM (on a besoin de vecteur)
train_data = train_data.reshape(train_data.shape[0], -1)
```

0.3 Partie 1 : algorithme EM

Nous allons implémenter l'algorithme EM (Expectation Maximisation) pour la reconnaissance des chiffres manuscrits dans le jeu de données MNIST. Les images sont modélisées comme un modèle de mélange de Bernoulli (Voir le livre de Bishop partie §9.3.3):

$$p(x|\mu,\pi) = \sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{(1-x_i)}$$

où x_i est la valeur du i ème pixel d'une image, le paramètre μ_{ki} représente la probabilité que le i ème pixel appartenant à la classe k soit noir, et $\{\pi_1, \ldots, \pi_K\}$ sont les proportions du mélange. On souhaite utiliser ce jeu de données pour construire une méthode de classification de nouvelles images de nombres.

0.3.1 Binarisation des données

La transformation des tenseurs *PyTorch* en vecteurs *numpy* a permis d'aplatir les images en vecteurs. Comme nous allons utiliser un modèle de mélange de Bernoulli, nous avons besoin de binariser les pixels d'un vecteur.

Question 1.1

- Écrire une fonction `binarize` pour convertir chaque vecteur de pixels représentant une image MNIST en vecteur de pixels binaires $x_i \in \{0, 1\}$, en utilisant un seuil (à choisir).
- Écrire une fonction une fonction qui prend en entrées un chiffre allant de 0 à 9 et un nombre de figures correspondantes à ce chiffre à échantillonner et à afficher à côté de la version binariser ce chaque figure. Échantillonner quelques images des chiffres 2, 5 et 9 pour tester la fonction. Mettre comme titre de l'image son étiquette.

0.3.2 Implémentation

Question 1.2

Écrire une fonction `EM(X, K, max_iter)` qui implémente l'algorithme EM pour estimer les paramètres d'une modèle de mélange de Bernoulli.

- Les seuls paramètres d'entrée de la fonction doivent être :
  - `X` :: Une Matrice (array numpy) de dimension $n \times d$ d'images binarisées à utiliser pour l'estimation des paramètres.
  - `K` :: Le nombre de classes du mélange de Bernoulli
  - `max_iter` :: Nombre maximal d'itérations, *i.e.* chaque itération est composée d'une étape E-step et une étape M-step
  - Initialiser les paramètres mu à l'aide d'une loi uniforme entre 0.15 et 0.85 et les π_k à $\frac{1}{K}$.
- L'implémentation des éléments retournés par la fonction sont laissés à votre choix.
- Utiliser un critère d'arrêt raisonnable afin d'éviter de parcourir inutilement toutes les itérations. Penser vectoriser les calculs en utilisant `numpy` quand c'est possible.
- Vous devez implémenter les fonctions `E_step(X, mu, pi)` et `M_step(X, gamma)` séparément pour les utiliser dans votre fonction `EM(X, K, max_iter)`.

0.3.3 Test avec trois chiffres

Question 1.3

- En analogie avec le livre de Bishop partie §9.3.3, échantillonnez un ensemble de données d'apprentissage composé uniquement d'images binarisées des chiffres 2, 3, et 4. Exécutez votre algorithme EM et afficher les paramètres μ_1, μ_2, μ_3 sous formes d'images.
- Pouvez-vous identifier quel classe correspond à quel chiffre ? Quels sont les tailles du mélange identifiées pour les chiffres 2, 3 et 4, ces valeurs correspondent-elles aux vraies proportions des images des 3 chiffres dans le jeu de données ?

0.3.4 Choix de modèle

La question 1.3 faisait appel à 3 classes d'images seulement (chiffres 2, 3 et 4) pour éviter les calcul trop lent. La nombre de classes $K = 3$ est connaissance a priori car nous avons sélectionné les données nous mêmes. Souvent le nombre de classes dans un jeu de données est inconnu et son estimation est traduite par une question de choix de modèle. Ici nous allons faire appel au critère BIC *Bayesian information criterion* pour choisir le nombre de classes. À nombre de classes fixé K le critère BIC est donné par

$$BIC(K) = 2\widehat{\ell}_K - p_K \ln(n),$$

où

- $\widehat{\ell}_K$ est la valeur maximale de la log-vraisemblence associé au modèle à K classes.
- p_K le nombre de paramètres estimés dans le modèle (**attention :** $\sum_{k=1}^{K} \pi_k = 1$).

Question 1.4

- Ajouter aux éléments retournés par la fonction `EM(X, K, max_iter)`, la valeur du BIC associé au modèle estimé.
- Tester les valeur $K = 2, \ldots, 5$. Quelle est la valeur de K qui maximise le critère BIC ?

0.3.5 Images mal-classées

Jusqu'ici, nous n'avons pas utilisé les vraies étiquettes des images dans une étape d'estimation.

Question 1.5

Utilisez les étiquettes des images pour identifier les cas "mal-classées" et afficher quelques exemples d'images mal-classées (affichier dans le titre la vraie étiquette et la fausse étiquette attribuée). Peut-on donner une explication du mauvais classement sur certains cas ?

0.3.6 Initialiser les paramètres π_k avec les vraies valeurs

Question 1.6

Initialisez les tailles des trois classes (π_1, π_2, π_3) avec les vraies valeurs (proportions d'images correspondantes aux chiffres 2, 3 et 4 respectivement dans le jeu de données). Que remarque-t-on ?