Examen Apprentissage Statistique L3 MIASHS 17/05/2021

R. Bailly

Présentation

Vous trouverez sur l'EPI de l'examen un fichier `shapes.py`. Il contient les données d'images de formes prises par une caméra: carré, triangle, disque, ou étoile. Les images sont sous forme bitmap (matrice de 10 pixels su 10 pixels) aplaties en un vecteur de taille 100.

Les données d'entrée sont sous la forme d'un arrays Numpy. Une ligne correspond à une image, il y a autant de lignes que d'images. Les données de sortie sont sous la forme d'un vecteur Numpy de la taille du nombre d'image. Une sortie est un entier 0, 1, 2, ou 3 représentant l'une des formes possibles – carré, triangle, disque, ou étoile.

Les données d'apprentissage se trouvent dans trois arrays (taille de l'échantillon: 500 images):

- `x_learn` représente les vecteurs d'entrée (500 lignes, une ligne par vecteur)
- `y_learn` représente les sorties attendues, sous forme d'entier compris entre 0 et 3 (4 valeurs possibles) (taille 500)
- `z_learn` représente les matrices correspondant aux entrées, de taille 10x10 (500 matrice 10x10)

Vous pouvez visualiser les images à l'aide de la fonction `matplotlib.pyplot.imshow()` en utilisant les matrices contenues dans `z_learn`.

Les données de validation se trouvent dans deux arrays (taille de l'échantillon: 100 images):

- `x_valid` représente les vecteurs d'entrée (100 lignes, une ligne par vecteur)
- `y_valid` représente les sorties attendues, sous forme d'entier compris entre 0 et 3 (4 valeurs possibles) (taille 100)

Les données de validation vous servent à valider le choix du modèle et/ou les paramètres.

Les données de test se trouvent dans un array (taille de l'échantillon: 100 images):

- `x_test` représente les vecteurs d'entrée (100 lignes, une ligne par vecteur)

L'examen porte sur la construction d'un classifieur permettant de déterminer automatiquement la forme (carré, disque...) des vecteurs d'images contenus dans `x_test`. La forme est encodée sous la forme d'un entier 0, 1, 2 ou 3.

Votre tâche est de construire un classifieur en utilisant des techniques d'apprentissage automatique (machine learning), à partir des données étiquetées `x_learn` et `y_learn`, et d'appliquer ce classifieur aux données de test, pour attribuer la bonne forme aux données non étiquetées de `x_test`.

Instructions

Votre objectif est de fournir un vecteur qui contienne la forme (carré, disque...) prédite par votre classifieur sur l'échantillon de test. La forme est encodée par entier.

1) Le premier fichier que vous rendrez sera un fichier `.txt` et les coordonnées du vecteur seront séparées par un espace ou une virgule. Ce fichier contiendra 100 valeurs (une prédiction par image de `x_test`).

Exemple: le fichier pourra commencer comme

```
0 1 2 3 1 2 0 2 ...
```

ou

```
0, 1, 2, 3, 1, 2, 0, 2 ...
```

ce qui signifie que la classe (forme) prédite est:

- 0 pour l'image `x_test[0]`
- 1 pour l'image `x_test[1]`
- 2 pour l'image `x_test[2]`
- 3 pour l'image `x_test[3]`
- 1 pour l'image `x_test[4]`
- 2 pour l'image `x_test[5]`
- 0 pour l'image `x_test[6]`
- 2 pour l'image `x_test[7]`

2) Le second fichier à rendre est celui du programme vous ayant servi à obtenir le résultat.

Une partie importante de votre note sera dépendante du taux de bonnes réponses sur l'échantillon de test!! (barême indicatif: une note de 10/20 correspond à un taux d'erreur inférieur à 10% sur l'échantillon de test)

Indications

Servez vous de l'échantillon de validation pour contrôler le sur-apprentissage, particulièrement si vous décidez d'utiliser un modèle non linéaire.

Vous pouvez utiliser le modèle et la méthode que vous souhaitez. Vous trouverez sur l'EPI deux fichiers pour l'implémentation d'un descente de gradient (simple) pour un modèle linéaire.

Ces fichiers sont `classif.py` et `graddsec.py` se trouvent sur l'EPI de l'examen. Le fichier `graddsec.py` contient la descente de gradient, et le fichier `classif.py` implémente une descente de gradient pour un modèle linéaire.

Le fichier `classif.py` mimplémente une descente de gradient pour le jeu de données `digits`, il faut l'adapter pour l'utiliser avec les données de `shapes.py`.

- Les données sont contenues dans la variable `S`
- Les paramètres du modèle sont dans la variable `T_vec`
- La fonction `nb_error(T_vec, S)` renvoie le nombre d'erreurs commises sur l'échantillon `S` avec le modèle de paramètres `T_vec`
- La fonction `prediction(T_vec, S)` renvoie le vecteur de prédiction pour l'échantillon `S` avec le modèle de paramètres `T_vec`
- La fonction `loss(T_vec, S)` renvoie la perte pour l'échantillon `S` avec le modèle de paramètres `T_vec`, calculée comme la somme des log-probabilités des observations (log-likelihood) item La fonction `grad_desc_n(loss, S, (dim_input+1)*nb_class, 100, step = 0.0001)` prend en argument:
  - La fonction de perte
  - l'échantillon d'apprentissage
  - la dimension du vecteur de paramètres
  - le nombre de pas de la descente de gradient
  - un paramètre optionnel `step` donnant la taille du pas (défaut: 0.01)
  - un paramètre optionnel `x_0` donnant un vecteur initial pour les paramètres (défaut: initialisation aléatoire)