TP 4 - Classification de chiffres 8*8 avec un modèle de régression logistique multivariée

Olivier Goudet

February 20, 2020

Exercice

Questions

1. Charger le dataset "digits.csv" disponible sur Moodle dans un numpy array nommé "data". Les chiffres (8,8) sont encodés en ligne avec 64 valeurs et les labels sont donnés sur la dernière colonne. Créer la matrice des entrées X et le vecteur des labels y à partir du numpy array "data".

2. Afficher le premier chiffre de la base après un redimensionnement. Pour l'affichage, on pourra utiliser la fonction imshow du module matplotlib.pyplot avec l'option cmap=gray_r pour afficher des nuances de gris.

3. Lors de cette séance, on va implémenter le modèle de régression multivarié pour classer les chiffres du dataset digits. Le modèle que l'on va implémenter est représenté sur le slide 3 du cours 7. Quels sont les valeurs d et k dans ce problème ? Combien y a-t-il de paramètres à apprendre dans ce modèle ?

4. Initialiser aléatoirement une matrice de paramètres nommée "weights" suivant une loi normale de moyenne 0 et d'écart type 0.01.

5. Ajouter une première colonne de "1" à la matrice X des entrées

6. Créer une fonction nommée *output* qui prend en paramètre une matrice d'entrées X et la matrice des paramètres du modèle et renvoie la matrice des sorties du modèle.

7. Tester cette fonction avec la matrice X des entrées du dataset digits afin d'obtenir une matrice f contenant l'ensemble des sorties du modèle. Afficher ensuite cette matrice des sorties du modèle et calculer la somme par ligne de cette matrice pour vérifier que tout est cohérent.

8. Transformer le vecteur y des labels de façon à obtenir une matrice avec un *one hot encoding* ligne par ligne. cf. en haut du slide 5 du cours 7.

9. Créer une fonction nommée *crossentropy* qui prend en paramètre une matrice de prédictions f et une matrice y_one_hot correspondant au "one hot encoding" des labels et qui calcule le terme d'erreur global défini par l'équation 2 du slide 5 du cours 7.

10. Tester cette fonction avec la matrice f précédemment obtenue et la matrice des labels avec le *one hot encoding* obtenu à la question 8.

11. Proposer une matrice de sortie du modèle plausible pour laquelle le terme d'erreur sera très proche de zéro.

12. Créer une fonction nommée *prediction* qui prend en entrée une matrice f contenant l'ensemble des sorties du modèle et renvoie le vecteur avec l'ensemble des classes prédites pour tous les exemples.

13. Tester cette fonction avec la matrice des sorties du modèle précédemment obtenue afin d'obtenir un vecteur y_pred correspondant aux labels des chiffres prédits par le modèle.

14. Implémenter une fonction error_rate qui prend en entrée les labels prédits par le modèle et les vrais labels du dataset et calcule le taux d'erreur global obtenu par le modèle.

15. Tester cette fonction en l'appliquant sur les vecteurs y_pred et y. Quelle valeur obtenez vous ? Est-ce que cela vous semble normal ?

16. Le calcul du gradient du terme d'erreur pour une matrice d'entrée X_, une matrice de sortie f_ et une matrice de label y_one_hot_ est donnée par la fonction suivante. Ajouter la à votre programme.

```
def gradient(X,f,y_one_hot):
    gradient = (np.transpose(X) @ (f - y_one_hot))/X.shape[0]
    return gradient
```

17. Séparer les données avec 70% pour l'entraînement et 30% pour le test de façon à obtenir les matrices X_train, y_train, X_test, y_test. Vous pouvez vous inspirer pour cela d'un bout de code donné en exemple dans le cours.

18. Créer la matrice y_one_hot_train de *one encoding* des labels (comme dans la question 8).

19. Écrire l'algorithme d'apprentissage en vous inspirant des algorithmes vus en cours. Tous les 100 epochs, on affichera le terme d'erreur sur l'ensemble d'apprentissage, ainsi que les taux erreurs sur les ensembles d'apprentissage et de test.

20. A quel moment semble-t-il judicieux d'arrêter l'algorithme ? Pourquoi ? Quelle meilleure performance sur l'ensemble de test obtenez vous ?

21. Bonus : comparer avec la méthode des k plus proches voisins implémentée au dernier TP.