Apprentissage non-supervisé

- Clustering & Réduction de la dimensionnalité -

Youssef MOURCHID (youssef.mourchid@isen-ouest.yncrea.fr)

Objectifs

- Assimiler le principe du clustering et de la réduction de la dimensionnalité
- L'interprétation des résultats (et non pas juste leur affichage)
- Prise en main des bibliothèques de la partie Liens utiles et savoir utiliser, idéalement, leurs documentations

Liens utiles

- Pandas
- Matplotlib
- Scikit-learn
- Numpy

I- Clustering & Reduction de la dimensionnalité

Dans une présentation ppt expliquez le principe de clustering, et puis une explication de l'algorithme de Kmeans, Ensuite expliquez le principe de la Reduction de la dimensionnalité, par la suite définir les deux méthodes PCA et t-SNE avec des exemples pratiques.

II- Affichage des données MNIST

1. Créez un Notebook sous le nom « Clustering_Reduction_Dimensionnalite »
2. Chargez la base de données MNIST depuis le module scikit-learn
3. Transformez le type des labels de string en uint8 en utilisant la fonction astype

1- Clustering

1. Créez un code qui partitionne les données en base d'apprentissage (60000) et base de test (10000).
2. Créez une figure avec des sous-graphiques 3x3 en utilisant matplotlib.pyplot. Parcourir les sous-graphiques et ajouter les 9 premières images mnist en utilisant **matshow()** et Afficher le graphique.

3. Les algorithmes de clustering utilisent presque toujours des données unidimensionnelles. Par exemple, si vous regroupiez un ensemble de coordonnées X, Y, chaque point serait passé à l'algorithme de clustering sous la forme d'un vecteur à une dimension d'une longueur de deux (exemple: [2,4] ou [-1, 4]). MNIST contient des images de taille 28x28 pixels; en conséquence, ils auront une longueur de 784 une fois que nous les transformons en un vecteur à une seule dimension. Convertissez chaque image dans les données d'apprentissage en un vecteur d'une seule dimension, normalisez ces données dans un intervalle entre 0 et 1.

4. Il est temps de commencer le clustering! En raison de la taille du jeu de données MNIST, nous utiliserons l'implémentation mini-batch du clustering k-means fourni par scikit-learn. Cela réduira considérablement le temps nécessaire pour adapter l'algorithme aux données. Créez un code qui permet de faire l'apprentissage sur les données d'entraînement normalisées en utilisant l'objet MiniBatchKMeans du sous module cluster de scikit-learn. Pour le paramètre n_clusters sa valeur est le nombre de classe dans le jeu de données.

5. Le K-means est une méthode d'apprentissage automatique non supervisée; par conséquent, les étiquettes attribuées par notre algorithme KMeans font référence au cluster auquel chaque vecteur a été assigné, et non à l'entier cible réel. Pour résoudre ce problème, définissons quelques fonctions qui prédiront quel entier correspond à chaque cluster :

    a- Définissez une fonction qui associe le label le plus probable à chaque cluster dans le modèle KMeans et qui renvoie un dictionnaire des clusters assignés à chaque label.
    b- Définissez une fonction qui détermine le label de chaque vecteur, en fonction du cluster auquel il a été affecté et qui renvoie les labels prédites pour chaque vecteur.

6. Avec les fonctions définies ci-dessus, nous pouvons maintenant déterminer la précision de nos algorithmes. Puisque nous utilisons cet algorithme de clustering pour la classification, la précision est finalement la métrique la plus importante; cependant, il existe d'autres mesures qui peuvent être appliquées directement aux clusters eux-mêmes, quelles que soient les labels associés. Deux de ces métriques que nous utiliserons sont l'inertie et l'homogénéité.

7. De plus, précédemment, nous avons fait l'hypothèse que K = 10 était le nombre approprié de clusters; cependant, ce n'est peut-être pas le cas. Ajustez l'algorithme de clustering K-means avec plusieurs valeurs différentes de K (ex : [10, 16, 36, 64, 144, 256]), puis évaluez les performances à l'aide de ces métriques.

2- Utilisation de la méthode PCA

L'objectif de cette partie est de pouvoir afficher les instances de la base de données MNIST.

Vous pouvez la faire dans un notebook séparé. Rechargez la base de données MNIST depuis le module scikit-learn.

L'affichage de toutes les instances va être chronophage, par conséquent nous allons afficher que 10000 instances choisies aléatoirement. Créez un code qui permet de réaliser ça en utilisant **random.permutation()** du module Numpy (avec mnist est la variable qui contient la base de données MNIST)

8. Afin de réduire la dimensionnalité des données, appliquez la méthode PCA sur les données (X) en choisissant un nombre de composants de 2. Pensez à utiliser la fonction fi_transform (et non pas fit) pour stocker les données réduites dans une variable.
9. Affichez les données réduites (les données après l'application de PCA) en suivant les étapes suivantes :
    a. importez le sous-module pyplot du module matplotlib
    b. pour changer la taille de la figure, utilisez la fonction "figure" en donnant en argument : figsize=(13,10)
    c. utilisez la fonction "scatter" de matplotlib pour l'affichage en donnant comme argument
        i. les abscisses et les ordonnées des données réduites
        ii. c=y , avec y est le vecteur des labels pour que la barre de couleur soit entre 0 et 9

d. désactivez l'affichage des axes avec la fonction axis tout en donnant comme argument la valeur 'off'
e. activez l'affichage de la barre des couleurs avec la fonction colorbar()

10. Le résultat d'affichage permet de donner une idée claire sur la distribution des instances de MNIST ?

3- Utilisation de la méthode t-SNE

11. Appliquez la méthode t-SNE sur les données (X) en choisissant un nombre de composant de 2
12. Affichez les données réduites en suivant les étapes de la question 7
13. Existe-t-il des chevauchements entre quelques chiffres ? si oui lesquels ?
14. Comparez les résultats d'affichage de la méthode PCA et t-SNE.

III- PCA sur les données MNIST

1- Résultats de RandomForest SANS la réduction de la dimensionnalité des données

15. Divisez la base de données MNIST (**originale**) en base d'apprentissage et base de test comme suit : Les 60 000 premières images composeront la base d'apprentissage et le reste des images constituera la base de test. Pensez à faire pareil pour les labels aussi.
16. Appliquez la méthode de classification RandomForest (avec un n_estimators=100) sur les données d'apprentissage tout en calculant le temps d'exécution nécessaire pour l'apprentissage. Pour ce deuxième objectif, calculez :

    a. avant l'apprentissage du modèle, le nombre de secondes passé depuis le 01/01/1970 en utilisant la fonction "time()" du module time
    b. après l'apprentissage du modèle, le nombre de secondes passé depuis le 01/01/1970 en utilisant la fonction "time()" du module time
    c. le temps d'exécution nécessaire pour l'apprentissage an appliquant la différence des résultats de la question 2.a et 2.b. Pensez à n'afficher que 2 chiffres après la virgule en utilisant la méthode "format()" (ci-dessous un exemple d'utilisation) :
    ```
    format(math.pi, '.2f')   # 3.14
    ```
    d. Évaluez le modèle d'apprentissage sur la base de test en affichant le taux de classification. Pour ce faire :

        i. prédisez les labels de la base de test

ii. utilisez la fonction accuracy_score du sous-module metrics du module sklearn tout en donnant en argument les labels réels de la base de test et les labels prédits

2- Résultats de RandomForest AVEC la réduction de la dimensionnalité des données

17. Appliquez la méthode PCA sur la base d'apprentissage avec un variance ratio de 95%
18. Appliquez à nouveau la méthode de classification RandomForest sur les données d'apprentissage réduites (après l'application du PCA) tout en calculant le temps d'exécution nécessaire pour l'apprentissage. Le temps d'apprentissage est plus rapide que celui du II-1 ? C'est le résultat attendu ?
19. Appliquez la méthode PCA sur la base de test avec un variance ratio de 95% (utilisez la fonction transform et non pas fit_trannsform)
20. Évaluez le modèle d'apprentissage sur la base de test en affichant le taux de classification. Comparez le résultat avec celui de la III-1.
21. L'application du PCA sur les données MNIST était fructueuse pour le temps d'apprentissage et le taux de classification dans le cas de RandomForest ?

3- Résultats de Softmax SANS la réduction de la dimensionnalité des données

22. Appliquez la méthode de classification LogisticRegression sur les données d'apprentissage tout en calculant le temps d'exécution nécessaire pour l'apprentissage.
23. Évaluez le modèle d'apprentissage sur la base de test en affichant le taux de classification

4- Résultats de Softmax AVEC la réduction de la dimensionnalité des données

24. Appliquez à nouveau la méthode de classification LogisticRegression sur les données d'apprentissage réduites (après l'application du PCA) tout en calculant le temps d'exécution nécessaire pour l'apprentissage. Le temps d'apprentissage est plus rapide que celui du III-3 ?
2. Appliquez la méthode PCA sur la base de test avec un variance ratio de 95% (utilisez la fonction transform et non pas fit_trannsform)
3. Évaluez le modèle d'apprentissage sur la base de test en affichant le taux de classification. Comparez le résultat avec celui de la III-3.
4. L'application du PCA sur les données MNIST était fructueuse pour le temps d'apprentissage et le taux de classification dans le cas de Softmax ?

5. L'application du PCA sur les données contribue toujours à accélérer le temps de calcul du modèle d'apprentissage ?