Apprentissage supervisé - Classification

December 15, 2020

Objectifs du Brief

- Assimiler le principe de la classification.
- Faire l'apprentissage de quelques algorithmes pour reconnaître l'écriture manuscrite des chiffres dans les images. Dans ce sens, nous allons utiliser les données de la référence suivante :

  LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
- Savoir évaluer un algorithme de classification
- Manipuler les différents types de classification
- L'interprétation des résultats (et non pas juste leur affichage)
- La prise en main des bibliothèques (Pandas, Matplotlib, Scikit-learn, Numpy)

1 Préparation des données

1.1 Téléchargement de données

Contrairement aux autres projets où la base de données était dans un fichier CSV, dans le présent projet nous allons charger une base de données directement via une fonction de la bibliothèque Scikit-Learn. Cette fonction est nommée "fetch_openml" et elle existe dans le sous-module datasets du module sklean. Utilisez cette fonction pour charger, dans une variable, la version 1 (version=1) de la base de données nommée "mnist_784".

Cette base de données est constituée de petites images représentant des chiffres écrits à la main. Elle est considérée comme le "hello world !" de la classification.

1.2 Information sur les données

1. Le retour de la fonction "fetch_openml" est un dictionnaire (mnist). Affichez ces clés.
2. En utilisant les clés du dictionnaire "mnist", affichez :
   - La taille des données (le nombre des features et la taille de chaque feature). Par convention, les données sont à stocker dans une variable "X".
   - La taille des labels=classes (le nombre des labels). Par convention, les classes sont souvent à stocker dans une variable "y"

By Youssef Mourchid

- Les différentes classes de la base de données
- Une description détaillée de la base de données.

3. A travers cette question, vous avez dû conclure que la base de données MNIST contient 70 000 images en niveau de gris, et chaque image est caractérisée par un features de taille 784 =28*28.
   - Affichez l'image de la première instance de la base de données.
   - Affichez la classe de la première instance
4. Affichez le type des labels (du premier label par exemple)
5. A travers la question précédente, vous avez dû remarquer que le type des labels est une chaîne de caractères. Il est préférable, dans les projets d'apprentissage automatique, d'utiliser des chiffres. Par conséquent, appliquez le casting sur les labels pour les transformer à des entiers.

1.3 Répartition des données

Contrairement aux précédents projets, les données dans MNIST sont déjà partitionnées en base d'apprentissage et en base de test. Les 60 000 premières images composeront la base d'apprentissage et le reste des images constituera la base de test. Écrivez un code qui met en exergue cette répartition tout en stockant les données de test/d'apprentissage et les classes de test/d'apprentissage dans 4 variables.

2 Apprentissage d'un classifieur binaire

Comme indiqué dans les objectifs, ce projet vise à classifier les chiffres. Une des solutions est d'utiliser un classifieur binaire qui est apte d'identifier que l'image représente bien le chiffre que nous cherchons ou non. Par exemple, nous nous focaliserons sur la reconnaissance du chiffre 5

2.1 Apprentissage des données

1. En utilisant les labels d'apprentissage et de test, écrivez un code qui stocke dans deux variables (une première pour la base d'apprentissage et une autre pour la base de test) distinctes True si le classe de l'instance est 5 et False Sinon. De ce fait, nous traitons une classification binaire.
2. Pour l'apprentissage des données, nous allons utiliser le classifieur Stochastic Gradient Descent (SGD). Utiliser ce modèle pour la prédiction sur un chiffre (exemple 5).

2.2 Évaluation du modèle d'apprentissage sur les données d'apprentissage

La mesure de performance d'une méthode de classification est souvent plus délicate qu'une méthode de régression. Cela est dû au nombre de mesures de performance existantes dans la littérature.

2.2.1 Taux de classification

1. Écrivez un code qui répartit la base d'apprentissage en base d'apprentissage et en base de validation en utilisant la méthode 3-fold cross-validation. Optez pour la valeur "accuracy" pour l'argument "scoring" pour afficher :
   - le taux de classification (accuracy) de chaque fold
   - la moyenne des taux de classification
2. Dans cette question, vous allez créer un classifieur simple qui classifie toutes les images de MNIST comme "non-5". Créez une classe Never5Classifier qui hérite de la classe BaseEstimator. la classe BaseEstimator existe dans le sous-module "base" du module "sklearn"
3. Dans la classe Never5Classifier, créez :
   - Une méthode fit qui prend en argument les données et les labels. Cette méthode ne va rien retourner et par conséquent va contenir que le mot-clé "pass". Vous allez implémenter cette méthode car l'héritage de la classe BaseEstimator l'exige
   - Une méthode predict qui prend en argument les données et retourne une structure de données ayant la taille des données et qui contient que la valeur False (= non-5). Utilisez la fonction "zeros" du module "numpy" avec un "dtype=bool"
4. Créez un objet de la classe Never5Classifier
5. Testez le classifieur en utilisant une validation croisée de type 3-fold cross-validation. Optez pour la valeur "accuracy" pour l'argument "scoring" pour afficher : le taux de classification (accuracy) de chaque fold; La moyenne des taux de classification.

2.2.2 Matrice de confusion:

1. En utilisant le modèle déjà construit du classifieur SGD, prédisez les classes des données d'apprentissage. Pour ce faire, utilisez la fonction "cross_val_predict" du sous-module "model_selection" du module "sklearn". Optez pour une validation croisée de type 3-fold cross validation.
2. Affichez la matrice de confusion du modèle d'apprentissage. Veillez à bien interpréter la matrice de confusion.

2.2.3 Précision et rappel:

1. Calculez la précision, le rappel et le score F1 du modèle d'apprentissage
2. Pour classifier les instances, le classifieur SGD calcule un score en se basant sur sa fonction de décision. Si le score est supérieur à un seuil, il affecte la classe positive à l'instance sinon il affecte la classe négative. Affichez les scores des différentes instances de la base d'apprentissage via la fonction "cross_val_predict". Optez pour une valeur de "decision_function" pour l'argument "method" et une validation croisée de type 3-fold cross-validation.
3. Calculez les précisions et les rappels de chaque instance en utilisant la fonction "precision_recall_curve"

4. Tracez la courbe des précisions/rappels. L'axe des abscisses doit contenir les rappels et l'axe des ordonnées doit contenir les précisions. Interprétez les résultats.

2.2.4 Courbe ROC:

1. Calculez le taux de faux positifs (tfp), le taux de vrais positifs (tvp) et les seuils utilisés pour classifier les données d'apprentissage. Pour ce faire, utilisez la fonction "roc_curve" du sous-module "metrics" du module "sklearn"
2. Tracez la courbe des ROC. L'axe des abscisses doit contenir les taux de faux positifs et l'axe des ordonnées doit contenir les taux de vrais positifs. Ajoutez des titres à ces deux axes et activer le mode "grid" avec la fonction grid de matplotlib
3. Calculez l'AUC (Area Under the Curve) du modèle d'apprentissage. Pour ce faire, utilisez la fonction "roc_auc_score" du sous-module "metrics" du module "sklearn".

3 Apprentissage d'un classifieur multi-classes

Dans cette partie, nous allons apprendre un classifieur à classifier les 10 classes (les chiffres de 0 à 9) de la base de données MNIST.

3.1 Apprentissage des données

1. En se basant sur le classifieur SGD, faites l'apprentissage du modèle en se basant sur toutes les instances (images) d'apprentissage. Ainsi le modèle va apprendre à partir de toutes les classes de la BD et non pas juste à partir de deux classes comme c'était le cas dans la partie II de TP.
2. Via le modèle d'apprentissage bâti, prédisez la classe de la première instance de la base de données (celle de la question 3). La prédiction va être, cette fois-ci, un chiffre de 0 à 9.
3. En utilisant l'objet qui instancie la classe SGDClassifier :
   - Affichez, via la méthode "decision_function", les 10 scores de décision utilisés par la méthode SGD pour classifier la première instance de la base de données.
   - Affichez, via l'attribut "classes_", les différentes classes utilisées par le classifieur. Qu'est ce que vous remarquez ?

3.2 Évaluation du modèle d'apprentissage sur les données d'apprentissage

3.2.1 Taux de classification

1. Écrivez un code qui répartit la base d'apprentissage en base d'apprentissage et en base de validation en utilisant la méthode 3-fold cross-validation. Optez pour la valeur "accuracy" pour l'argument "scoring" pour afficher : le taux de classification (accuracy) de chaque fold; la moyenne des taux de classification
2. Une des techniques d'amélioration des taux de classification est le "Scaling". Vérifiez ceci en appliquant une standardisation sur les données d'apprentissage avec la classe StandartScaler.

3. Évaluez le modèle sur les nouvelles données d'apprentissage avec une validation croisée de type 3-fold cross-validation. Les résultats sont-ils meilleurs que ceux avant le Scaling?

3.2.2 Matrice de Confusion

1. En utilisant le modèle déjà construit du classifieur SGD, prédisez les classes des données d'apprentissage.
2. Affichez la matrice de confusion du modèle d'apprentissage. Pour ce faire, utilisez la fonction "confusion_matrix" du sous-module "metrics" du module "sklearn". La taille de la matrice de confusion, cette fois-ci, est 10x10 car on traite un problème de classification de 10 classes. Interprétez les résultats.