Reconnaissance des digits audio

22 mars 2022

Auteurs :

Fabio Cassiano

Table des matières

1 Acquisition du DataSet

Le DataSet utilisé dans cette étude a été acqui manuellement par le biais d'un programme qui nous a été fournis. Il est composé de *features* extraite d'enregistrements audio, qui sont acquis grace à la fonction *sounddevice.rec*. Les enregistrements sont réalisé sur une durée de 2 secondes, avec une fréquence d'échantillonnage de 48000 Hz. Des *features* sont ensuite extraites de ces enregistrement à l'aide de la fonction *python_speech_features.mfcc*.

Pour chaque enregistrement on doit prononcer un chiffre qui s'affiche, les chiffres vont de 0 à 9. Pour notre étude 20 enregistrements ont été réalisées pour chaque chiffres. Nous permettant alors d'avoir un DataSet de dimension (200, 12), soit 12 features pour 200 instances.

2 Prétraitement du DataSet

Dans cette étape on s'intéresse aux données collectées, on vérifie normalement s'il n'y a pas de valeur manquante (*NaN*), si l'on a bien le même nombre d'instance pour chaque label, etc ... Dans notre cas d'étude il n'y a aucune valeurs manquante car nous avons enregistré nous même notre jeu de données.

Après avoir prétraité le jeu de données, il est séparer en deux pour ainsi obtenir un jeu d'entrainement (*train*) représentant 80 % du DataSet original, et un jeu de test correspondant aux 20 % restant.

3 Evaluation de différents modèles de classification

Le jeu d'entrainement est ensuite utilisé pour entrainner différents modèles de classification. Les modèles que nous avons décidé de tester sont les suivants :

— Les machines à support vectorielle (SVM)
— Les arbres des décisions
— La forêt aléatoire (Random Forest)
— L'amplification de Gradient
— Les K plus proche voisin (KNN)
— Le classifier XGboost
— Le perceptron multi-couche (MLP)

Pour évaluer ces différents modèles, une pipeline a été mise au point pour chacun de ces modèles avec un prétraitement basé sur la standardisation (*StandardScaler*). Pour chacun des modèles plusieurs paramètres ont été testés, en réalisant un *GridSearch*. Pour automatisé le traitement, les différentes pipeline ainsi que les paramètres associés ont été stockés dans une liste afin de réalisé une boucle qui executera les différents *GridSearch* au fur et à mesure.

4 Résultats obtenus

Les *accuracy* obtenus sont en général plutôt bonne sur le jeu de données de test. Le score le plus faible est de 62.5 % pour les arbres de décisions, et le meilleur résultat est obtenus avec le modèle MLP avec un score de 95.0 %.

Ce modèle est ensuite enregistrer au format *.joblib*, avec les meilleurs paramètres, à l'aide de la fonction *joblib.dump*. Permetant ensuite sont utilisation dans la fonction temps réel *tools.rec()*, après une légère modification du code afin d'importer le modèle.

5 Organisation de l'étude

La totalité de l'étude a été effectué sur un unique *jupyter notebook*, divisé en 3 grandes parties :

1. La collection du DataSet
2. Le traitement des données et entrainement des modèles
3. Application du meilleur modèle en temps réel