Rapport du challenge SD211

Réalisé par: SOUDANI Jawher, FERJANI Ahmed et YAHMED Taycir

INTRODUCTION

Dans ce challenge , notre tâche est de faire "classification de la scène acoustique". En particulier, compte tenu d'un signal audio d'environ 30 secondes, l'objectif est de trouver son "contexte" ou l'environnement correspondant au signal. Ces environnements peuvent être très différents comme une plage, un restaurant ou une station de métro. Au total, nous examinerons 15 classes différentes en utilisant la libraire librosa et en cherchant le meilleur classifieur parmi plusieurs utilisés

Mel-Frequency Cepstral Coefficients (MFCC)

Les MFCC représentent collectivement le spectre de puissance à court terme d'un Échantillon sonore. Ceux-ci sont largement utilisés comme fonctionnalités dans des applications comme la reconnaissance de la parole , la reconnaissance des locuteurs, la classification des genres et d'autres mesures similaires de recherche d'information musicale. Ceux-ci sont sensibles au bruit et pas la parole, ils sont donc plus appropriés d'être appliqués ici dans le cas de la classification des scènes.
Les MFCC sont obtenus comme suit:

1. Prenez la Transformée de Fourier à Temps Court d'un échantillon audio. (La motivation derrière la FFT est qu'un signal de parole varie avec le temps, mais est stationnaire sur de courts segments)
2. Placez le spectre de puissance obtenu ci-dessus sur l'échelle de fusion en multipliant le spectre par un mel-scaled triangle .
3. Prenez le log du mel-spectrum obtenu ci-dessus
4. Prendre la transformée cosinée discrète (discrete cosine transform DCT) du log spectrum pour obtenir le nombre requis de coefficient.
5. Les MFCC sont la sortie des amplitudes du DCT.

Dans ce challenge ,on a utilisé la librairie **librosa** pour extraire les features.

Le système de soumission proposé utilise les fonctionnalités de MFCC avec les paramètres bas qui doivent être transmis au classifieur qu'on va choisir
Les paramètres MFCC utilisés sont présentés ci-dessous:

Parameter	Value
No. of mfcc coeff. No. of mel bands FFT length hop length	40 defaults 4096 4096

Remarques:

- ➔ On a augmenté le nombre de coeff. MFCC pour obtenir un nombre plus grand de features, vu que les paramètres par défaut ne donnent pas de bonnes performances sur la dataset de validation.
- ➔ FFT_length = hop_Length pour éviter le chevauchement des trames. (Overlapping)
- ➔ Augmenter le nombre de FFT_length (FFT window size) pour avoir évaluer le maximum de coefficients dans une seule trame et donc avoir plus de performance en termes de précision.

Prétraitement

Pour permettre l'apprentissage sur des données de bonne qualité, on a essayé quelques

techniques de prétraitement des données. Certaines de ces techniques, ont été abandonnées faute de bon résultats, i.e. amélioration de la performance.

Débruitage
Cette tâche a été réalisée par un filtre de Wiener, permettant de réaliser une déconvolution mathématique pour éliminer ou atténuer une partie des bruits dans le signal.

Randomisation des données
Parmi les expérimentations qu'on a faites sur le formatage des données, on compte la concaténation des données horizontalement (No-fusion): Pour chaque échantillon, on transforme la matrice retournée par librosa en vecteur unidimensionnel. Dans ce cas précis, l'ordre ne compte plus, nous permettant de randomiser les données.

Normalisation des données
Vu que certains algorithmes sont sensibles au ordre de grandeur des variables, on a normalisé les données.

Elimination d'outliers
Certains échantillons ont été éliminés de données d'apprentissage/validation afin d'augmenter -éventuellement- la performance.

PCA
On a réalisé une décomposition en composantes principales afin de limiter le nombre de variables à un ensemble très représentatif de la variance des variables.

Importance relative des variables
L'une des méthode qu'on a aussi essayé d'appliquer c'est limiter l'ensemble des prédicteurs à un ensemble de variables très informatives. Cette sélection a été réalisée par deux méthodes:

- Une élimination récursive des variables permettant de déterminer le bon nombre de prédicteurs et d'en sélectionner les meilleures (les plus efficaces).

- Une mesure de l'importance des variables retournée par le modèle des forêts aléatoires (random forest).

Classifieurs utilisés

QDA : Quadratic classifier :
Ce classifieur utilise des surfaces quadratiques pour séparer les différentes classes.
On s'est servi de la bibliothèque Sickit-Learn pour l'utiliser.
Ce classifieur n'a pas d'hyper paramètres à régler. Sa performance était de 83% sur le jeu de données de Test.

SVM: Support Vector Machine Classifier:
On a utilisé ce classifieur avec notre jeu de données, après normalisation et réduction de dimension avec PCA.
On a fait un GridSearchCV pour déterminer les bons paramètres à utiliser.

Paramètres	Valeurs
kernel	rbf
C	0.1
gamma	1e-2

Performances: 68~72% sur la Dataset de validation et de 79~82% sur la Dataset de Test.

GMM classifier :
Vu que c'est parmi les classifieurs les plus recommandé pour des problématiques telles que la présente, on a entraîné un modèle de mélange de gaussiennes avec un nombre de composantes égal à 15 (nombre de classes). On ne note que c'est modèle a un très long temps de calcul et donne une performance inférieure à celle de MLP classifier.

Voting classifier :
Une des premières idées qu'on a essayé est l'entraînement de plusieurs algorithmes et la comparaison de leurs performances. Ces algorithmes faisait surtout partie de ceux cités dans des papiers scientifiques traitant la classification de scènes acoustiques. Ensuite, on a utilisé une méthode d'ensemble permettant de consolider les résultats des différents classifieurs, selon la probabilité prédite par chacun d'entre eux (soft voting). En ce qui concerne les poids utilisé dans ce vote, on associe chaque classifieur à un poid

proportionnel à sa performance sur l'ensemble de validation.
Ce méthode ne permet pas d'améliorer les résultats car les classifieurs de performance inférieure à celle de MLP classifier, induisent le modèle à l'erreur pour certains échantillon, ce qui réduit la performance du modèle. On conclut alors qu'il vaut mieux dans ce cas utiliser des classifieurs de performance comparable.

DNN classifier (keras):
Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow, CNTK ou Theano.
La structure de données de base de Keras est le modèle, un moyen d'organiser des couches. Le modèle le plus simple qu'on a utilisé est le modèle séquentiel, une pile linéaire de couches
Après l'évaluation du modèle par K-Fold cross validation , on a utilisé ces paramètres pour la prédiction finale

Parameter	Value
Model inputs Hidden nodes Activation loss optimizer batch_size	Sequential 40 300 relu categorical_crossentropy adam 2048

La précision obtenu sur la test data était 83% et sur la validation data était 72%.

Remarque:

Sous certaines conditions, on a remarqué que le modèle établi par keras sur-apprend les données d'entraînement. Par conséquent, on a utilisé la régularisation par "dropout". Cette technique proposée dans ce papier, consiste à éteindre et rallumer les neurones aléatoirement (on a pris: dropout rate = 0.5 ⇒ un neurone sur deux), dans le cadre d'entraînements successifs, rendant le réseau plus robuste et capable de mieux généraliser les concepts appris.

Autre expérimentation
On a également essayé une méthode basée sur la régression sur l'ensemble des prédicteurs, puis pour sélectionner le label on prend l'entier le plus proche du résultat retourné. Cette méthode nous a donné de très mauvais résultats.

Classifieur utilisé pour la prédiction finale

MLP classifier (sklearn) :
Methodology :

- Chargement des données
- Extraction des features par mfcc en utilisant la méthode late Fusion
- normalization par Standard Scaler
- Features réduction par PCA
- k cross validation pour chercher les bons paramètres sur les données de validation.
- Ttraining du classifieur sur train_data concaténé avec validation_data vu que le nombre des données n'est pas suffisant pour avoir des bons scores de précision.
- Réduction de la variance par le bagging classifier.
- Prédiction des classes finaux par la méthode de max number vote et non pas max prédiction probability vote.
- Enregistrement du tableau des classes en fichier texte.

Paramètres utilisés et justification du choix:

Après avoir fait un GridSearchCv sur le MLP classifier, on a eu comme valeurs maximisant le score de cross validation celles qui suivent:

Parameter	Value
Layers Nodes per Layer alpha tolérance	1 300 0.1 0.001

Résultats obtenus
Moyenne des Précisions obtenus 87~89% sur le jeu de données de Test.