Challenge Report

Anas Barakat, Bilal Mouhib

25/06/2017

SCORE : 89.2617449664 - Classement : 6èmes ex aequo

Le code utilisé avec la solution finale et les codes testés sont disponibles dans le même dossier que ce rapport. Une description des différents fichiers .py fournis est donnée à la fin de ce rapport. Seul le code qui a permis d'augmenter le score ou qui présente des résultats intéressants a été fourni en pièce jointe. Les autres codes sont à notre disposition mais ne sont pas joints.

1 Démarche de résolution

1.1 Démarche générale

— Extraction des features et construction des matrices X et des vecteurs de labels y pour le train, la validation et le test et scaling.
— Classification multi-classes

1.2 Optimisation

Nous avons joué sur plusieurs pistes d'amélioration en les testant :
— Early fusion vs late fusion vs une combinaison des 2 (combinaison retenue).
— Concaténation des datasets d'entrainement et de validation pour avoir une plus grande base d'apprentissage. Le score a alors progressé de manière conséquente.
— Paramètre de fusion des lignes des MFCC calculées sur lesquelles on moyenne.
— Rajout d'autres features aux MFCC seules comme Mellogspectrum, Tempo, MFCC delta de degrés 1 et 2.
— Choix des hyperparamètres des algorithmes (tuning) avec des GridSearch.

1.3 Algorithmes testés

Pour traiter le problème de classification du challenge, nous avons testé plusieurs algorithmes (voir tous) de la bibliothèque sklearn et le réseau de neurone MLP avec la bibliothèque Keras et PyBrain. Nous avons aussi implémenté des

GMM en fittant un GMM par classe puis en faisant la prédiction avec chacun d'entre eux puis en retenant la classe qui donne le maximum likelihood, sans succès (les performances sont très faibles). Nous avons retenu QDA (Quadratic Discriminant Analysis, MLP et Random Forest qui donnent des scores de plus de 75 voir 80 % sans optimisation (tuning des paramètres)). Pour avoir une idée sur les algorithmes de classification les plus performants sur le training set, nous avons utilisé un script qui fait une validation croisée pour les comparer.

1.4 Techniques supplémentaires de feature engineering

Faire de la feature selection en utilisant l'attribut feature importances du Random Forest pour récupérer les variables explicatives les plus significatives pour la classification puis construire à partir de celles-ci un nouvel estimateur n'a pas permis d'améliorer le score au début de notre démarche mais a finalement un tout petit peu amélioré le score à la fin pour passer de 88 à 89 %. La réduction de dimension en utilisant par exemple l'analyse par composante principale PCA n'a pas porté ses fruits et n'a fait que baisser les performances de classification.

1.5 Evaluation du modèle

L'utilisation de la validation croisée avec cross val score de sklearn nous a permis d'évaluer les modèles en utilisant le validation set.

2 Solution retenue

2.1 Etapes de l'algorithme

— Extraction des features MFCC et delta MFCC de degré 1 et de degré 2 pour augmenter le nombre de features à 60
— Combinaison entre early fusion et late fusion avec un paramètre Fus représentant les nombres de lignes à fusionner (moyenner). (Fus = 11 lignes à combiner, optimal) La fonction "uniqueFus" dans le code joint transforme la prédiction sur le Xtest qui contient plusieurs lignes pour un même fichier audio à une prédiction par fichier audio en utilisant un vote majoritaire.
— Combinaison des Xvalidation et Xtrain pour avoir un data set d'entrainement plus grand (nombre de samples plus grand)
— Scaling des matrices (preprocessing.scale de sklearn) pour ne pas biaiser l'apprentissage.
— 1ère classification à l'aide d'un MLP (Multi-Layer Perceptron) qui donne un score de 85 % minimum après choix des bons hyperparamètres
— Autre classification avec QDA (Quadratic Discriminant Analysis) (de même)
— Ajout dans le train des samples (avec les labels prédits) du Xtest sur lesquels les 2 algorithmes précédents sont d'accord (même prédiction)

pour augmenter le data set d'entraînement, ce qui est fiable étant donné les bons scores.
— Feature selection en utilisant "feature importance" du RandomForest (suppression des 10 features les moins significatives)
— Vote majoritaire des trois algorithmes (QDA, MLP et RandomForest qui ont tous les trois des scores d'au moins 85 pourcents) avant la fusion finale.

2.2 Réalisation

Le code correspondant est donné dans le fichier Python joint.

2.3 Pistes d'amélioration testées

Nous avons testé d'entrainer un classifieur binaire pour chaque classe et de répercuter les modifications de ce classifieur sur la prédiction que nous avions avant. La fonction codée appelée "correction.py" illustre cette méthode et figure parmi les fichiers python ci-joints.

2.4 Pistes d'amélioration non testées

Nous avons pensé à séparer les fichiers de test selon des groupes de classes de caractéristiques proches comme par exemple Bus et Tram, City center et residential area. En effet, en écoutant certains audios, on n'arrive nous même pas à les distinguer. De plus l'idée précédente testée (section précédente) nous a mis sur cette piste. En comparant ce que donne le classifieur binaire et notre prédiction sur les différentes classes, on constate que les modifications proposées par le classifieur binaire correspondent à des lieux de caractéristiques proches parfois difficiles à distinguer.

2.5 Description des fichiers Python

— compare_Algo.py : code de test des différents algorithmes par validation croisée.
— dataAugmentFusParam.py : pour l'extraction de features, la construction des matrices fusionnées puis concatenation des données du train et de validation, définition de uniqueFus qui permet de passer de la prédiction à plusieurs lignes pour le même fichier audio à la prédiction finale de taille 298 à soumettre sur le site, 1ère prédiction MLP
— super_train.py : 2ème prédiction en utilisant QDA, comparaison de QDA et MLP et rajout des lignes de Xtest (sur lesquelles les 2 algorithmes sont d'accord) dans le train. Xsuper_vrai, ainsi construite, est la matrice qui sera utilisée pour les prédictions finales avec ysuper_vrai correspondant.
— majorityVote3clf.py : feature selection en utilisant RandomForest, combinaison des prédictions des 3 classifieurs (RF,QDA,MLP)
— keras.py : réseau de neurones MLP avec la bibliothèque keras.

— correction.py : correction avec classifieur binaire (une classe contre tous) , démarche décrite en section 2.3

2.6 Remarque

Nous nous sommes inspirés de plusieurs articles de recherche sur l'acoustic scene classification :

— Acoustic Scene Recognition with Deep Neural Networks (DCASE challenge 2016), Dai Wei, Juncheng Li, Phuong Pham, Samarjit Das, Shuhui Qu

— Acoustic scene classification : an evaluation of an extremely compact feature representation, Gustavo Sena, MafraNgoc Q. K. Duong, Alexey Ozerov, Patrick Pérez, Budapest 2016