Kaggle Project: Classification for Audio Samples

1 Project description

Human ear can distinguish different sound, computer can't do though it has superb computing power. To help computer "understand" and "distinguish" different sound, people should transform audio file into data, so computer can read it. Then people can input some labeled data, or training data, into the computer. Combined with different algorithms, computer can figure out some regulation, or so-called model to help people predict unlabeled data. In this project, I use urban sound8k dataset to do the sound classification. To do classification, I use some common classification algorithms, such as KNN, Naïve Bayes, SVM etc. Among all models, RBF SVM and random forest give two of highest average test score.

2 Data description

The data is originally from Kaggle. This dataset contains 8732 labeled sound excerpts (<=4s) of urban sounds from 10 classes. The 10 classes are: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. Number of observations are shown in the table below.

Table 4-1

class	**Number of observations**
air conditioner	1000
car horn	429
children playing	1000
dog bark	1000
drilling	1000
engine idling	1000
gun shot	374
jackhammer	1000
siren	929
street music	1000
total	**8732**

The number for each class are imbalanced, which might cause some problem for analysis.

Notes from Kaggle:

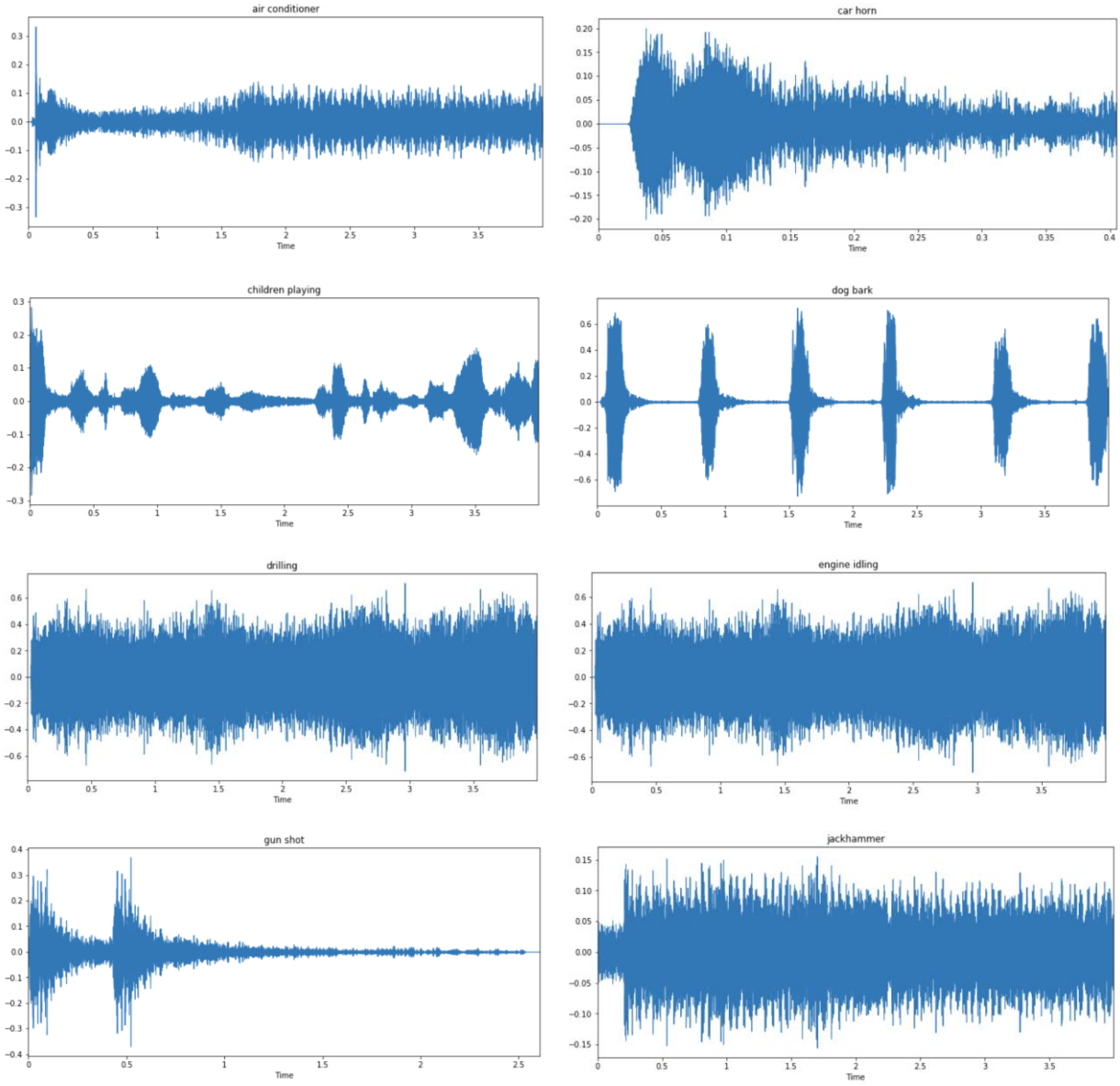
1. Don't reshuffle the data. Use the predefined 10 folds and perform 10-fold cross validation.

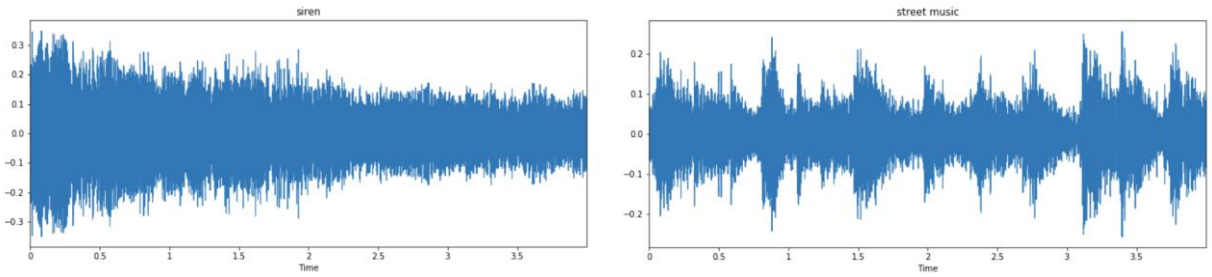
2. Don't evaluate just on one split.

3 Features exaction

To finish classification problem, we need to first extract features from audio files. The data provided of audio cannot be understood by the models directly to convert them into an understandable format feature extraction is used.

In the feature exaction part, I use *librosa* package to load data and extract information for audio data. To know more about the urban sound8k data, I draw the wave plot of sounds from different class.





Graph 4-1

I read some previous classification report on the same dataset and found some useful code to extract audio feature from wav file. Finally I got 195 features for every audio file.

MFCCS: Mel-frequency cepstral coefficients. This is because the data that the mfcc returns is not always the same length, and since the model expects the data to be the same length, so I compute the mean mfcc of each feature inside the mfcc.

chroma: Compute a chromagram from a waveform or power spectrogram

mel: Compute a mel-scaled spectrogram.

contrast: Compute spectral contrast

tonnetz: Computes the tonal centroid features (tonnetz)

spectral_centroids: number of ”center of mass”

zero_crossings: Compute the zero-crossing rate of an audio time series.

4 Dataset building

There are two functions used in dataset building part.

extract_feature: used for extract features from audio file, which takes in file name(string) and returns a feature record.

load_audio: used for transfer multiple audio files to a feature array, which takes in file list and file extension, then returns feature array.

5 Model selection

It is a classification task, so I apply KNN, Naive Bayes, Neurol Network, Support Vector Machine (SVM), and some Ensemble classifiers on this dataset.

5.1 KNN

The k-nearest neighbor algorithm (KNN) is a non-parametric method used for classification. The basic idea behind KNN is very simple—data is supposed to alike its neighbor—so with KNN, for every observation in test set, find k nearest neighbors to predict the correct class.

n_neighbors: 3, 5, 7, 9, 11, 15, 17, 21, 23, 25
weights: 'uniform', 'distance'
metric: 'euclidean', 'manhattan'

Best model:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',metric_params=None, n_jobs=None, n_neighbors=5, p=2,weights='distance')

5.2 Naïve Bayes

Based on Bayes theorem, Naïve Bayes calculates the probability of test set belonging to certain class. Naïve Bayes (Gaussian)don't have much parameters to tune. So I just use the default Naïve Bayes setting to do the classification.

5.3 Perceptron

max_iter: 50,100,200,500,1000,2000
eta0: 5,10,15….100
Shuffle: True, False

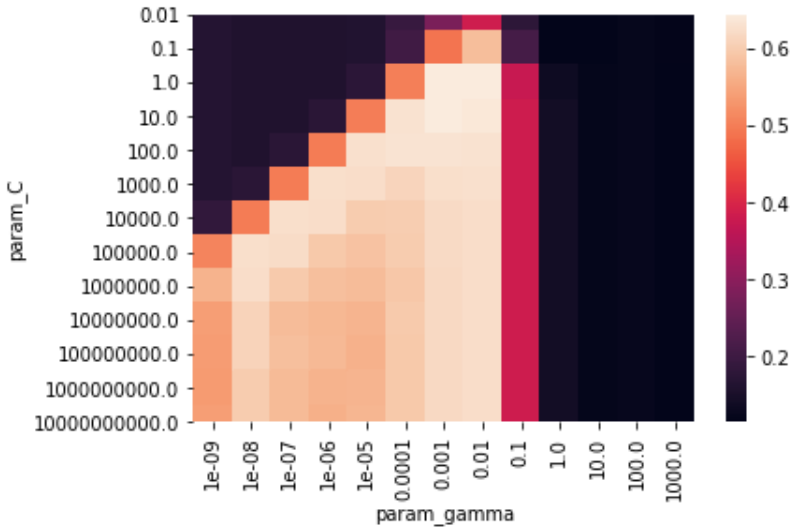
One of the best models is:
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=5,fit_intercept=True, max_iter=50, n_iter_no_change=5, n_jobs=None,penalty=None, random_state=814, shuffle=True, tol=0.001,validation_fraction=0.1, verbose=0, warm_start=False)

5.4 RBF SVM

Two important parameters in RBF SVM is C and gamma. The C parameter trades off correct classification of training examples against maximization of the decision function's margin. The gamma parameter is the inverse of the radius of influence of samples selected by the model as support vectors.

Using grid search, I set C ranged from 0.01 to e^{10} and gamma ranged from e^{-9} to 1000. It turns out that the model gets best score at 0.6441 when C equals to 10 and gamma equals to 0.001.

And I get the heatmap below showing the score with light color refering to high accuracy and dark color referring to low accuracy.



Graph4-2 Heatmap for RBF SVM

5.5 Random Forest

Random forest is an ensemble classifier, I first try default setting of sklearn random forest on my dataset. I got 0.99 score on training data and 0.56 on test data, which is pretty good. Then I tune parameter on following range with grid search:

n_estimators: 5,10,20,50,100,150,200,300,500

criterion: 'gini','entropy'

max_depth: 5,20,50,80,100,150

min_samples_split: 2,5,10,15,20,50,100,150,200,300

Best model:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=50,max_features=None, max_leaf_nodes=None,min_impurity_decrease=0.0, min_impurity_split=None,min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=300,n_jobs=None, oob_score=False, random_state=814,verbose=0, warm_start=False)

5.6 Models comparison

After grid search parameter for classifiers, I use 10-fold cross validation to compare different models.

Table 4-2 Models comparison

Classifier	Average training score	Average test score	Time
KNN	1.0	0.5548	4.47min
Naive Bayes	0.3986	0.3507	0.52s
Decision Tree	0.7181	0.4960	36.19s
Perceptron	0.7339	0.5419	3.11min
RBF SVM	0.9926	0.6441	119.79min
Random forest	1.0	0.6359	More than 1 day

Average training score and average test score show cross validation score on training set and test set respectively. Time includes time for grid search and time for cross validation. From Table4-2, it is obvious that overall prediction result is not good enough, with RBF SVM and random forest have highest accuracy at 0.64. Naïve Bayes model has the lowest accuracy at test dataset, which gives a baseline of accuracy for this project. KNN model is overfitting since its average training score is too high compared to its average test score.

So I would choose RBF SVM model to do the classification in urban sound 8k dataset given it has the highest accuracy and relatively lower time in training model.

6 Conclusion and limitation

In this report, I try 6 classifiers on urban sound 8k dataset to do sound classification by cross validation. Among all models, RBF SVM and random forest provide two of highest average test score of around 0.64 and training score of around 1.0. Given random forest takes more than a day to train the model, RBF SVM is a good chose to do the classification.

There are some limitations in this report. First, I do not use PCA to reduce the number of features, resulting long training and testing time. However, I did try PCA in some algorithms, there is 0.01 increase in average testing dataset. Second, though I extract 195 features in this data, I did not do feature engineer to determine which features are important and which aren't.