Musical Instrument Classification

Hyunsung Cho

April 4, 2019

1 Algorithm Description

The final algorithm and parameters used in the algorithm were determined following multiple iterations of testing. In this section are presented the algorithm and parameters resulting in the best classification test accuracy. The program consists of the dataset and three python scripts (`feature_extraction.py`, `feature_summary.py`, and `train_test.py`) which are explained in this section one by one.

1.1 Dataset

The program uses a subset of the NSynth dataset made by Google Magenta Project. It is an audio dataset containing a range of musical notes from different musical instruments. The used subset consists of 10 classes of musical instruments and is split into training, validation and test sets. There are 100 audio samples per instrument in the training set, and 20 in the validation and test sets resulting in a total of 1200 audio samples.

1.2 Feature Extraction

As the first step towards instrument classification, the program performs feature extraction in `feature_extraction.py`. It loads each audio file of the dataset and extracts features for each. Table 1.1 summarizes all audio features used for training the classifier.

Timbre Features

The program extracts features that represent timbre texture after applying the short time Fourier transform (STFT) on the time domain audio stream from the dataset. The STFT is run with a sampling rate of 22050 Hz and 18 different combinations of FFT window size, hop length, and DCT type using `librosa.core.stft`. Details on these combinations and results will be discussed later in the next section. The program extracts the following timbre features for classification:

Audio Features		Extracted Features
Timbre features	Zero-Crossing Rate	Zero-crossing rate
	Spectral Statistics	Spectral centroid Spectral rolloff Spectral flatness Spectral bandwidth Spectral contrast
	MFCC	MFCC Delta Double delta
	MFCC with Constant-Q Transform	MFCC Delta Double delta
Pitch/Harmony features	Chroma	STFT Constant-Q chromagram Chroma Energy Normalized (CENS)

Table 1: Features used in the classifier

1) Zero-Crossing Rate (ZCR): ZCR refers to the number of zero crossings in a given time frame. A zero crossing occurs when the amplitude of the time-series audio signal passes through the value of zero. ZCR is computed using `librosa.feature.zero_crossing_rate` function with frame length of 1024 and hop length of 512.

2) Spectral centroid: Spectral centroid is the center of gravity of the spectrum. This feature is used to classify musical instruments based on the brightness of sounds. This is computed using `librosa.feature.spectral_centroid` function with the magnitude spectrogram based on the STFT as a parameter.

3) Spectral rolloff: The algorithm sets the parameter *roll_percent* to 0.95 and computes the roll-off frequency of the magnitude spectrum by using the `librosa.feature.spectral_rolloff` method. The computed value represents the frequency under which 95% of the spectral energy is concentrated in.

4) Spectral flatness: Spectral flatness is a measure of the noisiness (as opposed to tonality) of the spectrum. The value equals the ratio between the geometric and arithmetic means given the magnitude spectrum. The algorithm uses `librosa.feature.spectral_flatness` method. This is a useful feature in musical instrument classification. For example, wind instruments tend to have greater tonality than instruments like the piano.

5) Spectral bandwidth: Spectral bandwidth or spectral spread is a measure of how spread out the spectrum is. The p'th-order (p=2) spectral bandwidth is calculated using `librosa.feature.spectral_bandwidth`.

6) Spectral contrast: Spectral contrast was first proposed by Jiang et al. as a feature that "considers the spectral peak, spectral valley and their difference in each sub-band" [Jiang, Lu, Zhang, Tao, and CaiJiang et al.2002]. The algorithm computes the spectral contrast for six frequency sub-bands by using the `librosa.feature.spectral_contrast` method. The function outputs seven spectral contrast values corresponding to each octave-based frequency sub-band in an array.

7) MFCC: Mel-Frequency Cepstral Coefficient (MFCC) is used to space the frequency bands according to the human auditory system. It obtains the mel-scaled spectrogram using `librosa.feature.melspectrogram`, performs log compression, and computes 13 MFCCs through `librosa.feature.mfcc` with DCT type-2. The MFCCs represent the spectral envelop of an audio frame. In order to add temporal dynamics, 13 delta of MFCCs and 13 double delta of MFCCs are further calculated and used as features in `feature_summary.py`.

8) MFCC with Constant-Q Transform: Other features listed above are traditional features used in music processing, but this is an experimental feature. It represents MFCCs generated using the constant-Q transform of the audio signal instead of the mel-scaled spectrogram. The algorithm calculates the constant-Q transform using `librosa.core.cqt` and uses the magnitude as the parameter for `librosa.feature.mfcc`. Similar to the above MFCCs, 13 MFCCs, 13 delta, and 13 double delta were computed accordingly. This experimental feature accounts for 1.5% increase in the best classification test accuracy compared to the classification without using this feature.

Pitch/Harmony Features

Chroma features are extracted for 12 pitch classes in using the three different methods: `librosa.feature.chroma_stft`, `librosa.feature.chroma_cqt`, and `librosa.feature.chroma_cens`. The `chroma_stft` uses the power spectrogram to compute a chromagram, and `chroma_cqt` uses the constant-Q transform instead. `chroma_cens` computes the chroma variant "Chroma Energy Normalized"that performs L-1 normalization and quantization of amplitude on top of the chroma vectors obtained from `chroma_cqt`.

1.3 Feature Summary

In `feature_summary.py` file, feature summarization is done to reduce the feature space for training. For all feature classes, temporal pooling is performed on mean, standard deviation, and variance over the window. The feature summarization results in a total of **378 feature dimensions**. The value is obtained by multiplying the three statistics by the sum of 5 (1 each from ZCR, spectral centroid, flatness, bandwidth, and rolloff); 7 from spectral contrast; 39 from MFCC and its delta and double delta; 39 from MFCC with CQT and its delta and double delta; and 36 (12 each from chromagram with STFT, CQT, and CENS).

1.4 Training Classifier

In `train_test.py` file, the training and testing take place using the extracted and summarized features in the above sections.

Cross Validation

The algorithm performs stratified k-fold cross validation on the train and validation data to build a robust model to new data. In the `run_test` method of the file, the program first loads the training, validation, and test data by calling the `combine_features` function of `feature_summary.py`. The labels for data are also generated. For stratified k-fold cross validation, it first prepares a combined set of the training and validation data. Then it applies the model selection method `librosa.model_selection.StratifiedKFold` (with the number of splits or folds equal to 6) for the combined data. Iterating through each set of split data, the program performs feature normalization on the split training and validation data; trains the model (by calling the `train_model` method in the same file); and saves the validation accuracy in an array.

Train and Test

In the `train_model` method, it trains the given data using the C-Support Vector Classification (`sklearn.svm.SVC` from scikit-learn library) with 'rbf' kernel. The validation accuracy is computed and returned. The model and validation accuracy for every iteration is stored each in an array. For the final testing, it chooses the model with the highest validation accuracy and runs the model with the normalized test set. The final test accuracy is computed, printed, and saved in a file.

2 Experiments and Results

2.1 Classifiers

In the course of building the model testing out different features, several classifiers including the linear SGD classifiers, Gausian Mixture Model (GMM), and K-Nearest Neighbor (k-NN) were tried out. However, the C-Suppor vector classifier (SVC) outperformed all other classifiers by far for this music instrument classification task.

2.2 MFCC Parameters

To find out the best combination of MFCC parameters, 54 combinations of FFT window size (512, 1024, and 2048), hop length (128, 256, 512), mel-bin size (40, 48, 64), and DCT type (2 and 3) are tested.

2.3 Results

The tests on 54 combinations show the highest test accuracy of **96.0%** and an average test accuracy of 94.5%. The best combination is summarized in Table 2.3, and the total results are presented in the Appendix.

FFT size	Hop length	Mel-bin size	DCT type	Test Accuracy
2048	512	64	2	96.0

Table 2: Best combination of parameters and accuracy

3 Discussion

3.1 Feature Selection

Through the results, it is shown that the classification model using the 15 features with 378 dimensions achieves a high accuracy of 96.0%. Because the goal of the classification is classifying musical instruments, it is important for the features to capture the key differences between instrumental sounds. Thus, any rhythmic features were excluded from consideration, and only timbre and pitch/harmony features are utilized. The experimental feature, MFCC with Constant-Q Transform, also plays a key role in improving the classification performance.

3.2 Tradeoff Between Accuracy and Time

Adding more features often (not always) results in an increase in the classification accuracy. However, a significant increase in the execution time of feature extraction, summarization, and training is unavoidable as well. The main focus of this work is finding the set of features and model that results in the highest accuracy. For this model to be applied in the real-world applications, however, the execution time must be tested and considered to make a good selection of the model because not only the accuracy but also energy consumption and training/classification time are important factors influencing other aspects of the system such as cost and user experience for example.

References

[Jiang, Lu, Zhang, Tao, and CaiJiang et al.2002] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. 2002. Music type classification by spectral contrast feature. In *Proceedings. IEEE International Conference on Multimedia and Expo*, Vol. 1. IEEE, 113–116.