Homework #1: Musical Instrument Recognition

The Robotics Program
20195595
Dooyoung Hong

1. Introduction

Mel-Frequency Cepstral Coefficients (MFCC) is the deterministic feature extractor based on speech domain knowledge and these formulas. MFCC contains a feature related to speech recognition that discards unnecessary information and leaves only important characteristics. The based model of Homework #1 performs MFCC extraction, train the model using a linear SVM model for classification, and validation it.

2. Algorithm description, Experiments, and results

The presented base code is easy and simple, but it has low validation performance. Therefore, I choose the following ideas for improving the base classification algorithm:

- Try different MFCC parameter settings: mel-bin size and DCT size
- Add delta and double-delta of MFCCs
- Try different classifiers: k-NN, SVM with non-linear kernels, and MLP

First, I use the short-time Fourier transform (STFT) function of Libros instead of the mfcc function. Fourier Transform is a function that gets a signal in the time domain as input and outputs its decomposition into frequencies. The STFT represents a single in the time-frequency domain by computing discrete Fourier transforms (DFT) over short overlapping windows. Librosa recommends using 2048 samples, which corresponds to a physical duration of 93 milliseconds at a sample rate of 22050 Hz because it is well adapted for music signals.

```
# STFT
S = librosa.core.stft(y, n_fft=2048, hop_length=512, win_length=512)

# power spectrum
D = np.abs(S)**2

# mel spectrogram (512 --> 40)
mel_basis = librosa.filters.mel(sr, 2048, n_mels=128)
mel_S = np.dot(mel_basis, D)

#log compression
log_mel_S = librosa.power_to_db(mel_S)

# mfcc (DCT)
mfcc = librosa.feature.mfcc(S=log_mel_S, n_mfcc=13)
#mfcc = librosa.feature.delta(mfcc)
mfcc = mfcc.astype(np.float32)    # to save the memory (64 to 32 bits)
```

Figure 1. STFT, Mel-filter, and DCT modifications.

Mel-filter expands the low-frequency part. I adjust the Mel-scale function that partitions the HZ into bis and transforms each bin into a corresponding in the Mel-scale using overlapping triangular filters. Discrete Cosine Transform (DCT) is a transformation that expresses a specific function as the sum of the cosine functions. However, Mel-filter already focused on the low-frequency part; energy concentration occurs at the lower frequencies after performing DCT. MFCC selects 12 coefficients with low frequency and dense information, and they use the energy of each frame about 12 coefficients as the 13th feature. So, I see the following approvement when I adapt first modification.

```
validation accuracy = 62.0 %
validation accuracy = 64.66666666666666 %
validation accuracy = 67.0 %
validation accuracy = 62.0 %
validation accuracy = 58.666666666666664 %
validation accuracy = 10.0 %
final validation accuracy = 67.0 %
```

Figure 2. Performance improvement adapted of the first modification.

Second, I add the delta and double-delta of MFCCs. The delta means the difference between frames of MFCCs, and double-delta means the difference between previous deltas. The main idea of deltas is simple. They derive a time-variant characteristic of the input data through the difference between each frame because STFT and DCT make discretized signal samples. So, I use a total of 39 input features as appending the delta and double-delta to MFCCs. It results in a huge performance improvement even in the base linear SVM model.

```
# mfcc (DCT)
mfcc = librosa.feature.mfcc(S=log_mel_S, n_mfcc=13)
mfcc_d = librosa.feature.delta(mfcc)
mfcc_d2 = librosa.feature.delta(mfcc, order=2)
mfcc = np.concatenate((mfcc, mfcc_d, mfcc_d2), axis=0)
mfcc = mfcc.astype(np.float32)    # to save the memory (64 to 32 bits)
```

Figure 3. Delta and double-delta addition.

```
validation accuracy = 90.33333333333333 %
validation accuracy = 88.66666666666667 %
validation accuracy = 90.33333333333333 %
validation accuracy = 82.0 %
validation accuracy = 73.33333333333333 %
validation accuracy = 10.0 %
final validation accuracy = 90.33333333333333 %
```

Figure 4. Performance improvement adapted of the second modification.

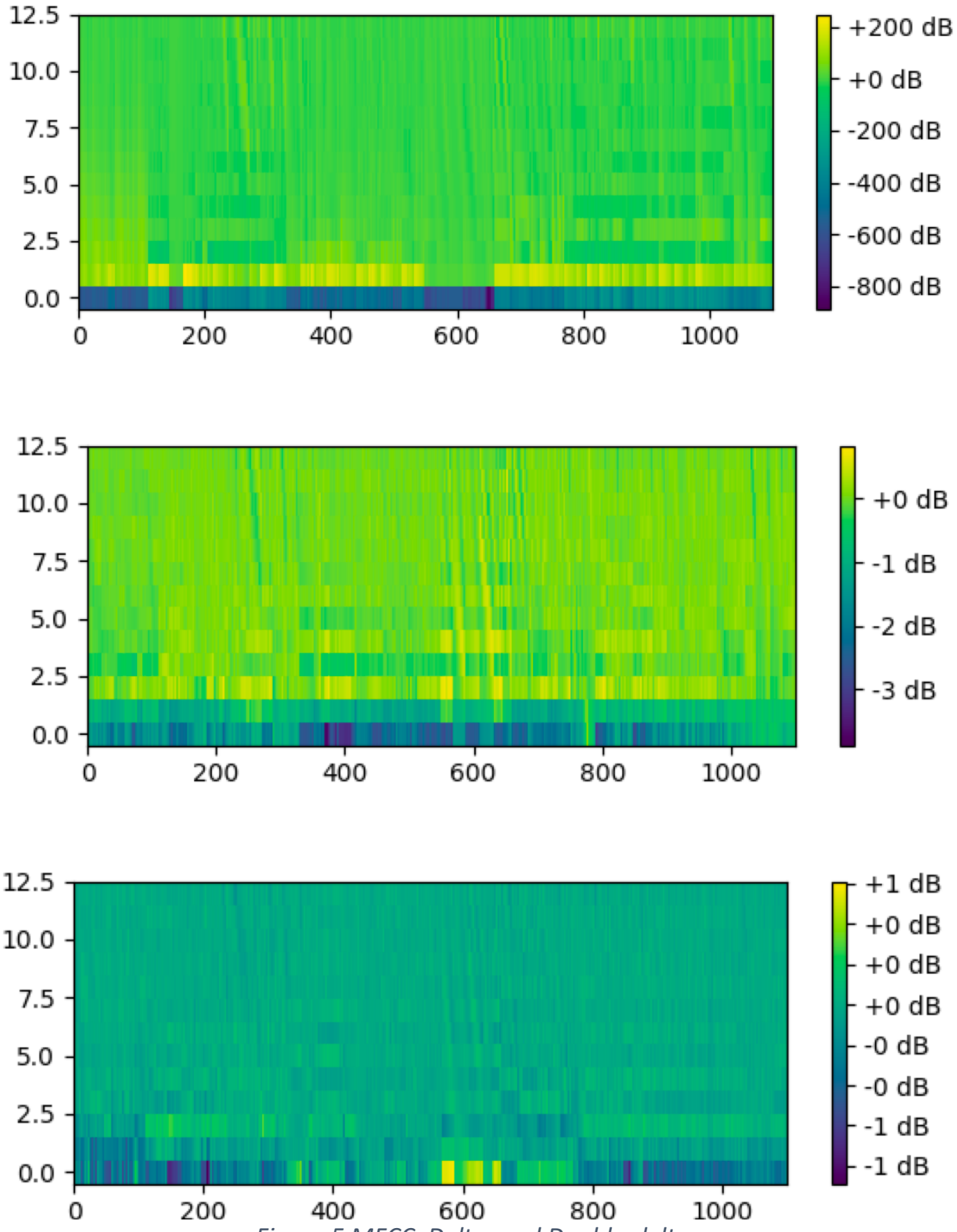

Figure 5 MFCC, Delta, and Double-delta.

Finally, I try to use other classifier models (K-NN, SVM with non-linear, and MLP).

```
# Choose a classifier (here, linear SVM)
# clf = SGDClassifier(verbose=0, loss="hinge", alpha=hyper_param1, max_iter=1000, penalty="l2", random_state=0)

# K-NN
# clf = KNeighborsClassifier(algorithm="auto", n_neighbors=5)

# SVM with non linear
# clf = svm.NuSVC(gamma='auto')

# MLP for classifier
clf = MLPClassifier(hidden_layer_sizes=(50,), activation='logistic',\
                    solver='adam', alpha=0.01, batch_size=128,\
                    learning_rate_init=0.01, max_iter=5000)
```

Figure 6. Implementation of classifier models.

K-NN model searches the nearest neighbor based on distance measures. It predicts a label among clustered neighbors after finding K neighbors that are close to each other. The performance of this model depends on the number of neighbors. It shows a weak performance improvement compared with the base model when choosing a random sample (k=5).

```
validation accuracy = 91.33333333333333 %
validation accuracy = 91.33333333333333 %
validation accuracy = 91.33333333333333 %
validation accuracy = 91.33333333333333 %
validation accuracy = 91.33333333333333 %
validation accuracy = 91.33333333333333 %
final validation accuracy = 91.33333333333333 %
```

Figure 7. Validation results of the K-NN model.

Support Vector Machine (SVM) performs the classification using the support vector and hyperplane. The SVM model that has non-linearity shows higher performance than the base model (linear SVM) by maximizing the margin through the combination of time-variant input features, non-linear hyperplane and support vectors.

```
validation accuracy = 93.0 %
validation accuracy = 93.0 %
validation accuracy = 93.0 %
validation accuracy = 93.0 %
validation accuracy = 93.0 %
validation accuracy = 93.0 %
final validation accuracy = 93.0 %
```

Figure 8. Validation results of the SVM (with non-linear) model.

Multi-Layer Perceptron (MLP) is a model in which several layers of neurons are sequentially attached. MLP is the most accessible feed-forward and fully connected neural network in machine learning. A machine learning model based on data extracts features of the data and learns them to increase the accuracy of the model. This model also exhibits non-linearity through the activation functions which makes the model robust even for noisy data. The accuracy of the MLP model is about 97%, which is the highest among the classifier models used.

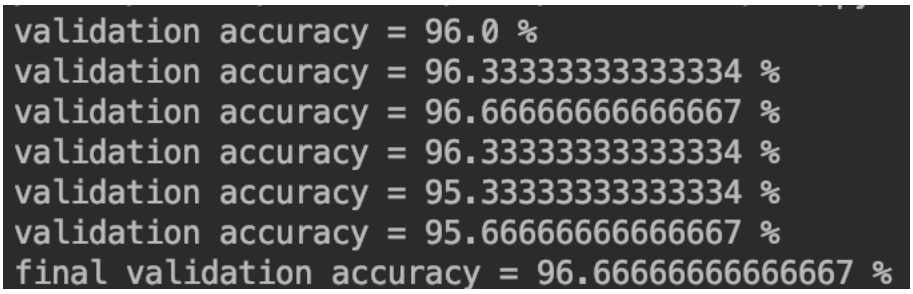

Figure 9. Validaton results of the MLP model.

3. Conclusion

HW1 performs pre-processing for music data. It forms a spectrogram through Fourier transform from wav files and generates MFCC using Mel filter and DCT. In addition, it conducts the overall functions of libros to evaluate the features of MFCCs extracted by using a special input for speech recognition with time-variant properties. As a result, the MLP model achieves a validation accuracy of 97 %.