GCT634-HW1 / 20244418 / 임주은

1. **Algorithm Description**
   1. Data Preprocessing
      1. Loading Audio Files

The audio file paths are read from the provided metadata text file, loaded, appended to a list, and stacked into a numpy array with the shape (num\_dataset, audio\_length).

* + 1. Extracting Audio Features

Using the specified STFT parameters and a list of audio feature types, the audio features are extracted from the input audio files. The output is a list containing each feature, preventing the need for repeated STFT computation.

* + 1. Processing Extracted Features
       1. Concatenation

The extracted audio feature list is concatenated into a numpy array with the shape (num\_dataset, sum\_dimension, num\_frames).

* + - 1. Compression

Depending on the chosen compression method, compression rate, and the axis along which to compress, the output and pooling size are determined. For PCA and k-means, models are additionally returned after fitting the training dataset, which can then be provided as a parameter to the validation dataset to apply the same model state.

* + - 1. Flatten

All feature information is flattened into a single feature vector.

* + - 1. Normalization

Z-normalization is applied to standardize the features.

* 1. Models
     1. Initialization

The model class is initialized with the specified model type. Sklearn models are utilized, and arbitrary parameters can be passed during model creation.

* + 1. Train

Calls fit function from the model and deliver train dataset

* + 1. Validation

Returns accuracy given validation dataset.

1. **Data Analysis**
   1. Visualization of Each Class

Audio waveforms and log spectrograms are visualized for each class. All samples are distinguishable by their shapes. A key consideration is that while there are distinct changes over time, the original code only utilizes a single mean value across the entire time axis. Additionally, the presence of thin wiggling line patterns in the spectrogram suggests that the frequency resolution needs to be sufficiently high. More plots are available in the notebook.

A close-up of a graph

AI-generated content may be incorrect.A close-up of a graph

AI-generated content may be incorrect.A close-up of a graph

AI-generated content may be incorrect.

* 1. Visualization of Spectrograms

The log spectrogram appears to contain the most visual information, suggesting that it may also preserve the most audio information. The log scale expands the details, making them more visible. Meanwhile, MFCC seems more effective for compression but have significant information loss.

A collage of different colored lines

AI-generated content may be incorrect.A collage of different colored lines

AI-generated content may be incorrect.

* 1. Hypothesis

Therefore, the hypothesis is:

* + 1. Higher resolution in both time and frequency will improve performance.
    2. Features that are more visually distinguishable in images will also have better performance, with the log spectrogram being the most effective, the log scale outperforming the linear scale, and MFCC showing weaker performance.
    3. Compression methods that preserve time information will yield better performance.

1. **Experiments**
   1. Audio Features
      1. Single Audio Feature Comparison

This experiment examined the impact of STFT parameters on classification performance. The FFT window size was set to 512, 1024, or 2048, with hop lengths of one-fourth, one-half, or the full window. Mel bins were tested at 32, 64, and 128, while MFCC DCT sizes were 10, 20, or 40. Analyzed features included magnitude, power, spectrogram variants, MFCCs, and spectral attributes. The FFT window length matches the windowing function size for consistency. Focusing on timbre classification, pitch, tonal, and rhythmic features were excluded. The baseline SVM model used hinge loss, L2 penalty, an alpha of 0.001, 1000 iterations, and a fixed random seed.

* + 1. Audio Feature Pair Comparison

In addition to single feature evaluations, 66 combinations of two different feature types were tested.

* 1. Data Compressions

This experiment applied various compression methods: mean pooling, max pooling, and PCA, on classification performance. K-means was excluded due to memory management issues with high-dimensional data. Compression was performed along three axes: all data (axis 0), the feature dimension (axis 1), and the time dimension (axis 2). Three compression ratios were tested: 0.5 (50% reduction), 0.75 (75% reduction), and 1.0 (extreme compression to a single value). To ensure a comprehensive analysis, all extracted audio features were concatenated into a single feature matrix before compression.

* 1. Models

K-NN, logistic regression, SVM, MLP, GMM, and Random Forest, were tested with default settings. Each was trained and validated on the same extracted and compressed features to assess performance.

1. **Results and discussions**
   1. Audio Features
      1. Single Audio Feature Comparison

Higher-resolution features generally improved classification accuracy, as seen with increasing values of n\_fft​, n\_mels, and n\_mfcc​. However, the impact of hop length varied by feature type, with some benefiting from longer hop length. Spectral features, when averaged along the time axis, led to performance drops, indicating excessive information loss. The log spectrogram consistently achieved the highest accuracy, confirming its effectiveness. Log-scale features outperformed linear-scale ones, reinforcing their advantage in representation. Despite initial expectations, MFCC performed well, proving to be a compact yet effective feature.

A graph of different colored lines

AI-generated content may be incorrect.

* + 1. Audio Feature Pair Comparison

The best combination was the log spectrogram and double-delta MFCC, with the latter complementing the former by capturing rapid spectral changes. Spectrogram-based features and MFCC variations contained redundant information, making their combinations less effective. Spectral features like centroid and bandwidth contributed minimally, suggesting limited usefulness for this task.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. Data Compressions

Mean pooling generally shows similar performance, except when the compression reaches 100%, suggesting it behaves like downsampling. Max pooling, on the other hand, enhances performance when compressing the time dimension, even at full compression, indicating that the maximum frequency values play a crucial role. PCA works well when reducing the feature dimension, suggesting it effectively captures important features. However, it struggles when compressing along axis 0, likely because the output dimension cannot exceed the number of samples. All methods performed poorly when reduced to a single value. Finally, contrary to the third hypothesis, the performance drop from reducing the time dimension depends more on the method and its interaction with the axis, rather than the axis itself.

A graph of a graph

AI-generated content may be incorrect.

* 1. Models

Logistic regression achieved the highest accuracy at 96.67%, making it the most effective model in this experiment. In contrast, k-NN performed significantly worse than the other methods, and parameter tuning did not lead to meaningful improvements, suggesting its limitations in capturing complex feature relationships. However, one unexplored possibility is that k-NN could perform better when paired with k-means clustering compression. Since k-means represents data as distances to the mean vectors of k-groups, this structure might better align with k-NN’s distance-based classification approach.

A graph of a graph with numbers

AI-generated content may be incorrect.

1. **Further Possible Experiments**
   1. Audio Features
      1. Experimenting with different feature combinations for various compression methods.
      2. Evaluating the impact of different feature sets on different models.
   2. Compressions
      1. Investigating the effect of combining multiple compression methods.
      2. Testing higher compression rates to enable the use of k-means clustering.
   3. Models
      1. Fine-tuning model parameters for better performance.
      2. Running GMM with optimized memory management
      3. Examining different combinations of compression methods and models.