Audio Content Analysis
Assignment # 3
Instructor: Juan Pablo Bello

A Implementation [7 pts]

In this section you will be implementing a series of functions, and finally a script tying them all together. Each function should be saved as a separate .m file, where the filename is identical to the function name. The functions definitions should ***exactly*** *follow the given code. You may use any previously submitted functions. Submit all code to NYU classes as a single zip file named "YourLastName3.zip".*

Details for each function can be found at the end of the assignment.

1. Write a function that converts a vector of values in Hz to Mels.
   ```
   melval = hz2mel(hzval)
   ```

2. Write a function that converts a vector of values in Mels to Hz.
   ```
   hzval = mel2hz(melval)
   ```

3. Write a function that computes a sequence of MFCCs from a mono audio file. This function should:

   - Compute the spectrogram of the signal using a hamming window and the given parameters.
   - Create a mel filterbank that begins on `min_freq` and ends on `max_freq` with `n_mel_filts` using (asymmetric) triangular windows. Your filterbank should be a matrix of size `n_mel_filts` x the number of frequency bins in the spectrogram. You may find the provided function `find_nearest` to be useful.
   - Compute the mel *power* spectrum in dB.
   - Compute the DCT of each column of the mel spectrum. You may use MATLAB's `dct` function for this. Warning - do not use the second parameter!
   - Remove the first (DC) coefficient of each resulting feature vector and normalize the remaining coefficients to sum to one.

   ```
   [mfccs, fs_mfcc] = compute_mfccs(filepath, win_size, hop_size, ...
                                    min_freq, max_freq, num_mel_filts, n_dct)
   ```

4. Write a function that takes a list of audio files and uses the `compute_mfccs` function to create a combined MFCC sequence along with the class labels. The labels corresponding to `fpath1` and `fpath2` should be the integers 1 and 2 respectively. For this assignment, you will call the same function to create training and testing sets, such that each file path will contain all of the training or testing data for a single instrument class. (NOTE: the `params` argument should have the datatype *struct*; review the Matlab documentation if necessary)

   ```
   [train_features, train_labels] = ...
       create_set(fpath1, fpath2, params)
   ```

   or

```
[test_features, test_labels] = ...
    create_set(fpath1, fpath2, params)
```

5. Write a function that predicts the class values for the test set given a training set using a nearest neighbor classifier. A nearest neighbor classifier computes the dot product between the feature vector corresponding to each sample in the testing set, and all the feature vectors in the training set. Then it selects the training sample with maximum dot product, and assigns its label to the test sample. If this label (e.g. flute) matches the label of the test sample then the classification is correct. This process is repeated for all samples in the test set [2 pts].
   ```
   predicted_labels = predict_labels(train_features, train_labels, test_features)
   ```

6. Write a function that computes the overall and per-class accuracy for the predicted labels [0.5 pts].
   ```
   [overall_accuracy, per_class_accuracy] = ...
       score_prediction(test_labels, predicted_labels)
   ```

7. Write a Matlab script called `assignment3.m` which runs the code used to write your report (see the Analysis section below) [0.5 pts].

B Analysis [3 pts]

Write a report addressing each of the questions below. Please submit your report as a pdf file to NYU Classes.

Download the audio files from NYU Classes. For the following, use the following parameter settings: `win_size=1024`, `hop_size=512`, `min_freq=86`, `max_freq=8000`, `num_mel_filts=40`, and `n_dct=15`.

1. Compute a training feature set using the files `piano_train.wav` and `trumpet_train.wav` [0.5 pts].

2. Compute a test feature set using the files `piano_test.wav` and `trumpet_test.wav` [0.5 pts].

3. Using these sets, compute (i) the percentage of frames that are correctly classified per source class, and (ii) the overall accuracy [0.5 pts].

4. Comment on the results. Which types of errors were made? [0.5 pts]

5. Repeat the previous steps, replacing the `piano` files with `trombone` files for training and testing. How do results change and why? [1 pts]

Function Details

The functions you write for section A should begin with the headers below, and the inputs/outputs should follow the specifications in the function's doc-strings.

```
function melval = hz2mel(hzval)
%   Convert a vector of values in Hz to Mels.
%
%   Parameters
%   ----------
%   hzval : 1 x N array
%       values in Hz
%
%   Returns
%   -------
%   melval : 1 x N array
%       values in Mels

function hzval = mel2hz(melval)
%   Convert a vector of values in Hz to Mels.
%
%   Parameters
%   ----------
%   melval : 1 x N array
%       values in Mels
%
%   Returns
%   -------
%   hzval : 1 x N array
%       values in Hz

function [mfccs, fs_mfcc] = compute_mfccs(filepath, win_size, hop_size, ...
    min_freq, max_freq, num_mel_filts, n_dct)
%    Compute MFCCs from audio file.
%
%   Parameters
%   ----------
%   filepath : string
%       path to .wav file
%   win_size : int
%       spectrogram window size (samples)
%   hop_size : int
%       spectrogram hop size (samples)
%   min_freq : float
%       minimum frequency in Mel filterbank (Hz)
%   max_freq : float
%       maximum frequency in Mel filterbank (Hz)
%   num_mel_filts: int
%       number of Mel filters
%   n_dct: int
%       number of DCT coefficients
%
%   Returns
%    -------
%   mfccs : n_dct-1 x NT array
%       MFCC matrix (NT is number spectrogram frames)
%   fs_mfcc : int
%       sample rate of MFCC matrix (samples/sec)
```

```
function [features, labels] = ...
    create_set(fpath1, fpath2, params)
%   Compute features and parameters for training data.
%
%   Parameters
%   ----------
%   fpath1: string
%       full path to audio file with training data from class 1
%   fpath2: string
%       full path to audio file with training data from class 2
%   params: struct
%       Matlab structure with fields are win_size, hop_size,
%       min_freq, max_freq, num_mel_filts, n_dct, the parameters
%       needed for computation of MFCCs
%
%   Returns
%   -------
%   features: NF x NE matrix
%       matrix of training/testing set features (NF is number of
%       features and NE is number of feature instances)
%   labels: 1 x NE array
%       vector of training/testing labels (class numbers) for each instance
%       of features


function predicted_labels = ...
    predict_labels(train_features, train_labels, test_features)
%   Predict the labels of the test features,
%   given training features and labels,
%   using a nearest-neighbor classifier.
%
%   Parameters
%   ----------
%   train_features: NF x NE_train matrix
%       matrix of training set features (NF is number of
%       features and NE_train is number of training feature instances)
%   train_labels: 1 x NE_train array
%       vector of labels (class numbers) for each instance
%       of train_features
%   test_features: NF x NE_test matrix
%       matrix of test set features (NF is number of
%       features and NE_test is number of testing feature instances)
%
%   Returns
%   -------
%   predicted_labels: 1 x NE_test array
%       array of predicted labels

function [overall_accuracy, per_class_accuracy] = ...
    score_prediction(test_labels, predicted_labels)
%   Compute the confusion matrix given the test labels and predicted labels.
%
%   Parameters
%   ----------
%   test_labels: 1 x NE array
%       array of ground truth labels for test data
%   predicted_labels: 1 x NE_test array
%       array of predicted labels
%
```

```
%   Returns
%   -------
%   overall_accuracy: scalar
%       The fraction of correctly classified examples.
%   per_class_accuracy: 1 x 2 array
%       The fraction of correctly classified examples
%       for each instrument class.
%       per_class_accuracy[1] should give the value for
%       instrument class 1, per_class_accuracy[2] for
%       instrument class 2, etc.
```