UCLA

Dept. of Electrical and Computer Engineering

EE214B

Problem Set 3

Due: 5/6/2018

Design a simple Digit Recognition System using the template-based approach. In order to get the similarity score between your data and template digits, you can directly use MATLAB's built-in function dtw() which performs Dynamic Time Warping over two matrices and gives distance scores. The score can be used for classification.

We have 50 examples of the digits 0-9 as spoken by an adult male. We chose randomly one example of each digit as the template so that there are 10 template utterances (one for each digit) and 490 test utterances. The label (or digit) of each utterance is the first character of the filename.

First, we generated the mfcc matrix for each digit. This is done by windowing the raw audio files using a Hamming window of 25 msec with 10 msec frame shift. The code for generating the MFCC matrix is included below. For testing, you should generate the mfcc matrix (considered as a sequence of MFCC vectors) for a given utterance, then calculate the distance to each template digit by using DTW for test matrix and template digit matrices. The template digit with the lowest distance (i.e. highest similarity) to your test mfcc matrix will be your prediction.

You should use the utterances in the templates folder as template digits. The data folder contains all test utterances. In your report, you should write *the error rate for* your implementation code. You can add a confusion matrix to understand misclassifications better.

You can download the dataset from CCLE.

Example pseudo code:

```
For each test utterance

        For digit 0 to 9

                distances(digit) = dtw(template_mfcc_matrix, test_mfcc_matrix)

        Find index with minimum distance

        if index not equal to test label

                error = error + 1

        end

error = error / #test utterances
```

Tasks:

1. Run the system with MFCCs and indicate the error rate (you should get more than 90 percent accuracy). Also examine which digits had the worst performance.

2. Change the feature matrix, to compare performance with the MFCC feature set, to one of your choice (PLP, LPCC, formants, etc.). Compare the performance.

3. (BONUS, Optional) Change the distance from Euclidian to another one (for example, Manhattan) and repeat part 1.

Example MFCC code:

```
function mfcco = MFCC(rawdata,Fs)

winlen      = 25;           % window length in 100 nsec
winshft     = 10;           % window shift in 100 nsec %samples
cepnum      = 12;            % number of cepstral coefficients
liftercoe   = 22;            % liftering coefficient
numchan     = 26;            % number of channels of the MEL filter bank 26
preemcoeff  = 0.97;          % coefficient for pre-emphasis
deltawindow = 2;             % window length to calculate 1st derivatives
accwindow   = 2;             % window length to calculate 2nd derivatives
C0          = 0;             % use zeroth cepstral coefficient (0/1)
usedeltas   = 1;
useenergy   = 0;

% -------------------------------------------------------------
% START OF PROGRAM
input = rawdata;
fsamp = Fs;
winlen = round(winlen * 10^(-3) * fsamp);
winshft = winshft * 10^(-3) * fsamp;
if isempty(whos('FrameNo' ))
    FrameNo = ceil((length(input) - winlen) / winshft);
end
nfft = 2*(2^nextpow2(winlen)); % FFT size

% initialize MEL filter bank
fbank = initfiltb(winlen, numchan, fsamp, nfft);

% initialize lifter coefficients
lifter = (1 + (liftercoe/2)*sin((pi/liftercoe)*(0:cepnum)) );

% pre-emphasis
am = [1 0];  % denominator polynomial
bm = [1 -preemcoeff];             % numerator   polynomial
preem = filter(bm, am, input);
```

```
% change signal (a vector) into frame (a matrix), where each collum is a
frame
frmwin = sig2fm(preem, winlen, winshft, FrameNo);
[winlen, framenum] = size(frmwin);

% Hamming window each frame
frmwin = frmwin .* (hamming(winlen) * ones(1, framenum));


% Log energy
LE = log(sum(frmwin.*frmwin));

% FFT
fft_mag = zeros(floor(nfft/2), framenum);
for i = 1:framenum
    temp = abs(fft(frmwin(:, i), nfft));
    fft_mag(:, i) = temp(1:floor(nfft/2));
end

% MEL filtering
fb = fbank*fft_mag;

% take logarithm of MEL filter output
fbfloor = mean(mean(fb)) * 0.00001;
logfb = log(max(fb, fbfloor*rand(size(fb))));

% take DCT
mfcco = dct(logfb);
if C0
    mfcco = mfcco(1 : cepnum + 1, :);
else
    mfcco = [LE;mfcco(2 : cepnum + 1, :)];
    %lifter = lifter(2 : end);
end

% do liftering with a lifted sin wave
mfcco = mfcco .* (lifter' * ones(1, framenum));


if(~useenergy)
    mfcco = mfcco(2:end,:);
end

% calculate 1st derivative (velocity)
dt1 = deltacc(mfcco, deltawindow);

% calculate 2nd derivative (acceleration)
dt2 = deltacc(dt1, accwindow);

if(usedeltas)
    % append dt1 and dt2 to mfcco
    mfcco = [mfcco; dt1];
end
```

```
end

% END OF PROGRAM
% ----------------------------------------------------------------

% ----------------------------------------------------------------
% START FUNCTION DEFINITIONS

function mels = mel(freq)
% change frequency from Hz to mel
mels = 1127 * log( 1 + (freq/700) );
end

% ----------------------------------------------------------------
function wins = sig2fm(input, winlen, winshft, frameno)
% put vector into matrix, each column is a frame.
% The rest of signal that is less than one frame is discarded
% winlen, winshft are in number of sample, notice winshft is not limited to
% integer
input = input(:);
wins=zeros(winlen, frameno);

for i = 1 : frameno
    b = round((i-1) * winshft);
    c = min(winlen, length(input) - b);
    wins(1:c,i) = input(b+1 : min(length(input), b+winlen));
end
end

% ----------------------------------------------------------------

function fbank = initfiltb(framelen,numchan,fsamp,nfft)
% triangle shape melfilter initialization

fftfreqs = ((0:(nfft/2-1))/nfft)*fsamp;  % frequency of each fft point (1-
fsamp/2)
melfft = mel(fftfreqs);   % mel of each fft point

mel0 = 0;
mel1 = mel(fsamp/2);       % highest mel
melmid = ((1:numchan)/(numchan+1))*(mel1-mel0) + mel0; % middle mel of each
filter

fbank = zeros(numchan,nfft/2);

% non overlaping triangle window is used to form the mel filter
for k = 2:(nfft/2)  % for each fft point, to all the filters,do this:
  chan = max([ 0 find(melfft(k)>melmid) ]); % the highest index of melfft
that is larger than the middle mel of all channels
  if(chan==0)  % only the first filter cover here
    fbank(1,k) = (melfft(k)-mel0)/(melmid(1)-mel0);
  elseif(chan==numchan)  % only the last filter covered here
    fbank(numchan,k) = (mel1-melfft(k))/(mel1-melmid(chan));
  else                   % for any other part, there will be two filter cover
that frequency, in the complementary manner
```

```matlab
    fbank(chan,k) = (melmid(chan+1)-melfft(k))/(melmid(chan+1)-melmid(chan));
    fbank(chan+1,k) = 1-fbank(chan,k);  % complementary
  end
end
end

% ---------------------------------------------------------------

function dt = deltacc(input, winlen)
% calculates derivatives of a matrix, whose columns are feature vectors

tmp = 0;
for cnt = 1 : winlen
    tmp = tmp + cnt*cnt;
end
nrm = 1 / (2*tmp);

dt   = zeros(size(input));
rows = size(input,1);
cols = size(input,2);
for col = 1 : cols
    for cnt = 1 : winlen
        inx1 = col - cnt; inx2 = col + cnt;
        if inx1 < 1;     inx1 = 1;     end
        if inx2 > cols;  inx2 = cols;  end
        dt(:, col) = dt(:, col) + (input(:, inx2) - input(:, inx1)) * cnt;
    end
end
dt = dt * nrm;
end

% END FUNCTION DEFINITIONS
% ---------------------------------------------------------------
```