EE519 Homework 4 Samwoo Seong

Problem1

Part 1

(b)

Assumption: Length of all audio signals is 16000(i.e. 1 sec with Fs = 16000Hz)

- ⇨ Total number of frames: 99 frames where length of each frame is 400 samples(=25 msec) and shift is equal to 10msec

(c)

(i)

-How many filters do you need?

8000Hz => 19.7089 Bark => To be approximately 1 Bark for the distance between one filter and another filter, I need 20 filters.

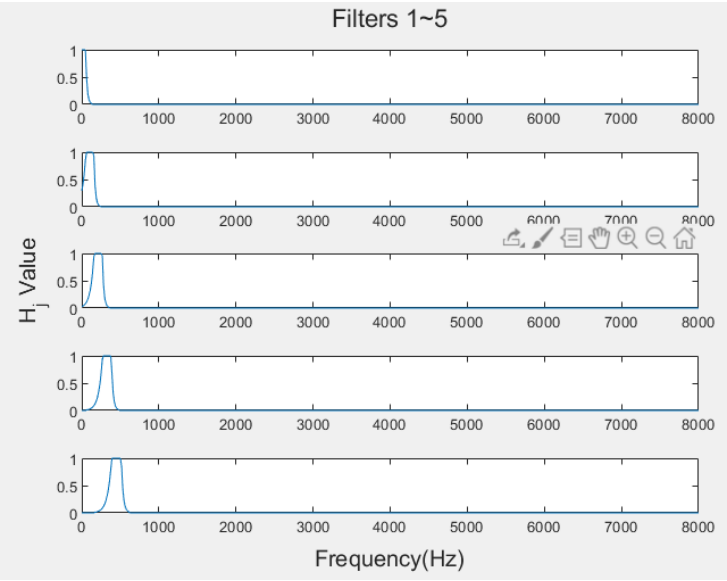
-Central frequency for 3rd filter

=>filter interval: 19.7089/20 = 1.0373

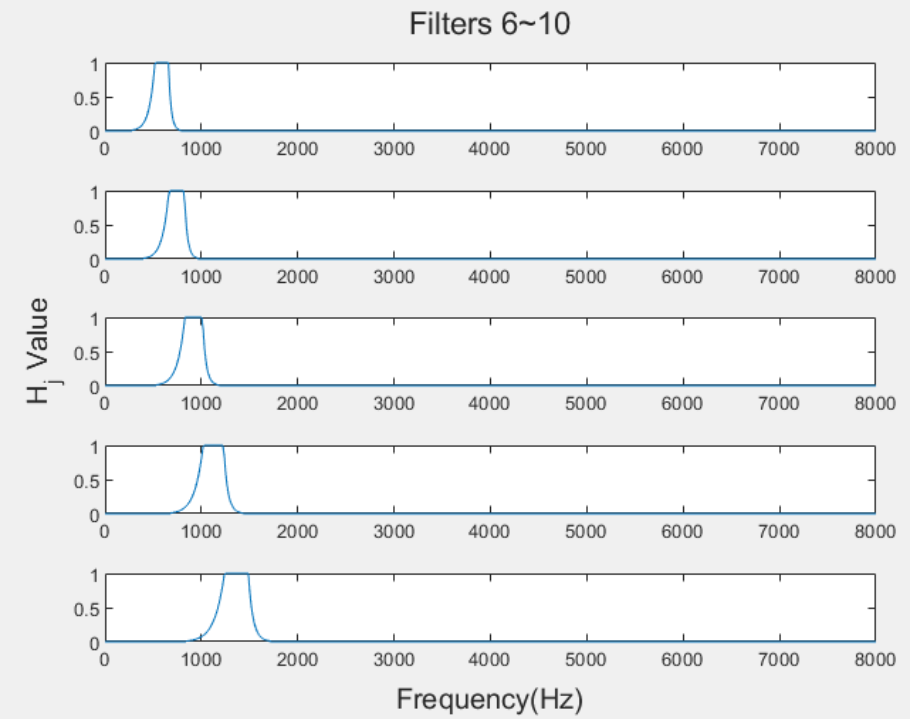
=> Central frequency for 3rd filter =1.0373*2 = 2.0746 Bark (= 600*sinh(2.0746/6) = 211.6186 Hz)

Similarly Central frequency for 5th filter = 1.0373*4 = 4.1492 Bark (= 600*sinh(4.1492/6) = 448.7902 Hz)

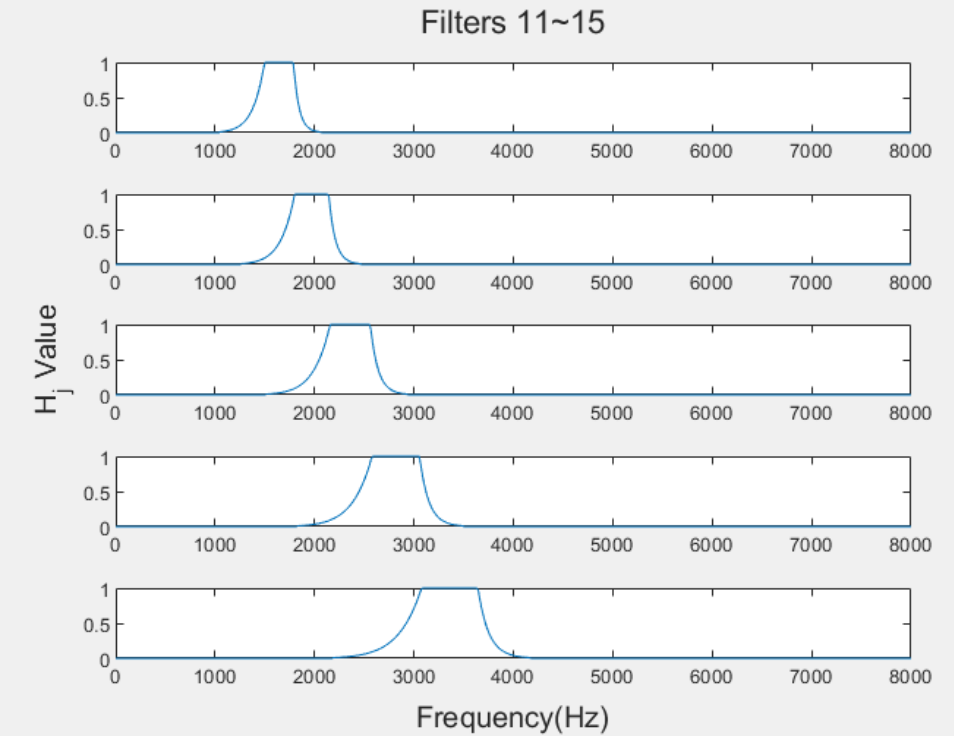
(ii)Filter bank plots



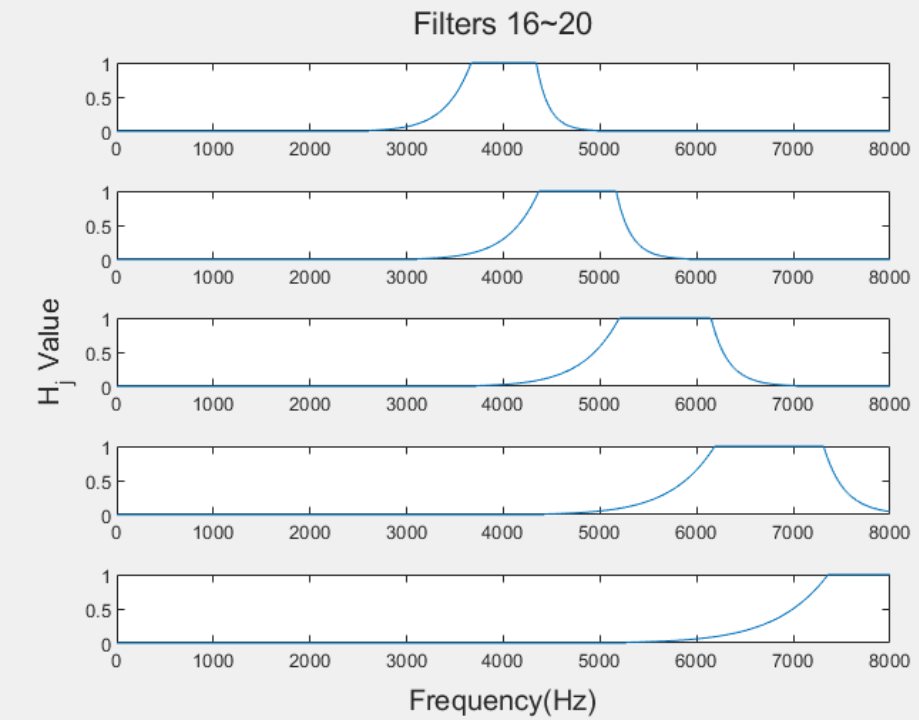
Top down order e.g. First row: first filter, Second row: second filter, … , Fifth row: fifth filter



Top down order e.g. First row: 6th filter, Second row: 7th filter, … , Fifth row: 10th filter

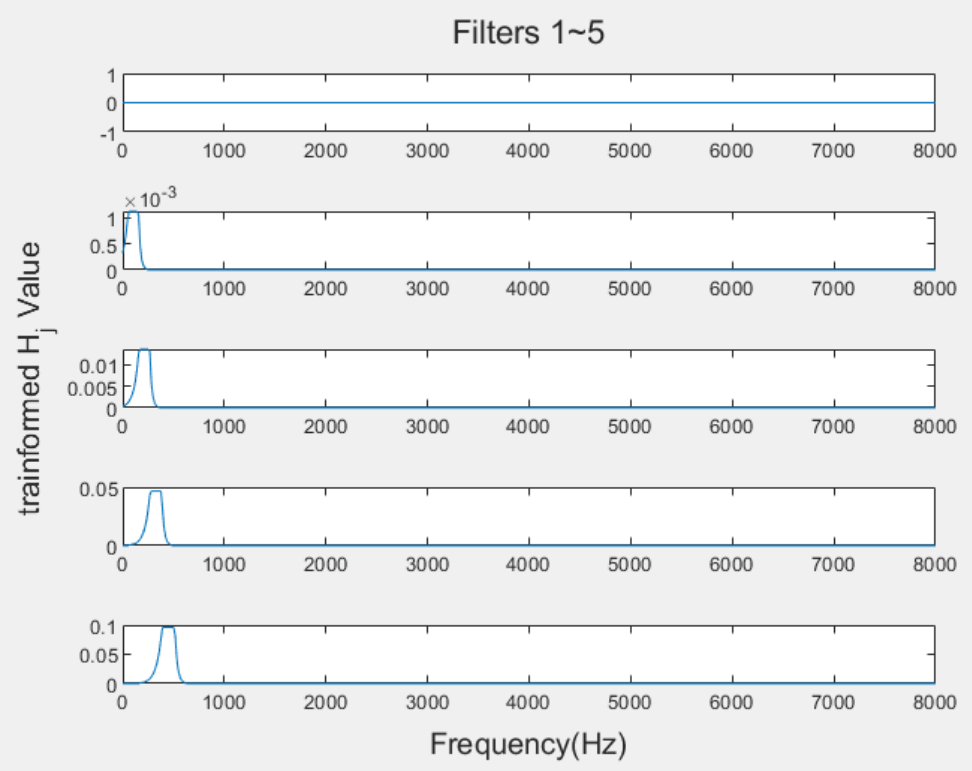


Top down order e.g. First row: 11th filter, Second row: 12th filter, … , Fifth row: 15th filter

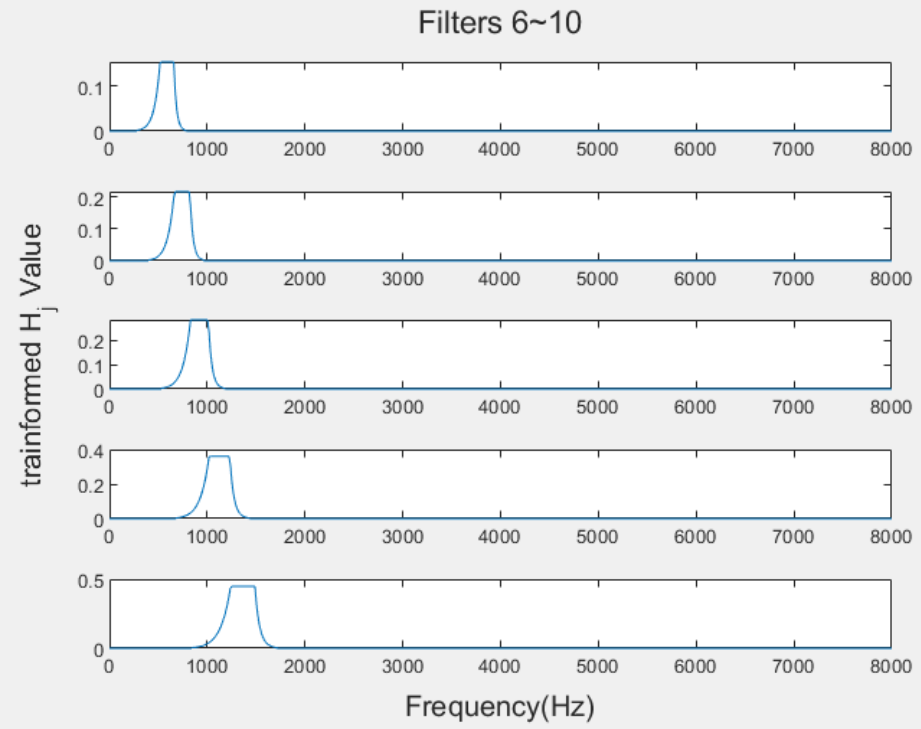


Top down order e.g. First row: 16th filter, Second row: 17th filter, … , Fifth row: 20th filter

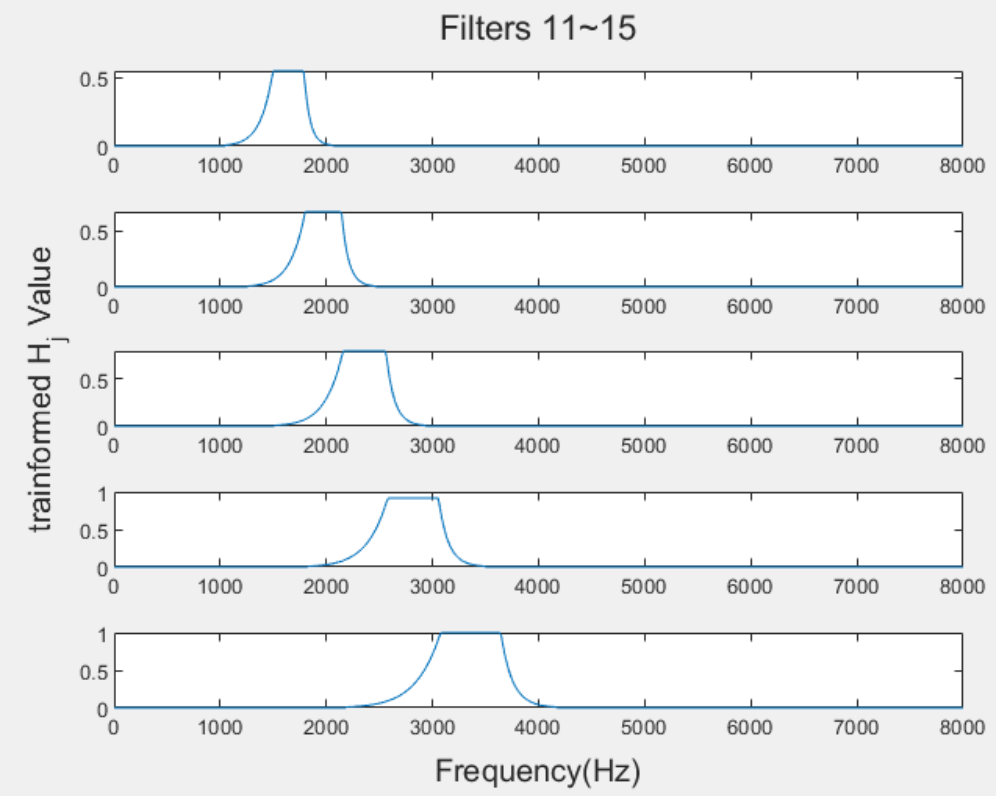
(iii) Transformed filters plots



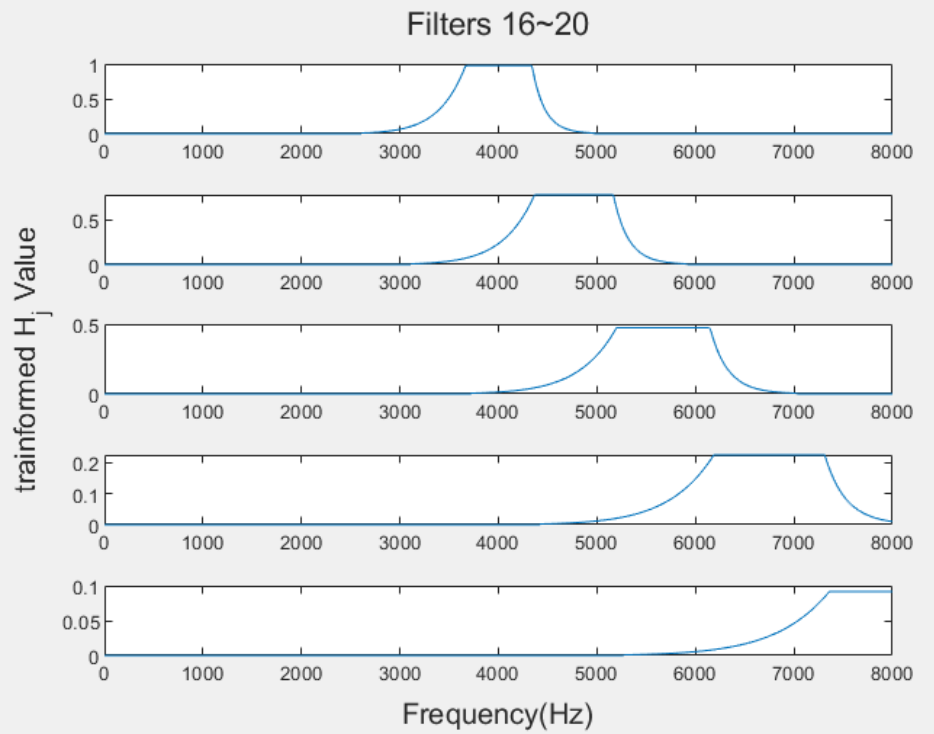
Top down order e.g. First row: first filter, Second row: second filter, … , Fifth row: fifth filter



Top down order e.g. First row: 6th filter, Second row: 7th filter, … , Fifth row: 10th filter

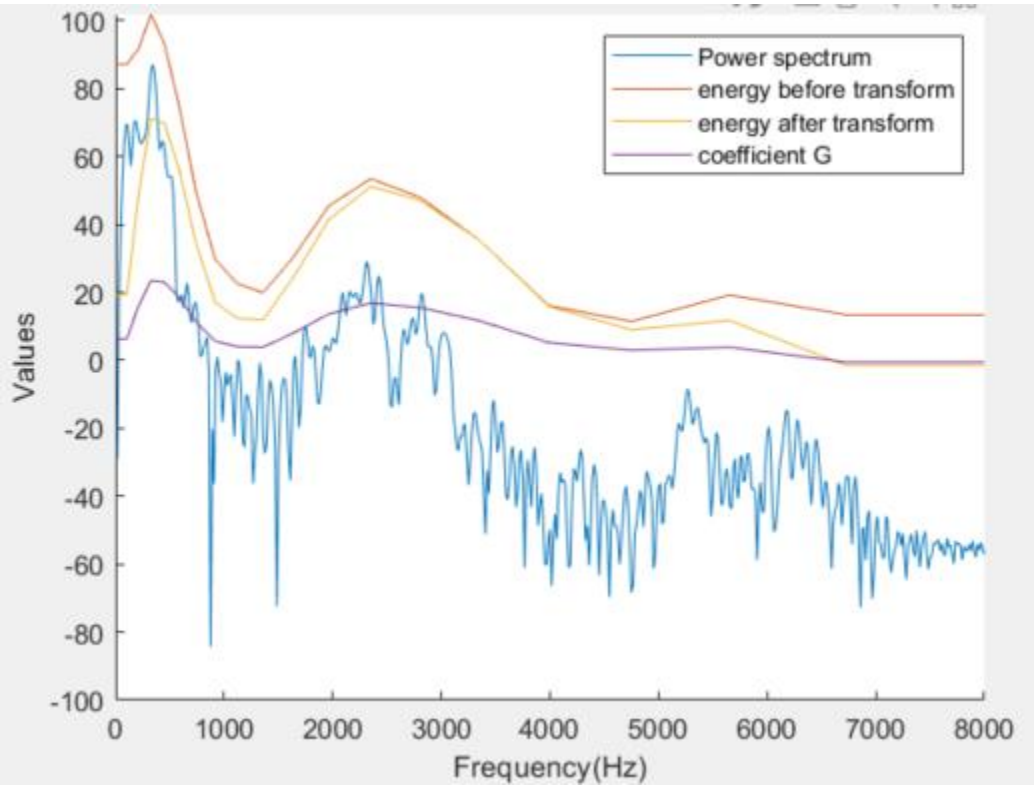


Top down order e.g. First row: 11th filter, Second row: 12th filter, … , Fifth row: 15th filter



Top down order e.g. First row: 16th filter, Second row: 17th filter, … , Fifth row: 20th filter

(d)



Generally speaking, there is more information (a.k.a energy), graph shows higher value at low frequency and it barely change as its frequency increases. Also, we can observe several envelops because speech signal is resonated by formants.

(e)

(i)LPC coeeficients

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	-0.2574	0.0863	-0.1608	0.3141	0.1908	0.0115	-0.0054	0.0547	0.0296	0.0520	-0.0216	0.0317

Left to Right (a0 to a12)

(ii)Cepstral Coefficients before lifting

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.5091	-0.2574	0.1194	-0.1887	0.3661	0.0818	0.0345	-0.0609	0.1069	0.0403	0.0738	-0.0608	0.0689

Left to Right (c0 to c12)

(iii)

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.5091	-0.2574	0.1809	-0.3648	0.8410	0.2149	0.1010	-0.1958	0.3723	0.1506	0.2937	-0.2563	0.3060

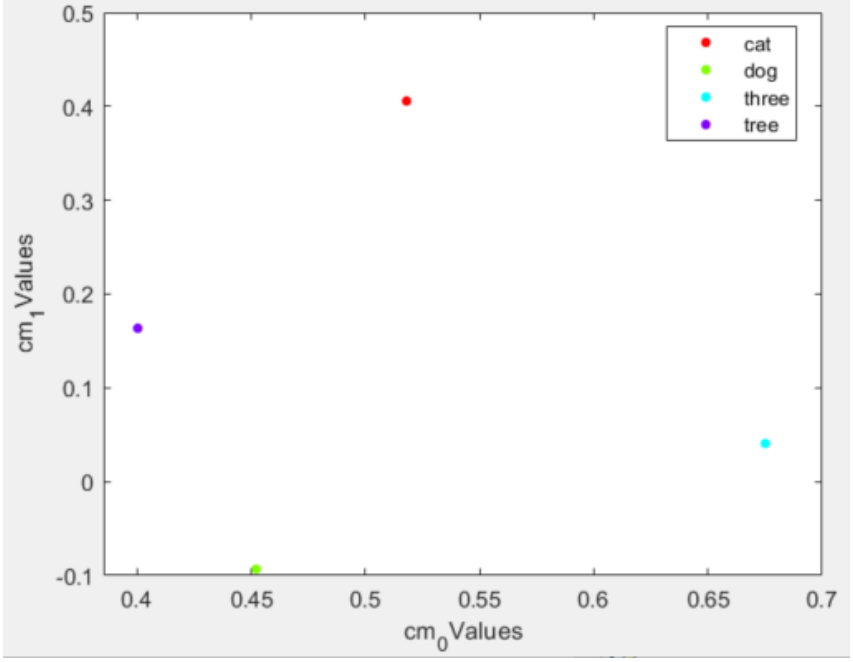
Left to Right (c0 to c12)

Part 2

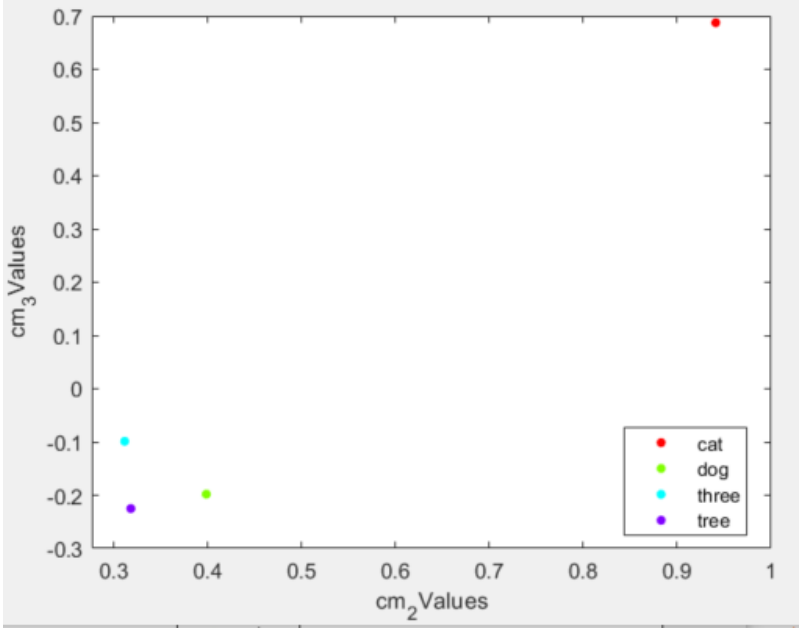
(a)

For just one record file for each word

(cm0, cm1) pair



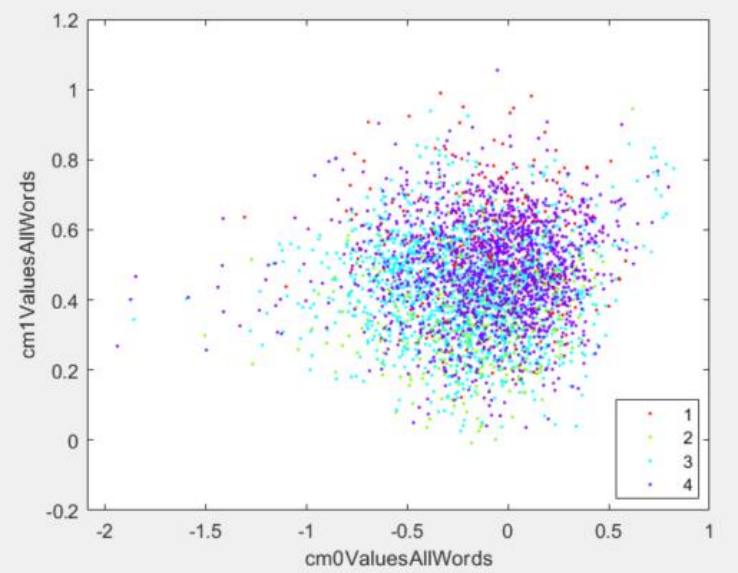
(cm2, cm3) pair



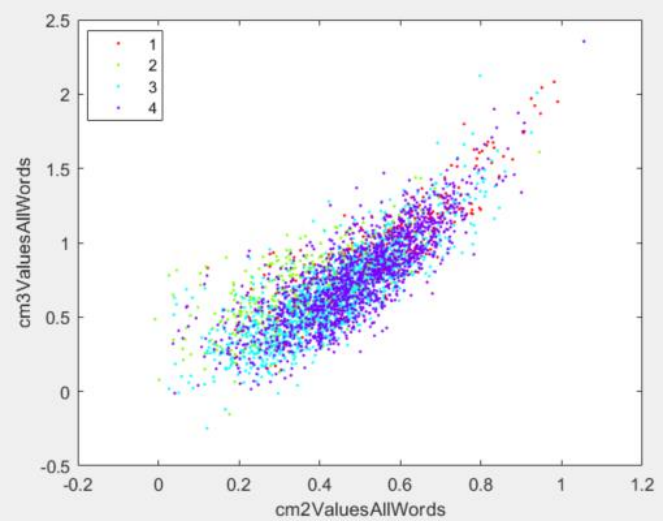
It seems different pairs have diffident ability to separate different words. In this case (cm0,cm1) pair has more discriminant power because distances among words are larger than distances among word computed by (cm2,cm3) pair

For all record files (1:Cat, 2:Dog, 3:Three, 4:Tree)

(cm0, cm1) pair

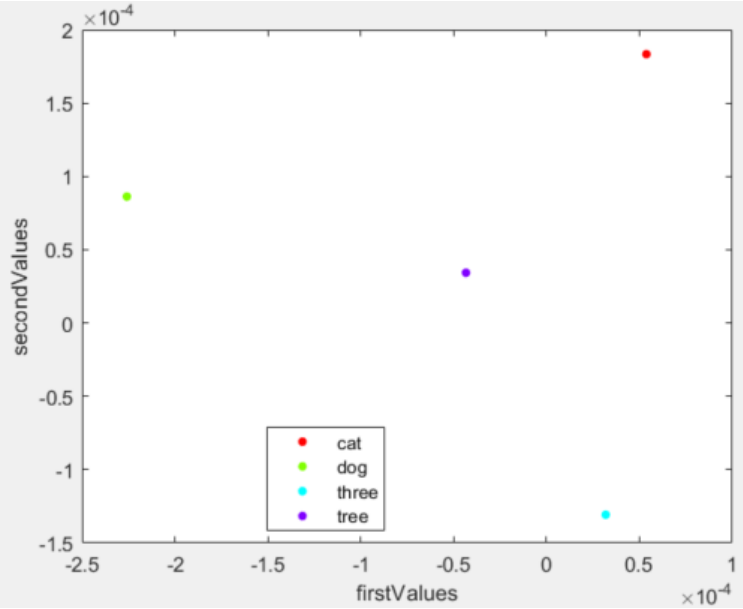


(cm2, cm3) pair (1:Cat, 2:Dog, 3:Three, 4:Tree)

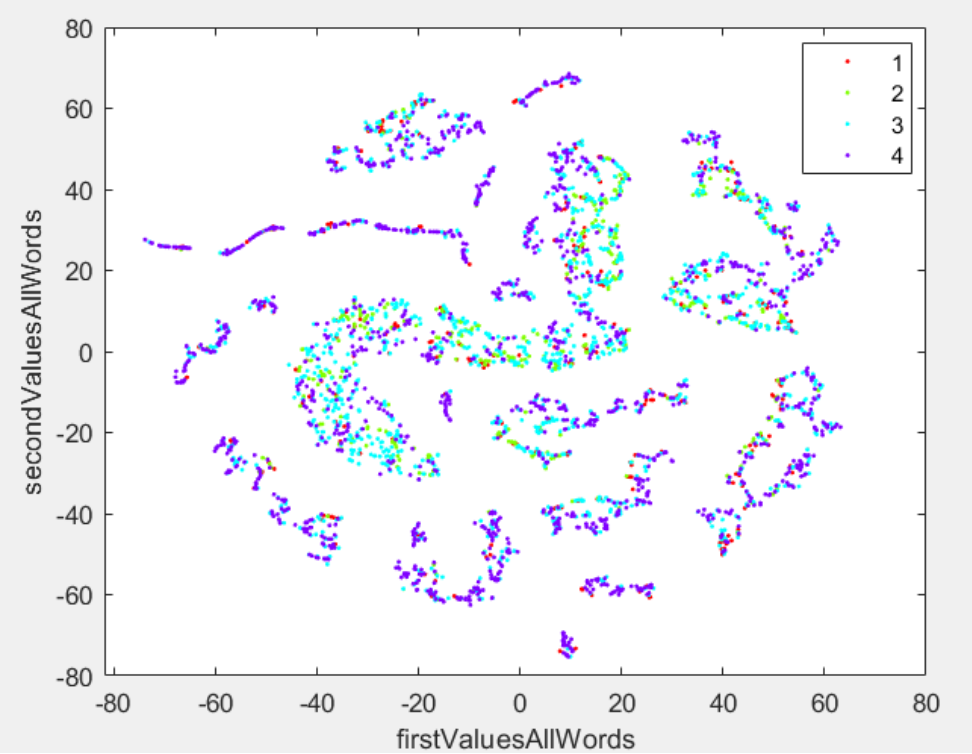


(b)

For just one record file for each word



For all record files (1:Cat, 2:Dog, 3:Three, 4:Tree)



(c)

For just one record file for each word

	Accuracy
Trial1	1
Trial2	1

For all record files

	Accuracy
Trial1	0.3237
Trial2	0.3238
Trial3	0.3238
Trial4	0.3238
Trial5	0.3237

Base line accuracy: 2144/(1522+1539+2144+1521) = 2144/6726 = 0.3188

Observation:

It looks like extracted features are not very separable based on the plot I draw above (but it is also possible I implemented some part wrong) Therefore, K mean clustering accuracy is not that different

from base line accuracy where it always chooses majority class. Since K mean clustering has randomness, it could cause different accuracy for each run.

Appendix

HW4_Pr1.m

```
%Add data set directories
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\cat')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\dog')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\three')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\tree')


%% PART 1
%(a)Time-domain Standardization
%Load raw speech signal
%cat
fileName_cat1 = '00b01445_nohash_0.wav';%1st file
fileName_cat2 = '00f0204f_nohash_0.wav';%2nd file
fileName_cat3 = '00f0204f_nohash_1.wav';%3rd file
fileName_cat4 = '00f0204f_nohash_2.wav';%4th file
fileName_cat5 = '0ac15fe9_nohash_0.wav';%7th file

[sn_cat1,~] = audioread(fileName_cat1);
[sn_cat2,~] = audioread(fileName_cat2);
[sn_cat3,~] = audioread(fileName_cat3);
[sn_cat4,~] = audioread(fileName_cat4);
[sn_cat5,~] = audioread(fileName_cat5);

%dog
fileName_dog1 = '00f0204f_nohash_1.wav';%1st
fileName_dog2 = '00f0204f_nohash_0.wav';%2nd
fileName_dog3 = '00f0204f_nohash_1.wav';%3rd
fileName_dog4 = '00f0204f_nohash_2.wav';%4th
fileName_dog5 = '0a7c2a8d_nohash_0.wav';%5th

[sn_dog1,~] = audioread(fileName_dog1);
[sn_dog2,~] = audioread(fileName_dog2);
[sn_dog3,~] = audioread(fileName_dog3);
[sn_dog4,~] = audioread(fileName_dog4);
[sn_dog5,~] = audioread(fileName_dog5);

%tree
fileName_tree1 = '0d393936_nohash_0.wav';%1st
fileName_tree2 = '0a7c2a8d_nohash_0.wav';%2nd
fileName_tree3 = '0ac15fe9_nohash_0.wav';%3rd
```

```
fileName_tree4 = '0b40aa8e_nohash_1.wav';%6th file
fileName_tree5 = '0b40aa8e_nohash_0.wav';%5th

[sn_tree1,~] = audioread(fileName_tree1);
[sn_tree2,~] = audioread(fileName_tree2);
[sn_tree3,~] = audioread(fileName_tree3);
[sn_tree4,~] = audioread(fileName_tree4);
[sn_tree5,~] = audioread(fileName_tree5);

%three
fileName_three1 = '0d53e045_nohash_0.wav';%1st
fileName_three2 = '00b01445_nohash_1.wav';%2nd
fileName_three3 = '00b01445_nohash_2.wav';%3rd
fileName_three4 = '0a9f9af7_nohash_0.wav';%4th
fileName_three5 = '0b40aa8e_nohash_0.wav';%5th

[sn_three1,~] = audioread(fileName_three1);
[sn_three2,~] = audioread(fileName_three2);
[sn_three3,~] = audioread(fileName_three3);
[sn_three4,~] = audioread(fileName_three4);
[sn_three5,~] = audioread(fileName_three5);

%Standardization
%cat
sn_std_cat1 = zscore(sn_cat1);
sn_std_cat2 = zscore(sn_cat2);
sn_std_cat3 = zscore(sn_cat3);
sn_std_cat4 = zscore(sn_cat4);
sn_std_cat5 = zscore(sn_cat5);

%dog
sn_std_dog1 = zscore(sn_dog1);
sn_std_dog2 = zscore(sn_dog2);
sn_std_dog3 = zscore(sn_dog3);
sn_std_dog4 = zscore(sn_dog4);
sn_std_dog5 = zscore(sn_dog5);

%tree
sn_std_tree1 = zscore(sn_tree1);
sn_std_tree2 = zscore(sn_tree2);
sn_std_tree3 = zscore(sn_tree3);
sn_std_tree4 = zscore(sn_tree4);
sn_std_tree5 = zscore(sn_tree5);
```

```
%three
sn_std_three1 = zscore(sn_three1);
sn_std_three2 = zscore(sn_three2);
sn_std_three3 = zscore(sn_three3);
sn_std_three4 = zscore(sn_three4);
sn_std_three5 = zscore(sn_three5);

%(b) Framing and Windowing
Fs=16000; %Sampling frequency 16000Hz
durationOfWindow = 0.025; %25 msec
lengthOfWindow25 = durationOfWindow * Fs ;

%generate Hamming window
hammingWindow25 = hamming(lengthOfWindow25);
%zero pad audio signal to avoid a case that the last window
overshoots the
%length of signal(original length = 16000 -> padded length
= 16080
paddedLength = 16080;
originalLength = 16000;

zeroPaddedSn_std_cat1 = zeros(1,paddedLength);
zeroPaddedSn_std_cat1(1:originalLength) = sn_std_cat1;

zeroPaddedSn_std_dog1 = zeros(1,paddedLength);
zeroPaddedSn_std_dog1(1:originalLength) = sn_std_dog1;

zeroPaddedSn_std_tree1 = zeros(1,paddedLength);
zeroPaddedSn_std_tree1(1:originalLength) = sn_std_tree1;

zeroPaddedSn_std_three1 = zeros(1,paddedLength);
zeroPaddedSn_std_three1(1:originalLength) = sn_std_three1;

%Framming
lengthOfOverlap10 = 240; %400-0.01*16000=240 i.e. shift
window by 10 msec
opt = 'nodelay';
framedZeroPaddedSn_std_cat1 =
buffer(zeroPaddedSn_std_cat1,lengthOfWindow25,lengthOfOverl
ap10,opt);

framedZeroPaddedSn_std_dog1 =
buffer(zeroPaddedSn_std_dog1,lengthOfWindow25,lengthOfOverl
ap10,opt);
```

```
framedZeroPaddedSn_std_tree1 =
buffer(zeroPaddedSn_std_tree1,lengthOfWindow25,lengthOfOver
lap10,opt);

framedZeroPaddedSn_std_three1 =
buffer(zeroPaddedSn_std_three1,lengthOfWindow25,lengthOfOve
rlap10,opt);

%windowing
windowedFramedSn_std_cat1 =
windowingOperator(framedZeroPaddedSn_std_cat1,hammingWindow
25);

windowedFramedSn_std_dog1 =
windowingOperator(framedZeroPaddedSn_std_dog1,hammingWindow
25);

windowedFramedSn_std_tree1 =
windowingOperator(framedZeroPaddedSn_std_tree1,hammingWindo
w25);

windowedFramedSn_std_three1 =
windowingOperator(framedZeroPaddedSn_std_three1,hammingWind
ow25);

%(c)Critical Bands and Equal-Loudness Filter
%Critical Bands Filter Bank and plot them
[criticalBandFilterBank,centralBarkArray] =
criticalBankFilterGenerator();

%Transform filters with U(w)
transformedBandFilterBank =
bankTransformer(criticalBandFilterBank);


%(d)Energy Calculation and Cubit-Root Compression
fileName_three_target =
'C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\three\f9273a21_noh
ash_1.wav';
sn_three_target = audioread(fileName_three_target);

%Standardization
sn_std_three_target = zscore(sn_three_target);
```

```
zeroPaddedSn_std_three_target = zeros(1,paddedLength);
zeroPaddedSn_std_three_target(1:originalLength) =
sn_std_three_target;

%Framming
lengthOfOverlap10 = 240; %400-0.01*16000=240 i.e. shift
window by 10 msec
opt = 'nodelay';
framedZeroPaddedSn_std_three_target =
buffer(zeroPaddedSn_std_three_target,lengthOfWindow25,lengt
hOfOverlap10,opt);

%windowing
windowedFramedSn_std_three_target =
windowingOperator(framedZeroPaddedSn_std_three_target,hammi
ngWindow25);

%Work with 70th frame of the signal sn_70
orderOfFrame_three = 70;
sn_70_three =
windowedFramedSn_std_three_target(:,orderOfFrame_three);
sn_70_three_tr = sn_70_three'; %make it 1x400 size

%(d)-(i)
N_DFT = 1024;
S_k_70_three = fft(sn_70_three_tr,N_DFT);
%squared S[k]
squaredS_k_70_three = abs(S_k_70_three).^2;
%dB scale
logSquaredS_k_70_three = 10*log(squaredS_k_70_three);

%(d)-(ii)
energyWithOriFilterBank =
energyCoefficientCalculator(criticalBandFilterBank,S_k_70_t
hree);
logEnergyWithOriFilterBank =
10*log(energyWithOriFilterBank);
%(d)-(iii)
energyWithTransformedFilterBank =
energyCoefficientCalculator(transformedBandFilterBank,S_k_7
0_three);
logEnergyWithTransformedFilterBank =
10*log(energyWithTransformedFilterBank);
```

```
%(d)-(iv)
cubic_rootCoefficient =
energyWithTransformedFilterBank.^(0.33); %G_i(j). Formula
is given on HW4 description
logCubic_rootCoefficient = 10*log(cubic_rootCoefficient);
%(d)-plotting
%Convert Central frequency in bark to in Hz
centralHzArray = 600.*sinh(centralBarkArray./6);
%Frequency Array for DFT
freqArrayForDFT = [1:N_DFT].*(Fs/N_DFT) ;

figure(5)
hold on
%plot values within index 1~512
plot(freqArrayForDFT(1:N_DFT/2),logSquaredS_k_70_three(1:N_
DFT/2))
plot(centralHzArray,logEnergyWithOriFilterBank)
plot(centralHzArray,logEnergyWithTransformedFilterBank)
plot(centralHzArray,logCubic_rootCoefficient)
xlabel('Frequency(Hz)')
ylabel('Values')
legend('Power spectrum','energy before transform','energy
after transform','coefficient G')
hold off

%(e)
%(i),(ii),(iii)
[LPC_Coefficients,
cepstralCoefficientsBeforeLifting,cepstralCoefficientsAfter
Lifting] =
LPCNCeptralCoefficientCalculator(cubic_rootCoefficient);


%% PART 2
%(a)
PLPbasedFeatures_cat1 = PLPfeatureExtractor(fileName_cat1);
PLPbasedFeatures_dog1 = PLPfeatureExtractor(fileName_dog1);
PLPbasedFeatures_three1 =
PLPfeatureExtractor(fileName_three1);
PLPbasedFeatures_tree1 =
PLPfeatureExtractor(fileName_tree1);

%Extract (cm[0],cm[1]) pair
```

```
cm_0Values =
[PLPbasedFeatures_cat1(1,1);PLPbasedFeatures_dog1(1,1);PLPb
asedFeatures_three1(1,1);PLPbasedFeatures_tree1(1,1)];
cm_1Values =
[PLPbasedFeatures_cat1(1,2);PLPbasedFeatures_dog1(1,2);PLPb
asedFeatures_three1(1,2);PLPbasedFeatures_tree1(1,2)];
labels = ["cat";"dog";"three";"tree"];

figure(6)
gscatter(cm_0Values,cm_1Values,labels)

%Extract other pairs
%(cm[2],cm[3])
cm_2Values =
[PLPbasedFeatures_cat1(1,3);PLPbasedFeatures_dog1(1,3);PLPb
asedFeatures_three1(1,3);PLPbasedFeatures_tree1(1,3)];
cm_3Values =
[PLPbasedFeatures_cat1(1,4);PLPbasedFeatures_dog1(1,4);PLPb
asedFeatures_three1(1,4);PLPbasedFeatures_tree1(1,4)];

figure(7)
gscatter(cm_2Values,cm_3Values,labels)

%(b)2-dimensional t-Distributed Stochastic Neighbor
Embeddings
tsneFeature_cat1 = tsne(PLPbasedFeatures_cat1);
tsneFeature_dog1 = tsne(PLPbasedFeatures_dog1);
tsneFeature_three1 = tsne(PLPbasedFeatures_three1);
tsneFeature_tree1 = tsne(PLPbasedFeatures_tree1);
%Form a proper data size for plotting

firstValues =
[tsneFeature_cat1(1,1);tsneFeature_dog1(1,1);tsneFeature_th
ree1(1,1);tsneFeature_tree1(1,1)];
secondValues =
[tsneFeature_cat1(1,2);tsneFeature_dog1(1,2);tsneFeature_th
ree1(1,2);tsneFeature_tree1(1,2)];

figure(8)
gscatter(firstValues,secondValues,labels)

%(c)
%Construct data set for K means algorithm
```

```
X_train =
[PLPbasedFeatures_cat1 ;PLPbasedFeatures_dog1 ;PLPbasedFeat
ures_three1 ;PLPbasedFeatures_tree1 ];
trueLabels = [1;2;3;4]; %1:cat, 2:dog, 3:three, 4:tree
numOfClusters = 4; %4 words
predictedLabels = kmeans(X_train,numOfClusters);

[y_pred_matched, acc] = optimalMatch(predictedLabels,
trueLabels);



%Extract features from all files
%cat
numOfFileNames_cat = 1733;%211 non 16000 length files
fileNames_cat =
nameExtractor('cat',numOfFileNames_cat); %1733x1 size
numOfNon16000LengthFiles_cat = 211;
X_train_cat =
trainSetConstructor(fileNames_cat,numOfNon16000LengthFiles_
cat);%1522x13 size
%dog
numOfFileNames_dog = 1746;
numOfNon16000LengthFiles_dog = 207;
fileNames_dog =
nameExtractor('dog',numOfFileNames_dog); %1746x1 size
X_train_dog =
trainSetConstructor(fileNames_dog,numOfNon16000LengthFiles_
dog);%1539x13 size
%three
numOfFileNames_three = 2356;
numOfNon16000LengthFiles_three = 212;
fileNames_three =
nameExtractor('three',numOfFileNames_three); %2356x1 size
X_train_three =
trainSetConstructor(fileNames_three,numOfNon16000LengthFile
s_three);%2144x13 size
%tree
numOfFileNames_tree = 1733;
numOfNon16000LengthFiles_tree = 212;
fileNames_tree =
nameExtractor('tree',numOfFileNames_tree); %1733x1 size
```

```
X_train_tree =
trainSetConstructor(fileNames_tree,numOfNon16000LengthFiles
_tree);%1521x13 size


%Construct all train set and labels
X_train_allWords =
[X_train_cat;X_train_dog;X_train_three;X_train_tree];
numOfDataPoints_cat = size(X_train_cat,1);
numOfDataPoints_dog = size(X_train_dog,1);
numOfDataPoints_three = size(X_train_three,1);
numOfDataPoints_tree = size(X_train_tree,1);

Y_labels_allWords =
YLabelGenerator(numOfDataPoints_cat,numOfDataPoints_dog,num
OfDataPoints_three,numOfDataPoints_tree);

%(a) for all words
%Extract (cm[0],cm[1]) pair
cm0ValuesAllWords = X_train_allWords(:,1);
cm1ValuesAllWords = X_train_allWords(:,2);
%Extract other pair
cm2ValuesAllWords = X_train_allWords(:,2);
cm3ValuesAllWords = X_train_allWords(:,3);

cm4ValuesAllWords = X_train_allWords(:,4);
cm5ValuesAllWords = X_train_allWords(:,5);

cm6ValuesAllWords = X_train_allWords(:,6);
cm7ValuesAllWords = X_train_allWords(:,7);

cm8ValuesAllWords = X_train_allWords(:,8);
cm9ValuesAllWords = X_train_allWords(:,9);

cm10ValuesAllWords = X_train_allWords(:,10);
cm11ValuesAllWords = X_train_allWords(:,11);

cm12ValuesAllWords = X_train_allWords(:,12);
cm13ValuesAllWords = X_train_allWords(:,13);



figure(9)
```

```
gscatter(cm0ValuesAllWords,cm1ValuesAllWords,Y_labels_allWo
rds);
figure(10)
gscatter(cm2ValuesAllWords,cm3ValuesAllWords,Y_labels_allWo
rds);

%Check all figures of possible 156(=169-13) pairs
%dummy =
allFiguresChecker(X_train_allWords,Y_labels_allWords);

%(b) for all words
tsneFeatureAllWords = tsne(X_train_allWords);
firstValuesAllWords = tsneFeatureAllWords(:,1);
secondValuesAllWords = tsneFeatureAllWords(:,2);

figure(11)
gscatter(firstValuesAllWords,secondValuesAllWords,Y_labels_
allWords);

%(c) Kmeans for all words
numOfClustersAllWords = 4; %4 words
predictedLabelsAllWords =
kmeans(X_train_allWords,numOfClustersAllWords);

[y_pred_matched_allwords, accAllWords] =
optimalMatch(predictedLabelsAllWords, Y_labels_allWords);

%Calculate accuracy
```

**********allFiguresChecker.m

```
function dummy =
allFiguresChecker(X_train_allWords,Y_labels_allWords)
%ALLFIGURESCHECKER Summary of this function goes here
%   Detailed explanation goes here


numOfFeatures = size(X_train_allWords,2);%e.g. 13
dummy = 1
for orderOfFeature = 1:numOfFeatures
    for orderOfFeature1 = 1:numOfFeatures
        figure(20)
        currentC1 = X_train_allWords(:,orderOfFeature);
        currentC2 = X_train_allWords(:,orderOfFeature1);
```

```
        gscatter(currentC1,currentC2,Y_labels_allWords);
        X = ['current pair
('num2str(orderOfFeature),',',num2str(orderOfFeature1),')'
];
        disp(X)
    end
end


end
```

********** bankTransformer.m

```
function normalizedTransformedBandFilterBank =
bankTransformer(criticalBandFilterBank)
%UNTITLED Summary of this function goes here
%Transform given filter bank with U(w)
%U(w) is explained in HW4 description
%   Detailed explanation goes here

N_DFT = 1024;
halfN_DFT = N_DFT/2;

%Generate frequency and Bark values
Fs =16000;
nyquist = Fs/2;
frequencyInterval = nyquist/(halfN_DFT-1); % need to
subtract by -1 because frequency start from 0

freqValArray = [0:(halfN_DFT-1)].*frequencyInterval;% need
to subtract by -1 because frequency start from 0



numOfFilters = 20; %Number of filters needed to make
distance of each filter's centeral frequency approximatley
1 Bark

%Generate central bark values
lastBark = 6*asinh(nyquist/600);
barkInterval = lastBark/(numOfFilters-1);
centralBarkArray = [0:(numOfFilters-1)].*barkInterval;

transformedBandFilterBank = criticalBandFilterBank;
```

```
%trial 1
%Multiply corresponding U(w) to each filter
for filterOrder = 1:numOfFilters
    currentBcj = centralBarkArray(1,filterOrder);
    Uw = UwGenerator(currentBcj,Fs);
    transformedBandFilterBank(filterOrder,:) =
criticalBandFilterBank(filterOrder,:).*Uw;
end

% %trial 2
% %Multiply corresponding U(w) to each filter
% for filterOrder = 1:numOfFilters
%     Uw = UwGeneratorV1(freqValArray,Fs);
%     transformedBandFilterBank(filterOrder,:) =
criticalBandFilterBank(filterOrder).*Uw;
% end

%find maximum value of entire filter bank
maxInFilterBank =
maxFinder(transformedBandFilterBank,numOfFilters);


%Normalize filter values with max normalization method.
normalizedTransformedBandFilterBank =
transformedBandFilterBank;
for filterOrder = 1:numOfFilters

    normalizedTransformedBandFilterBank(filterOrder,:) =
transformedBandFilterBank(filterOrder,:)./maxInFilterBank;
end


%Plot each filter
figure(1)
t1=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(1,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(2,:))

nexttile
```

```
plot(freqValArray,normalizedTransformedBandFilterBank(3,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(4,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(5,:))

title(t1,'Filters 1~5')
xlabel(t1,'Frequency(Hz)')
ylabel(t1,'trainformed H_j Value')

figure(2)
t2=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(6,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(7,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(8,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(9,:))

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(10,:)
)

title(t2,'Filters 6~10')
xlabel(t2,'Frequency(Hz)')
ylabel(t2,'trainformed H_j Value')


figure(3)
t3=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(11,:)
)

nexttile
```

```
plot(freqValArray,normalizedTransformedBandFilterBank(12,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(13,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(14,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(15,:)
)

title(t3,'Filters 11~15')
xlabel(t3,'Frequency(Hz)')
ylabel(t3,'trainformed H_j Value')


figure(4)
t4=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(16,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(17,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(18,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(19,:)
)

nexttile
plot(freqValArray,normalizedTransformedBandFilterBank(20,:)
)

% Add shared title and axis labels
```

```
title(t4,'Filters 16~20')
xlabel(t4,'Frequency(Hz)')
ylabel(t4,'trainformed H_j Value')


end
```

********** bankTransformerV1.m

```
function normalizedTransformedBandFilterBank =
bankTransformerV1(criticalBandFilterBank)
%UNTITLED Summary of this function goes here
%Transform given filter bank with U(w)
%U(w) is explained in HW4 description
%   Detailed explanation goes here

N_DFT = 1024;
halfN_DFT = N_DFT/2;

%Generate frequency and Bark values
Fs =16000;
nyquist = Fs/2;
frequencyInterval = nyquist/(halfN_DFT-1); % need to
subtract by -1 because frequency start from 0

freqValArray = [0:(halfN_DFT-1)].*frequencyInterval;% need
to subtract by -1 because frequency start from 0



numOfFilters = 20; %Number of filters needed to make
distance of each filter's centeral frequency approximatley
1 Bark

%Generate central bark values
lastBark = 6*asinh(nyquist/600);
barkInterval = lastBark/(numOfFilters-1);
centralBarkArray = [0:(numOfFilters-1)].*barkInterval;

transformedBandFilterBank = criticalBandFilterBank;

%trial 1
```

```
%Multiply corresponding U(w) to each filter
for filterOrder = 1:numOfFilters
    currentBcj = centralBarkArray(1,filterOrder);
    Uw = UwGenerator(currentBcj,Fs);
    transformedBandFilterBank(filterOrder,:) =
criticalBandFilterBank(filterOrder,:).*Uw;
end


%find maximum value of entire filter bank
maxInFilterBank =
maxFinder(transformedBandFilterBank,numOfFilters);


%Normalize filter values with max normalization method.
normalizedTransformedBandFilterBank =
transformedBandFilterBank;
for filterOrder = 1:numOfFilters

    normalizedTransformedBandFilterBank(filterOrder,:) =
transformedBandFilterBank(filterOrder,:)./maxInFilterBank;
end




end
```

********** cepstralCoefficientCalculator.m

```
function cepstralCoefficientsBeforeLifting =
cepstralCoefficientCalculator(LPC_Coefficients,gain)
%CEPSTRALCOEFFICIENTCALCULATOR Summary of this function
goes here
%Calculate cepstral coefficient using reculsive
relationship with given LPC
%coefficients and gain
%   Detailed explanation goes here

numOfCoefficients = size(LPC_Coefficients,2); %e.g. 13
including h[0]
```

```
cepstralCoefficientsBeforeLifting =
zeros(1,numOfCoefficients);

for orderOfCoefficients = 0: numOfCoefficients-1
    currentIndex = orderOfCoefficients +1;
    if(orderOfCoefficients == 0)
        cepstralCoefficientsBeforeLifting(1,currentIndex) =
log(gain);
    elseif(orderOfCoefficients == 1)
        cepstralCoefficientsBeforeLifting(1,currentIndex) =
LPC_Coefficients(1,currentIndex);
    elseif(orderOfCoefficients > 0)
        sumVal =
sumCalculator(LPC_Coefficients,cepstralCoefficientsBeforeLi
fting,orderOfCoefficients);
        cepstralCoefficientsBeforeLifting(1,currentIndex) =
LPC_Coefficients(1,currentIndex) + sumVal ;
    end
end


end
```

********** criticalBankFilterGenerator.m

```
function [criticalBandFilterBank,centralBarkArray] =
criticalBankFilterGenerator()
%CRITICALBANKFILTERGENERATOR Summary of this function goes
here
%   Detailed explanation goes here

N_DFT = 1024;
halfN_DFT = N_DFT/2;
numOfFilters = 20; %Number of filters needed to make
distance of each filter's centeral frequency approximatley
1 Bark

%Generate frequency and Bark values
Fs =16000;
nyquist = Fs/2;
frequencyInterval = nyquist/(halfN_DFT-1); % need to
subtract by -1 because frequency start from 0
```

```
freqValArray = [0:(halfN_DFT-1)].*frequencyInterval;% need
to subtract by -1 because frequency start from 0
barkValArray = 6*asinh(freqValArray./600);

%Generate central bark values
lastBark = 6*asinh(nyquist/600);
barkInterval = lastBark/(numOfFilters-1);
centralBarkArray = [0:(numOfFilters-1)].*barkInterval;

criticalBandFilterBank = zeros(numOfFilters, halfN_DFT);

%Generate filters and save them to the filter bank
for filterOrder = 1:numOfFilters
    currentCentralBark = centralBarkArray(1,filterOrder);
    criticalBandFilterBank(filterOrder,:) =
HjFilterGenerator(barkValArray,currentCentralBark);

end

%Plot each filter
figure(1)
t1=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,criticalBandFilterBank(1,:))

nexttile
plot(freqValArray,criticalBandFilterBank(2,:))

nexttile
plot(freqValArray,criticalBandFilterBank(3,:))

nexttile
plot(freqValArray,criticalBandFilterBank(4,:))

nexttile
plot(freqValArray,criticalBandFilterBank(5,:))

title(t1,'Filters 1~5')
xlabel(t1,'Frequency(Hz)')
ylabel(t1,'H_j Value')

figure(2)
t2=tiledlayout(5,1); % Requires R2019b or later
```

```
nexttile
plot(freqValArray,criticalBandFilterBank(6,:))

nexttile
plot(freqValArray,criticalBandFilterBank(7,:))

nexttile
plot(freqValArray,criticalBandFilterBank(8,:))

nexttile
plot(freqValArray,criticalBandFilterBank(9,:))

nexttile
plot(freqValArray,criticalBandFilterBank(10,:))

title(t2,'Filters 6~10')
xlabel(t2,'Frequency(Hz)')
ylabel(t2,'H_j Value')


figure(3)
t3=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,criticalBandFilterBank(11,:))

nexttile
plot(freqValArray,criticalBandFilterBank(12,:))

nexttile
plot(freqValArray,criticalBandFilterBank(13,:))

nexttile
plot(freqValArray,criticalBandFilterBank(14,:))

nexttile
plot(freqValArray,criticalBandFilterBank(15,:))

title(t3,'Filters 11~15')
xlabel(t3,'Frequency(Hz)')
ylabel(t3,'H_j Value')


figure(4)
```

```
t4=tiledlayout(5,1); % Requires R2019b or later

nexttile
plot(freqValArray,criticalBandFilterBank(16,:))

nexttile
plot(freqValArray,criticalBandFilterBank(17,:))

nexttile
plot(freqValArray,criticalBandFilterBank(18,:))

nexttile
plot(freqValArray,criticalBandFilterBank(19,:))

nexttile
plot(freqValArray,criticalBandFilterBank(20,:))

% Add shared title and axis labels
title(t4,'Filters 16~20')
xlabel(t4,'Frequency(Hz)')
ylabel(t4,'H_j Value')

end
```

********** criticalBankFilterGeneratorV1.m

```
function [criticalBandFilterBank,centralBarkArray] =
criticalBankFilterGeneratorV1()
%CRITICALBANKFILTERGENERATOR Summary of this function goes
here
%   Detailed explanation goes here

N_DFT = 1024;
halfN_DFT = N_DFT/2;
numOfFilters = 20; %Number of filters needed to make
distance of each filter's centeral frequency approximatley
1 Bark

%Generate frequency and Bark values
Fs =16000;
nyquist = Fs/2;
```

```
frequencyInterval = nyquist/(halfN_DFT-1); % need to
subtract by -1 because frequency start from 0

freqValArray = [0:(halfN_DFT-1)].*frequencyInterval;% need
to subtract by -1 because frequency start from 0
barkValArray = 6*asinh(freqValArray./600);

%Generate central bark values
lastBark = 6*asinh(nyquist/600);
barkInterval = lastBark/(numOfFilters-1);
centralBarkArray = [0:(numOfFilters-1)].*barkInterval;

criticalBandFilterBank = zeros(numOfFilters, halfN_DFT);

%Generate filters and save them to the filter bank
for filterOrder = 1:numOfFilters
    currentCentralBark = centralBarkArray(1,filterOrder);
    criticalBandFilterBank(filterOrder,:) =
HjFilterGenerator(barkValArray,currentCentralBark);

end



end
```

********** energyCoefficientCalculator.m

```
function energyCoefficient =
energyCoefficientCalculator(bandFilterBank,S_k)
%ENERGYCOEFFICIENTCALCULATOR Summary of this function goes
here
%Calculate E_i(j) based on S[k] and band filter bank
%Formula is on HW4 description
%   Detailed explanation goes here

numOfFilter = size(bandFilterBank,1); % e.g. Q = 20

energyCoefficient = zeros(1,numOfFilter); %
halfN_DFT = 512;
portionOfS_k = S_k(1:halfN_DFT);
```

```
for orderOfCoefficient = 2:(numOfFilter-1)
    absSquarePortionS_k = abs(portionOfS_k).^2; %|S_i[k]|^2
    absCurrentFilter =
abs(bandFilterBank(orderOfCoefficient,:)) ; %|H_j[k]|
    energyCoefficient(1,orderOfCoefficient) =
sum(absSquarePortionS_k.*absCurrentFilter) ;
end

%for the first coefficient
energyCoefficient(1,1) = energyCoefficient(1,2);
%for the last coefficient
energyCoefficient(1,numOfFilter) =
energyCoefficient(1,numOfFilter-1);




end
```

********** featureVectorCalculator.m

```
function featureVector =
featureVectorCalculator(cepstralCoefficientsAfterLiftingFor
AllFrame)
%FEATUREVECTORCALCULATOR Summary of this function goes here
%   Detailed explanation goes here

lengthOfFeatureVector =
size(cepstralCoefficientsAfterLiftingForAllFrame,2); % 13
numOfFrames =
size(cepstralCoefficientsAfterLiftingForAllFrame,1);%e.g.99

featureVector = zeros(1,lengthOfFeatureVector);

for orderOfFeature = 1:lengthOfFeatureVector
    currentCoefficientForAllFrame =
cepstralCoefficientsAfterLiftingForAllFrame(:,orderOfFeatur
e)'; %1x99 size
    currentFeatureVal =
(sum(currentCoefficientForAllFrame))/numOfFrames;
    featureVector(1,orderOfFeature) = currentFeatureVal;
end
```

```
end
```

********** HjFilterGenerator.m

```
function HjFilterValArray =
HjFilterGenerator(B_ValArray,Bcj)
%HJFILTERGENERATOR Summary of this function goes here
%Generate H_j filter based on B_cj(jth H_j filter)
%B_ValArray: B value array in Bark
%Bcj: Center frequency in Bark
%   Detailed explanation goes here


lengthOfB_ValArray = size(B_ValArray,2); %e.g. 512
HjFilterValArray = zeros(1,lengthOfB_ValArray);

%Formular is in HW4 description
for BValOrder = 1:lengthOfB_ValArray
    curretB = B_ValArray(1,BValOrder);
    if( curretB<(-2.5+Bcj) )
        HjFilterValArray(1,BValOrder) = 0;
    elseif( curretB >= (-2.5+Bcj) && curretB <= (-
0.5+Bcj) )
        HjFilterValArray(1,BValOrder) = 10^(curretB-
Bcj+0.5);
    elseif( curretB > (-0.5+Bcj) && curretB < (0.5+Bcj) )
        HjFilterValArray(1,BValOrder) = 1;
    elseif( curretB >= (0.5+Bcj) && curretB <= (1.3+Bcj) )
        HjFilterValArray(1,BValOrder) = 10^((-
2.5)*(curretB-Bcj-0.5));
    elseif( curretB > (1.3+Bcj) )
        HjFilterValArray(1,BValOrder) = 0;
    end
end

end
```

********** lifterGenerator.m

```
function w_n = lifterGenerator(w_n_length)
%LIFTERGENERATOR Summary of this function goes here
%   Detailed explanation goes here

w_n = zeros(1,w_n_length);

for index = 1:w_n_length
    current_n = index -1 ; %0~12
    if(current_n == 0)
        w_n(1,index) = 1;
    else
        w_n(1,index) = current_n^0.6 ;
    end
end


end
```

********** LPCNCeptralCoefficientCalculator.m

```
function [LPC_Coefficients,
cepstralCoefficientsBeforeLifting,cepstralCoefficientsAfter
Lifting] =
LPCNCeptralCoefficientCalculator(cubic_rootCoefficient)
%LPCNCEPTRALCOEFFICIENTCALCULATOR Summary of this function
goes here
%Calculate LPC coefficients, cepstrall coefficients before
lifting,
%cepstral coefficients after lifting
%   Detailed explanation goes here

%Expand G_i
lengthOfG_i = size(cubic_rootCoefficient,2) ; % e.g. Q = 20
= length
lengthOfExpanded_G_i = lengthOfG_i + (lengthOfG_i-2) ; %
exclude G_i(Q) and G_i(1) when we expand G_i
expanded_G_i = zeros(1,lengthOfExpanded_G_i);

expanded_G_i(1,1:lengthOfG_i) = cubic_rootCoefficient;

reverseQIndex = lengthOfG_i - 1;
```

```
for orderOfRestIndex = (lengthOfG_i+1):lengthOfExpanded_G_i
    expanded_G_i(1,orderOfRestIndex) =
cubic_rootCoefficient(reverseQIndex);
    reverseQIndex = reverseQIndex - 1;
end

%Calculate inverse of expanded G_i
inverseExpG_i = ifft(expanded_G_i,lengthOfExpanded_G_i);
portionOfInverseExpG_i = inverseExpG_i(1:lengthOfG_i);%Kep
only the first half

%Find LPC coefficients with order of p = 12
p = 12;
[LPC_Coefficients,predictionErr] =
levinson(portionOfInverseExpG_i,p);
gain = predictionErr^0.5;

%1x13 size
cepstralCoefficientsBeforeLifting =
cepstralCoefficientCalculator(LPC_Coefficients,gain);

%Generate a given lifter in HW4
w_n_length = size(cepstralCoefficientsBeforeLifting,2); %
e.g. 13
w_n = lifterGenerator(w_n_length);
cepstralCoefficientsAfterLifting =
cepstralCoefficientsBeforeLifting.*w_n;



end
```

********** maxFinder.m

```
function maxInFilterBank =
maxFinder(transformedBandFilterBank,numOfFilters)
%MAXIMUMFINDER Summary of this function goes here
%   Detailed explanation goes here

maxInFilterBank = -100000000000000;

for filterOrder = 1:numOfFilters
```

```
    currentMax =
max(transformedBandFilterBank(filterOrder,:));
    if(currentMax >=maxInFilterBank)
        maxInFilterBank = currentMax;
    end
end


end
```

********** nameExtractor.m

```
function fileNames =
nameExtractor(targetFolderName,numOfFileNames)
%NAMEEXTRACTOR Summary of this function goes here
%Extract all file names in targer folder
%   Detailed explanation goes here

folderInfo = dir(targetFolderName);

fileNames = cell(numOfFileNames,1);
numOfExtraFiles = 2; %For some reason there are always 2
more file names such as '.'and '..'

for fileOrder = 1:numOfFileNames
    currentIndex = fileOrder+numOfExtraFiles;%Start from 3
    fileNames(fileOrder,1) =
{folderInfo(currentIndex).name};
end


end
```

********** PLPfeatureExtractor.m

```
function featureVector = PLPfeatureExtractor(audioFileName)
%PLPFEATUREEXTRACTOR Summary of this function goes here
%Extract PLP coefficients based features from an audio
signal
%   Detailed explanation goes here
```

```
[sn,Fs] = audioread(audioFileName); %sn: audio signal, Fs:
Sampling frequency
orderOfLPC = 12 ;
N_DFT = 1024;

%Standardization
sn_std = zscore(sn);

%Framing and Windowing
durationOfWindow = 0.025; %25 msec
lengthOfWindow25 = durationOfWindow * Fs ;
%generate Hamming window
hammingWindow25 = hamming(lengthOfWindow25);
%zero pad audio signal to avoid a case that the last window
overshoots the
%length of signal(original length = 16000 -> padded length
= 16080
paddedLength = 16080;
originalLength = 16000;

zeroPaddedSn_std = zeros(1,paddedLength);
zeroPaddedSn_std(1:originalLength) = sn_std;

%Framming
lengthOfOverlap10 = 240; %400-0.01*16000=240 i.e. shift
window by 10 msec
opt = 'nodelay';
framedZeroPaddedSn_std =
buffer(zeroPaddedSn_std,lengthOfWindow25,lengthOfOverlap10,
opt);

%windowing
windowedFramedSn_std =
windowingOperator(framedZeroPaddedSn_std,hammingWindow25);

%Critical Bands Filter Bank
[criticalBandFilterBank,~] =
criticalBankFilterGeneratorV1();

%Transform filters with U(w)
transformedBandFilterBank =
bankTransformerV1(criticalBandFilterBank);

%Take DFT onto windowed sn
```

```
numOfFrames = size(windowedFramedSn_std,2); %e.g.99 frames
numOfFilters = size(transformedBandFilterBank,1); %e.g.20
filters
windowedFramedSn_std_tr = windowedFramedSn_std';
S_kForAllFrames = zeros(numOfFrames,N_DFT); %99x1024
for orderOfFrame = 1:numOfFrames
    currnetWindowedFramedSn_std =
windowedFramedSn_std_tr(orderOfFrame,:);
    S_kForAllFrames(orderOfFrame,:) =
fft(currnetWindowedFramedSn_std,N_DFT);%e.g.99(frames)x1024
(frameLength)
end

energyWithTransformedFilterBankForAllFrames =
zeros(numOfFrames,numOfFilters); %99x20 size

%calculate energy coefficient for all framed S_k
for orderOfFrame = 1:numOfFrames
    currentFrameSk = S_kForAllFrames(orderOfFrame,:);
    currentEnergyCoefficient =
energyCoefficientCalculator(transformedBandFilterBank,curre
ntFrameSk);

energyWithTransformedFilterBankForAllFrames(orderOfFrame,:)
= currentEnergyCoefficient;
end

%Calculate G for all frames
cubic_rootCoefficientForAllFrames =
energyWithTransformedFilterBankForAllFrames.^(0.33); %99x20
size

%Calculate cepstral coefficients after liting for all
frames
cepstralCoefficientsAfterLiftingForAllFrame =
zeros(numOfFrames,orderOfLPC+1); %e.g. 99x(12+1)

for orderOfFrame = 1:numOfFrames
    currentCubitCoefficient =
cubic_rootCoefficientForAllFrames(orderOfFrame,:);
    [~, ~,currentCepstralCoefficientsAfterLifting] =
LPCNCeptralCoefficientCalculator(currentCubitCoefficient);
```

```
cepstralCoefficientsAfterLiftingForAllFrame(orderOfFrame,:)
= currentCepstralCoefficientsAfterLifting;
end

%Construct feature vector
featureVector =
featureVectorCalculator(cepstralCoefficientsAfterLiftingFor
AllFrame);



end
```

********** signalLengthCalculator.m

```
function currentSignalLength =
signalLengthCalculator(currentFileName)
%SIGNALLENGTHCALCULATOR Summary of this function goes here
%   Detailed explanation goes here

currentSn = audioread(currentFileName);
currentSignalLength = size(currentSn,1);

end
```

********** sumCalculator.m

```
function sumVal =
sumCalculator(LPC_Coefficients,cepstralCoefficientsBeforeLi
fting,orderOfCoefficients)
%SUMCALCULATOR Summary of this function goes here
%Calculate summation term in recursive relationship
%   Detailed explanation goes here

n = orderOfCoefficients;

%partioning h[n] for summation
startingIndex = 2;
```

```
portion_hn =
cepstralCoefficientsBeforeLifting(1,startingIndex:n);
portion_ak = LPC_Coefficients(1,startingIndex:n);

ak_index = n-1 ;
sumVal = 0;
for hn_Index = 1:n-1
    sumVal = sumVal +
(hn_Index/n)*portion_hn(1,hn_Index)*portion_ak(1,ak_index);
    ak_index = ak_index -1 ;
end




end
```

********** trainSetConstructor.m

```
function X_train =
trainSetConstructor(fileNames,numOfNon16000LengthFiles)
%TRAINSETCONSTRUCTOR Summary of this function goes here
%   Detailed explanation goes here
%Add data set directories
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\cat')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\dog')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\three')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homeowrk4\tree')

numOfFiles = size(fileNames,1);
numOfFeatureValues = 13 ; % assume it is known in this
practice.

X_train = zeros(numOfFiles-
numOfNon16000LengthFiles,numOfFeatureValues); % e.g.
1733x13 size
counter = 0;
X_train_index = 1;
for fileOrder = 1:numOfFiles
    currentFileName = fileNames{fileOrder};
    currentSignalLength =
signalLengthCalculator(currentFileName);
```

```
    if(currentSignalLength == 16000)
        X_train(X_train_index,:) =
PLPfeatureExtractor(currentFileName);
        X_train_index = X_train_index +1;
    else
        counter = counter +1 ;

    end
end

disp(counter)

end
```

********** UwGenerator.m

```
function Uw = UwGenerator(currentBcj,Fs)
%UWGENERATOR Summary of this function goes here
%Generate U(w) based on the currentB
%   Detailed explanation goes here

%Mapping formular from Bcj to wcj
wcj = 1200*pi*sinh(currentBcj/6);

%Fs=16000>10000hz in this practice
if(Fs<=10000)
    Uw =
( wcj^4*(wcj^2+56.8*(10^6)) )/((wcj^2+6.3*(10^6))*(wcj^2+0.
38*(10^9)));
elseif(Fs>10000)
    Uw =
( wcj^4*(wcj^2+56.8*(10^6)) )/(((wcj^2+6.3*(10^6))^2)*(wcj^
2+0.38*(10^9))*(wcj^6+9.58*(10^26)));
end


end
```

********** UwGeneratorV1.m

```
function Uw = UwGeneratorV1(freqValArray,Fs)
%UWGENERATORV1 Summary of this function goes here
%   Detailed explanation goes here
%Mapping formular from Bcj to wcj
wcjArray = (2*pi).*freqValArray;

%Fs=16000>10000hz in this practice
if(Fs<=16000)
    Uw =
( wcjArray.^4.*(wcjArray.^2+56.8*(10^6)) )./((wcjArray.^2+6
.3*(10^6)).*(wcjArray.^2+0.38*(10^9)));
elseif(Fs>16000)
    Uw =
( wcjArray.^4.*(wcjArray.^2+56.8*(10^6)) )./((wcjArray.^2+6
.3*(10^6)).*(wcjArray.^2+0.38*(10^9)).*(wcjArray.^6+9.58*(1
0^26)));
```

```
end
end
```

**********windowingOperator .m

```
function windowedFramedSn_std =
windowingOperator(framedZeroPaddedSn_std,hammingWindow25)
%WINDOWINGOPERATOR Summary of this function goes here
%window a given framedZeroPadded sn_std signal
%framedZeroPaddedSn_std: it has (length of window) x
(number of frames) size
%   Detailed explanation goes here

numOfFrames = size(framedZeroPaddedSn_std,2); %e.g. 99
windowedFramedSn_std = framedZeroPaddedSn_std;

for frameOrder = 1:numOfFrames
    windowedFramedSn_std(:,frameOrder) =
framedZeroPaddedSn_std(:,frameOrder).*hammingWindow25;
end


end
```

********** YLabelGenerator.m

```
function Y_labels_allWords =
YLabelGenerator(numOfDataPoints_cat,numOfDataPoints_dog,num
OfDataPoints_three,numOfDataPoints_tree)
%YLABELGENERATOR Summary of this function goes here
%Generate label vector
(numOfCat+numOfDog+numOfThree+numOfTree)x1 size
%   Detailed explanation goes here

%1:cat, 2:dog, 3:three, 4:tree
labels_cat = ones(numOfDataPoints_cat,1);
labels_dog = ones(numOfDataPoints_dog,1).*2;
labels_three = ones(numOfDataPoints_three,1).*3;
labels_tree = ones(numOfDataPoints_tree,1).*4;
```

```
Y_labels_allWords =
[labels_cat;labels_dog;labels_three;labels_tree];


end
```