Homework 5 EE519

Name: Samwoo Seong
USC ID: 7953-6137-66

Pr1. (a)

Soll)

O = V V U U U V V V U U

argmax(P(V)) = state3 $\triangleq S_3$

argmax(P(U)) = state1 or state 5

$\triangleq S_1$ or S_5

⇒ one possible most probable state sequence

$Q^* = S_3 \cdot S_3\ S_1 S_1 S_1\ S_3 S_3 S_3\ S_1 S_1$

where argmax(P(U)) = S_1 is chosen.

(b) $P^* = P(O, Q^* | \lambda)$

$= P(O) \cdot P(Q^*)$

$= P(V,V,U,U,U,V,V,V,U,U)$

$\cdot P(S_3, S_3, S_1, S_1, S_1, S_3, S_3, S_3, S_1, S_1)$

markov property

$= (0.8)^{10} \cdot P[S_3] \cdot P[S_3|S_3]$

$\cdot P[S_1|S_3] \cdot P[S_1|S_1]$

$P[S_1|S_1]\ P[S_3|S_1]\ P[S_3|S_3]$

$P[S_3|S_3]\ P[S_1|S_3]\ P[S_1|S_1]$

$= (0.8)^{10}\ \pi_3 \cdot a_{33} \cdot a_{13} \cdot a_{11}$

$\cdot a_{11} \cdot a_{31} \cdot a_{33} \cdot a_{33}$

$a_{13} \cdot a_{11}$

$= (0.8)^{10} \cdot (0.2)^{10} = 1.0995e-8$

$= 1.0995 \times 10^{-8}$

(c) $Q_1 = S_1 S_1 S_1 S_1\ S_1 S_1 S_1 S_1 S_1 S_1$

$P(O, Q_1)$

$= P(O)\, P(Q_1)$

$= (0.2)^5 (0.8)^5 \cdot \pi_1 \cdot a_{11}^9$

$= (0.2)^5 (0.8)^5 \cdot 0.2 \cdot (0.2)^9$

$= 1.0737e-11$

$= 1.0737 \times 10^{-11}$

& $P(O) = (0.2)^5 (0.8)^5 = 1.0486e-4$

$= 1.0486 \times 10^{-4}$

(d)

Pr2.

speech recognition and processing for multimedia

① phonemes

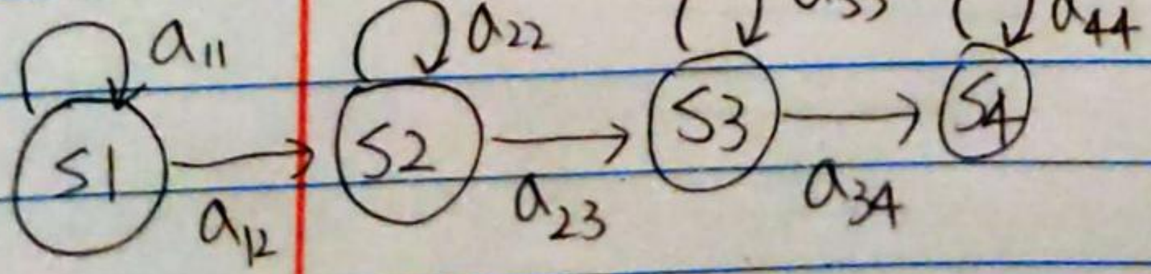
/S/ /P/ /IY/ /CH/ /R/ /EH/ /K/ /AH/
/G/ /N/ /IH/ /SH/ /AH/ /N/ /AH/ /N/
/D/ /P/ /R/ /AA/ /S/ /EH/ /S/ /IH/
/NG/ /F/ /AO/ /R/ /M/ /AH/ /L/
/T/ /IY/ /M/ /IY/ /D/ /IY/ /AH/

② triphones

/(sil) S (P)/ /(S) P (IY)/ /(P) IY (CH)/
/(IY) CH (R)/ /(CH) R (EH)/ /(R) EH (K)/
/(EH) K (AH)/ /(K) AH (G)/ /(AH) G (N)/
/(G) N (IH)/ /(N) IH (SH)/ /(IH) SH (AH)/
/(SH) AH (N)/ /(AH) N (AH)/ /(N) AH (N)/
/(AH) N (D)/ /(N) D (P)/ /(D) P (R)/
/(P) R (AA)/ /(R) AA (S)/ /(AA) S (EH)/
/(S) EH (S)/ /(EH) S (IH)/ /(S) IH (NG)/
/(IH) NG (F)/ /(NG) F (AO)/
/(F) AO (R)/ /(AO) R (M)/
/(R) M (AH)/ /(M) AH (L)/
/(AH) L (T)/ /(L) T (IY)/
/(T) IY (M)/ /(IY) M (IY)/ /(M) IY (D)/
/(IY) D (IY)/ /(D) IY (AH)/ /(IY) AH (sil)/

Pr3.

(a) For 4-states HMM



⇒ property of Markove model

$$\sum_{i=1}^{4} a_{ji} = 1$$

$$\Rightarrow a_{j1} + a_{j2} + a_{j3} + a_{j4} = 1$$

⇒ Due to the given topology,

$$a_{13} = a_{14} = a_{21} = a_{24} = a_{31} = a_{32} = a_{41} = a_{42} = a_{43} = 0$$

⇒ Therefore, we can initialize a transition matrix as follows

$$A_4 = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

where $a_{11} + a_{12} = 1$

$a_{22} + a_{23} = 1$

$a_{33} + a_{34} = 1$

$a_{44} = 1$

we can make all values of two term in each row equal except for the fourth row.

$$\Rightarrow A_4 = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly we can initialize a transition matrix A_6 as follows for 6 states HMM

$$A_6 = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & a_{66} \end{bmatrix}$$

where $a_{11} + a_{12} = 1$

$a_{22} + a_{23} = 1$

$a_{33} + a_{34} = 1$

$a_{44} + a_{45} = 1$

$a_{55} + a_{56} = 1$

$a_{66} = 1$

Furthermore,

$\Rightarrow a_{11} = a_{12} = a_{22} = a_{23} = a_{33} = a_{34}$

$= a_{44} = a_{45} = a_{55} = a_{56} = 0{,}5$

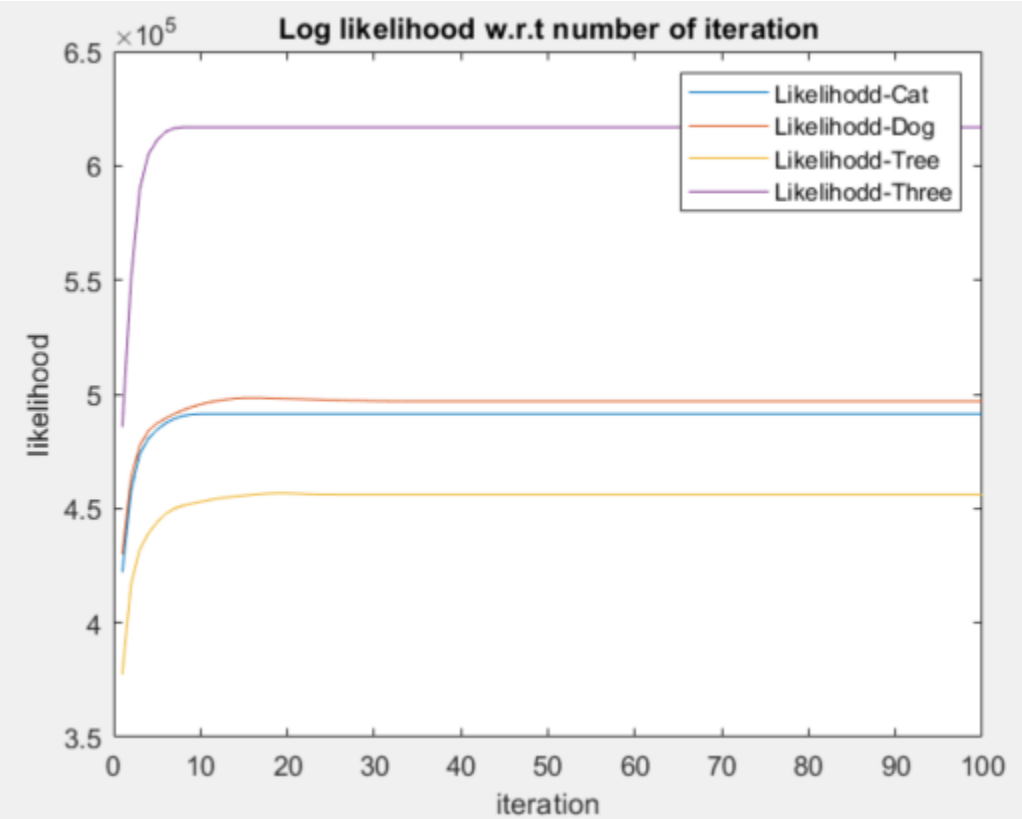
Samwoo Seong
7953-6137-66
samwoose@usc.edu

EE519 Homework #5

Part 1: Initialization
(b)
I have 1733 feature vectors from the original recordings for cat.
I have chosen only feature vectors that have 99 frames (Yet, in real world, I should be able to handle recordings with more and less frames than 99 frames). Therefore, I end up with 1515 data samples. For train set, I partition into 1060 which is 70% of 1515 samples.
I divide 99 frames into 5 parts for 5 states.
Thus, I have 20 frames for state 1, 2, 3, and 4, and 19 frames for state 5.
Finally, I have 21200 (=1060x20) samples for GMM of state 1, 2, 3, and 4, and 20140(=1060x19) for GMM of state 5.

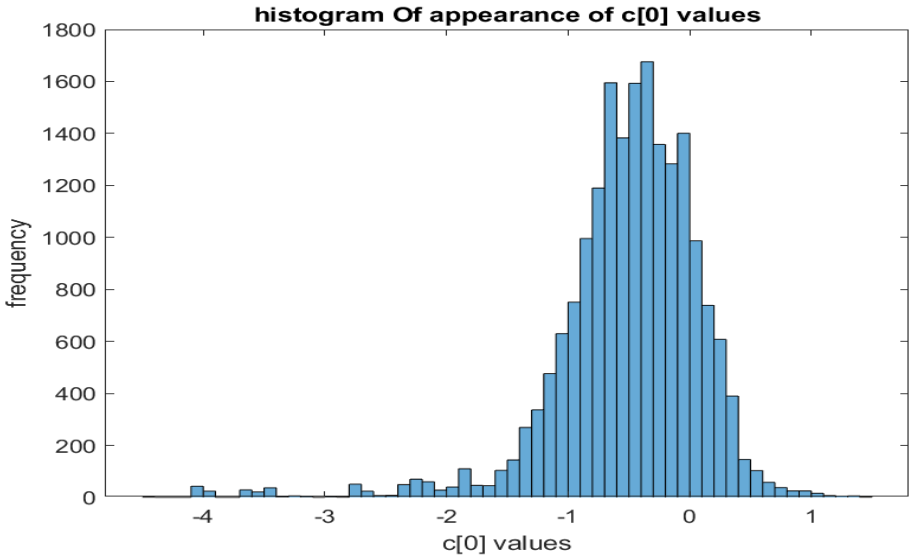
Part 2: Training
(a)

Maximum iteration = 100.



(b)

(1) Histogram for state 2 and 1st feature.

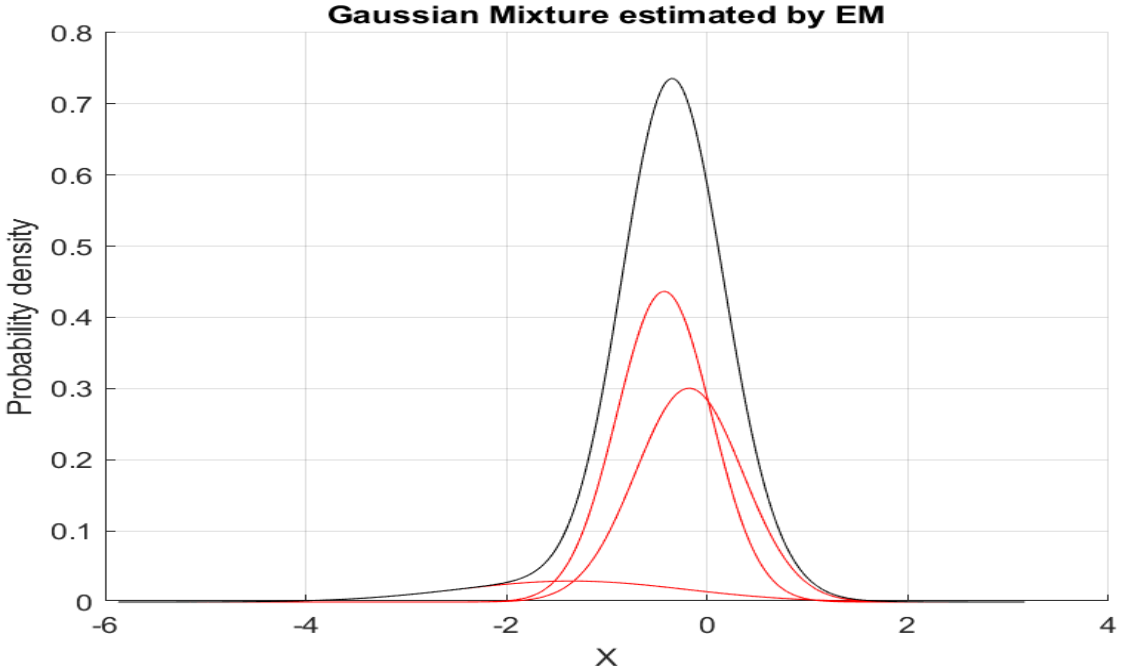


(2) underlying sample distribution for state 2 and first feature for cat.

Legend:

Each red curve: each mixture model.

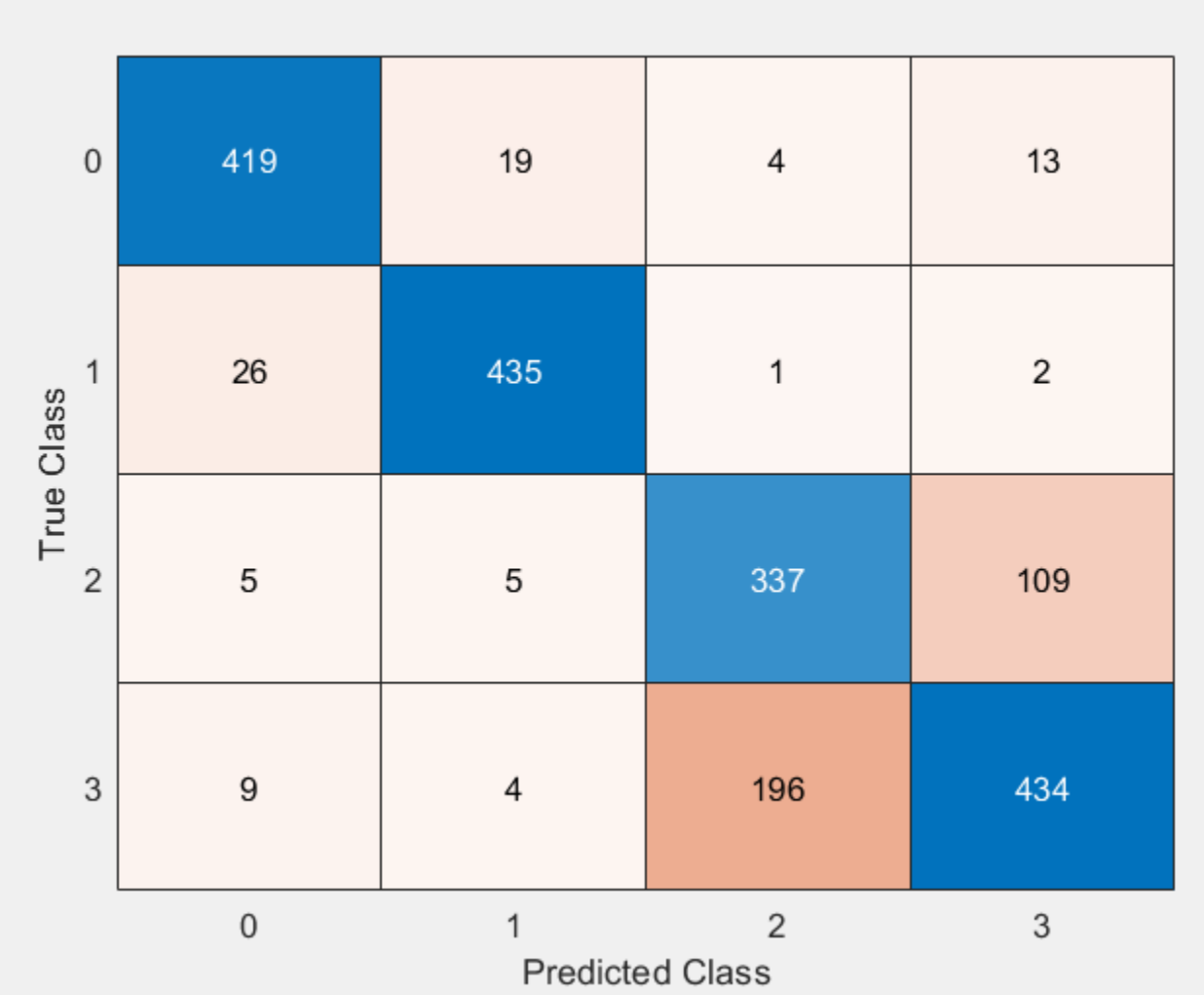
Black curve: mixed mixture model of 3 mixture models.



⇨ Yes, the probability distribution has a similar shape to the underlying sample distribution as visualized by the histogram.

Part 3: Evaluation
Accuracy of my system: 0.8053 (i.e., 80.53% of accuracy)

Confusion matrix
Note: 0: cat, 1: dog, 2: tree, 3: three



F1 score

	Cat	Dog	Tree	Three
F1 score	0.9168	0.9385	0.6781	0.7227

As we can observe in results of confusion matrix and F1 score, this system predicts well words like dog and cat. However, it gets confused (i.e. less accurate) when it predicts words between tree, and three.

Appendix

**********HW5_Pr3.m

```
%Load PLP data
load('plps_hw4.mat')

%Add path for a toolbox
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\HMM')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\KPMstats')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\KPMtools')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\netlab3.3')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\Plot_GM')



%% Partioning data into training(70%) set and test(30%) set
digit_cat =1;
digit_dog =2;
digit_tree = 3;
digit_three = 4;

totalData = plp;
totalData_cat = [totalData{1,digit_cat}];
totalData_dog = [totalData{1,digit_dog}];
totalData_tree = [totalData{1,digit_tree}];
totalData_three = [totalData{1,digit_three}];

ratioOfTrainSet = 0.7;
ratioOfTestSet = 0.3;

[train_cat, test_cat] =
dataSetRandomlyDivider(totalData_cat,ratioOfTrainSet,ratioO
fTestSet);
[train_dog, test_dog] =
dataSetRandomlyDivider(totalData_dog,ratioOfTrainSet,ratioO
fTestSet);
```

```
[train_tree, test_tree] =
dataSetRandomlyDivider(totalData_tree,ratioOfTrainSet,ratio
OfTestSet);
[train_three, test_three] =
dataSetRandomlyDivider(totalData_three,ratioOfTrainSet,rati
oOfTestSet);

%% Part 1: Initialization
%(b)
[dataForS1_cat, dataForS2_cat, dataForS3_cat,
dataForS4_cat, dataForS5_cat ] =
dataMakerForGMM_S5Version(train_cat);
[dataForS1_dog, dataForS2_dog, dataForS3_dog,
dataForS4_dog, dataForS5_dog ] =
dataMakerForGMM_S5Version(train_dog);
[dataForS1_tree, dataForS2_tree, dataForS3_tree,
dataForS4_tree, dataForS5_tree ] =
dataMakerForGMM_S5Version(train_tree);
[dataForS1_three, dataForS2_three, dataForS3_three,
dataForS4_three, dataForS5_three ] =
dataMakerForGMM_S5Version(train_three);

%get mean and std
numOfMixtures = 3;
numOfFeatures = 13;
numOfStates = 5;

%cat
[mu4S1_cat, Sigma4S1_cat, weights4S1_cat] =
mixgauss_init(numOfMixtures, dataForS1_cat, 'diag');
[mu4S2_cat, Sigma4S2_cat, weights4S2_cat] =
mixgauss_init(numOfMixtures, dataForS2_cat, 'diag');
[mu4S3_cat, Sigma4S3_cat, weights4S3_cat] =
mixgauss_init(numOfMixtures, dataForS3_cat, 'diag');
[mu4S4_cat, Sigma4S4_cat, weights4S4_cat] =
mixgauss_init(numOfMixtures, dataForS4_cat, 'diag');
[mu4S5_cat, Sigma4S5_cat, weights4S5_cat] =
mixgauss_init(numOfMixtures, dataForS5_cat, 'diag');


mu4AllStates_cat = muCombinder(mu4S1_cat, mu4S2_cat,
mu4S3_cat, mu4S4_cat, mu4S5_cat, numOfFeatures,
numOfStates, numOfMixtures);
```

```
Sigma4AllStates_cat =
sigmaCombinder(Sigma4S1_cat,Sigma4S2_cat,Sigma4S3_cat,Sigma
4S4_cat,Sigma4S5_cat,numOfFeatures, numOfStates,
numOfMixtures);

%dog
[mu4S1_dog, Sigma4S1_dog, weights4S1_dog] =
mixgauss_init(numOfMixtures, dataForS1_dog, 'diag');
[mu4S2_dog, Sigma4S2_dog, weights4S2_dog] =
mixgauss_init(numOfMixtures, dataForS2_dog, 'diag');
[mu4S3_dog, Sigma4S3_dog, weights4S3_dog] =
mixgauss_init(numOfMixtures, dataForS3_dog, 'diag');
[mu4S4_dog, Sigma4S4_dog, weights4S4_dog] =
mixgauss_init(numOfMixtures, dataForS4_dog, 'diag');
[mu4S5_dog, Sigma4S5_dog, weights4S5_dog] =
mixgauss_init(numOfMixtures, dataForS5_dog, 'diag');

mu4AllStates_dog = muCombinder(mu4S1_dog, mu4S2_dog,
mu4S3_dog, mu4S4_dog, mu4S5_dog, numOfFeatures,
numOfStates, numOfMixtures);
Sigma4AllStates_dog =
sigmaCombinder(Sigma4S1_dog,Sigma4S2_dog,Sigma4S3_dog,Sigma
4S4_dog,Sigma4S5_dog,numOfFeatures, numOfStates,
numOfMixtures);

%tree
[mu4S1_tree, Sigma4S1_tree, weights4S1_tree] =
mixgauss_init(numOfMixtures, dataForS1_tree, 'diag');
[mu4S2_tree, Sigma4S2_tree, weights4S2_tree] =
mixgauss_init(numOfMixtures, dataForS2_tree, 'diag');
[mu4S3_tree, Sigma4S3_tree, weights4S3_tree] =
mixgauss_init(numOfMixtures, dataForS3_tree, 'diag');
[mu4S4_tree, Sigma4S4_tree, weights4S4_tree] =
mixgauss_init(numOfMixtures, dataForS4_tree, 'diag');
[mu4S5_tree, Sigma4S5_tree, weights4S5_tree] =
mixgauss_init(numOfMixtures, dataForS5_tree, 'diag');

mu4AllStates_tree = muCombinder(mu4S1_tree, mu4S2_tree,
mu4S3_tree, mu4S4_tree, mu4S5_tree, numOfFeatures,
numOfStates, numOfMixtures);
Sigma4AllStates_tree =
sigmaCombinder(Sigma4S1_tree,Sigma4S2_tree,Sigma4S3_tree,Si
gma4S4_tree,Sigma4S5_tree,numOfFeatures, numOfStates,
numOfMixtures);
```

```
%three
[mu4S1_three, Sigma4S1_three, weights4S1_three] =
mixgauss_init(numOfMixtures, dataForS1_three, 'diag');
[mu4S2_three, Sigma4S2_three, weights4S2_three] =
mixgauss_init(numOfMixtures, dataForS2_three, 'diag');
[mu4S3_three, Sigma4S3_three, weights4S3_three] =
mixgauss_init(numOfMixtures, dataForS3_three, 'diag');
[mu4S4_three, Sigma4S4_three, weights4S4_three] =
mixgauss_init(numOfMixtures, dataForS4_three, 'diag');
[mu4S5_three, Sigma4S5_three, weights4S5_three] =
mixgauss_init(numOfMixtures, dataForS5_three, 'diag');

mu4AllStates_three = muCombinder(mu4S1_three, mu4S2_three,
mu4S3_three, mu4S4_three, mu4S5_three, numOfFeatures,
numOfStates, numOfMixtures);
Sigma4AllStates_three =
sigmaCombinder(Sigma4S1_three,Sigma4S2_three,Sigma4S3_three
,Sigma4S4_three,Sigma4S5_three,numOfFeatures, numOfStates,
numOfMixtures);

%% Part2: Training
%(a)
%initialize prior and transition matrix
prior = [1 ;0 ;0 ;0 ;0] ;%not sure about this
transitMatrix = [0.5 0.5 0 0 0; 0 0.5 0.5 0 0; 0 0 0.5 0.5
0; 0 0 0 0.5 0.5; 0 0 0 0 1];
mixmat_cat = [weights4S1_cat' ;weights4S2_cat' ;
weights4S3_cat' ; weights4S4_cat' ; weights4S5_cat'];
mixmat_dog = [weights4S1_dog' ; weights4S2_dog' ;
weights4S3_dog' ; weights4S4_dog'; weights4S5_dog'];
mixmat_tree = [weights4S1_tree'; weights4S2_tree';
weights4S3_tree'; weights4S4_tree'; weights4S5_tree'];
mixmat_three = [weights4S1_three'; weights4S2_three';
weights4S3_three'; weights4S4_three'; weights4S5_three'];
numOfMaxIteration = 100;
%cat
%[LL_cat, prior_cat, transmat_cat, mu_cat, Sigma_cat,
mixmat_cat] = mhmm_em(train_cat, prior, transitMatrix,
mu4AllStates_cat, Sigma4AllStates_cat,
ones(numOfStates,numOfMixtures));
[LL_cat, trained_prior_cat, trained_transmat_cat,
trained_mu_cat, trained_Sigma_cat, trained_mixmat_cat] =
```

```
mhmm_em(train_cat, prior, transitMatrix, mu4AllStates_cat,
Sigma4AllStates_cat, mixmat_cat,'max_iter',
numOfMaxIteration,'cov_type', 'diag');

%dog
[LL_dog, trained_prior_dog, trained_transmat_dog,
trained_mu_dog, trained_Sigma_dog, trained_mixmat_dog] =
mhmm_em(train_dog, prior, transitMatrix, mu4AllStates_dog,
Sigma4AllStates_dog, mixmat_dog,'max_iter',
numOfMaxIteration,'cov_type', 'diag');

%tree
[LL_tree, trained_prior_tree, trained_transmat_tree,
trained_mu_tree, trained_Sigma_tree, trained_mixmat_tree] =
mhmm_em(train_tree, prior, transitMatrix,
mu4AllStates_tree, Sigma4AllStates_tree,
mixmat_tree,'max_iter', numOfMaxIteration,'cov_type',
'diag');

%three
[LL_three, trained_prior_three, trained_transmat_three,
trained_mu_three, trained_Sigma_three,
trained_mixmat_three] = mhmm_em(train_three, prior,
transitMatrix, mu4AllStates_three, Sigma4AllStates_three,
mixmat_three,'max_iter', numOfMaxIteration,'cov_type',
'diag');

iterationArr = [1:numOfMaxIteration];

%make same length LL_cat, LL_tree, LL_tree
convergedVal_cat = LL_cat(1,11);
convergedVal_dog = LL_dog(1,34);
convergedVal_tree = LL_tree(1,28);
convergedVal_three = LL_three(1,9);

sameLengLL_cat =
ones(1,numOfMaxIteration)*convergedVal_cat;
sameLengLL_dog =
ones(1,numOfMaxIteration)*convergedVal_dog;
sameLengLL_tree =
ones(1,numOfMaxIteration)*convergedVal_tree;
sameLengLL_three =
ones(1,numOfMaxIteration)*convergedVal_three;
```

```
sameLengLL_cat(1,1:11) = LL_cat;
sameLengLL_dog(1,1:34) = LL_dog;
sameLengLL_tree(1,1:28) = LL_tree;
sameLengLL_three(1,1:9) = LL_three;



%plot
figure(1)
plot(iterationArr,sameLengLL_cat)
title('Log likelihood w.r.t number of iteration')

hold on
plot(iterationArr,sameLengLL_dog)

plot(iterationArr,sameLengLL_tree)

plot(iterationArr,sameLengLL_three)

legend('Likelihodd-Cat','Likelihodd-Dog','Likelihodd-
Tree','Likelihodd-Three')
xlabel('iteration')
ylabel('likelihood')

hold off


%(b)
%find best path using viterbi algorithm
% [obslik_cat_r1,~] =
mixgauss_prob(train_cat(:,:,1),trained_mu_cat,trained_Sigma
_cat,trained_mixmat_cat);
% path_cat_r1 =
viterbi_path(trained_prior_cat,trained_transmat_cat,obslik_
cat_r1);

feature1ForS2NWholeTrainData =
feature1ForS2Extractor(train_cat,trained_mu_cat,trained_Sig
ma_cat,trained_mixmat_cat,trained_prior_cat,trained_transma
t_cat);

%Plot Histogram.
figure(2)
h = histogram(feature1ForS2NWholeTrainData);
```

```
title('histogram Of appearance of c[0] values')
xlabel('c[0] values')
ylabel('frequency')
%plot 1D GMM for state 2, for cat
trained_mu_cat_S2 = trained_mu_cat(1,2,:);
trained_Sigma_cat_S2 = trained_Sigma_cat(1,1,2,:);
figure(3)
Plot_GM(weights4S2_cat',trained_mu_cat_S2,trained_Sigma_cat
_S2);

%% Part3: Evaluation
%Construct total test set label
%0:cat, 1:dog, 2:tree, 3:three
y_test = zeros(1,455+464+643+456);
y_test(1,456:919) = ones(1,464);
y_test(1,920:1375) = 2*ones(1,456);
y_test(1,1376:2018) = 3*ones(1,643);

y_predicted =
HMM4Predictor(test_cat,test_dog,test_tree,test_three,traine
d_prior_cat,trained_transmat_cat,trained_mu_cat,trained_Sig
ma_cat,trained_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three );
accuracy_HMM4 = accuracyCalculator(y_predicted,y_test);

figure(4)
cm = confusionchart(y_test,y_predicted);

confusionMatrix = cm.NormalizedValues;

recall = zeros(1,4);
precision = zeros(1,4);
F1Score = zeros(1,4);


for orderOfClass = 1:4
    recall(1,orderOfClass) =
confusionMatrix(orderOfClass,orderOfClass)/sum(confusionMat
rix(orderOfClass,:));
```

```
    precision(1,orderOfClass) =
confusionMatrix(orderOfClass,orderOfClass)/sum(confusionMat
rix(:,orderOfClass));
    F1Score(1,orderOfClass) =
2*(recall(1,orderOfClass)*precision(1,orderOfClass))/(recal
l(1,orderOfClass)+precision(1,orderOfClass));
End
```

**********accuracyCalculator.m

```
function accuracy_HMM4 =
accuracyCalculator(y_predicted,y_test)
%ACCURACYCALCULATOR Summary of this function goes here
%   Detailed explanation goes here


lengthOfLabels = size(y_predicted,2);

numOfCorrectlyClassified = 0;

for index = 1:lengthOfLabels
    if(y_predicted(1,index) == y_test(1,index))
        numOfCorrectlyClassified = numOfCorrectlyClassified
+ 1;
    end
end

accuracy_HMM4 = numOfCorrectlyClassified/lengthOfLabels;


end
```

**********dataMakerForGMM_S5Version.m

```
function [dataForS1, dataForS2, dataForS3, dataForS4,
dataForS5 ] = dataMakerForGMM_S5Version(train)
%DATAMAKERFORGMM_S5VERSION Summary of this function goes
here
```

```
%Obtain data sets with a proper size using the given data
set for GMM
%initialization
%   Detailed explanation goes here

numOfFrame4S1 = 20;
% numOfFrame4S2 = 20;
% numOfFrame4S3 = 20;
% numOfFrame4S4 = 20;
% numOfFrame4S5 = 19;

% tensorDepth = size(train,3); %e.g 1060 for cat, 1083 for
dog, 1501 for three, 1065 for tree
numOfTotalFrames = size(train,2); % e.g. 99 frames

% numOfSamples4S1 = numOfFrame4S1 * tensorDepth;
% numOfSamples4S2 = numOfFrame4S2 * tensorDepth;
% numOfSamples4S3 = numOfFrame4S3 * tensorDepth;
% numOfSamples4S4 = numOfFrame4S4 * tensorDepth;
% numOfSamples4S5 = numOfFrame4S5 * tensorDepth;

% dataForS1 = zeros(numOfFeature,numOfSamples4S1);
% dataForS2 = zeros(numOfFeature,numOfSamples4S2);
% dataForS3 = zeros(numOfFeature,numOfSamples4S3);
% dataForS4 = zeros(numOfFeature,numOfSamples4S4);
% dataForS5 = zeros(numOfFeature,numOfSamples4S5);

%First partitioning data
dataForS1_1st = train(:,1:numOfFrame4S1,:);
dataForS2_1st =
train(:,(numOfFrame4S1+1):(numOfFrame4S1*2),:);
dataForS3_1st =
train(:,((numOfFrame4S1*2)+1):(numOfFrame4S1*3),:);
dataForS4_1st =
train(:,((numOfFrame4S1*3)+1):(numOfFrame4S1*4),:);
dataForS5_1st =
train(:,((numOfFrame4S1*4)+1):numOfTotalFrames,:);



%Second partitoning data
dataForS1 = dataSetReshaperForGMM(dataForS1_1st);
dataForS2 = dataSetReshaperForGMM(dataForS2_1st);
dataForS3 = dataSetReshaperForGMM(dataForS3_1st);
```

```
dataForS4 = dataSetReshaperForGMM(dataForS4_1st);
dataForS5 = dataSetReshaperForGMM(dataForS5_1st);
end
```

**********dataSetRandomlyDivider.m

```
function [train, test] =
dataSetRandomlyDivider(totalData,ratioOfTrainSet,ratioOfTes
tSet)
%DATASETRANDOMLYDEVIDER Summary of this function goes here
%Divide total data set into train data set and test data
set by given ratio
%of data sets
%   Detailed explanation goes here

%Get only features with 99 frames
reducedTotalData =
frameLengthFilter(totalData); %reducedTotaldata has
13x99xreducedNumOfSamples dimension

reducedNumOfSamples = size(reducedTotalData,3); %e.g. 1515
for cat, 1547 for dog, 1521 for tree, 2144 for three

%Draw indeces at random
[trainIndex,~,testIndex ]= dividerand(reducedNumOfSamples,
ratioOfTrainSet,0,ratioOfTestSet);

train = reducedTotalData(:,:,trainIndex);
test = reducedTotalData(:,:,testIndex);


end
```

**********dataSetReshaperForGMM.m

```
function dataForS = dataSetReshaperForGMM(dataForS_1st)
%DATASETRESHAPERFORGMM Summary of this function goes here
%   Detailed explanation goes here
tensorDepth = size(dataForS_1st,3);%e.g 1060 for dog
```

```
numOfFrames = size(dataForS_1st,2);%e.g 20 frames for the
first state
numOfFeatures = size(dataForS_1st,1);%e.g. 13 dimension

numOfSamples = tensorDepth*numOfFrames; %e.g. 1060*20

dataForS = zeros(numOfFeatures,numOfSamples);

sampleIndex = 1;
for widthIndex = 1:numOfFrames
    for depthIndex =1:tensorDepth
        dataForS(:,sampleIndex) =
dataForS_1st(:,widthIndex,depthIndex);
        sampleIndex = sampleIndex + 1;
    end
end


end
```

```
************feature1ForS2Extractor.m
function feature1ForS2NWholeTrainData =
feature1ForS2Extractor(train,trained_mu,trained_Sigma,train
ed_mixmat,trained_prior,trained_transmat)
%FEATURE1FORS2EXTRACTOR Summary of this function goes here
%   Detailed explanation goes here

numOfData = size(train,3); %e.g. 1060
helperIndex = 1;
for dataIndex = 1:numOfData
    feature1ForS2ForOneFile =
feature1ForS2ForOneFileExtractor(train(:,:,dataIndex),train
ed_mu,trained_Sigma,trained_mixmat,trained_prior,trained_tr
ansmat);
    sizeOfFeature1ForS2ForOneFile =
size(feature1ForS2ForOneFile,2);
```

```
    for index2 = 1:sizeOfFeature1ForS2ForOneFile
        feature1ForS2NWholeTrainData(1,helperIndex) =
feature1ForS2ForOneFile(1,index2);
        helperIndex = helperIndex + 1;
    end
    disp(dataIndex)
end



end
```

```
************feature1ForS2ForOneFileExtractor.m
function feature1ForS2ForOneFile =
feature1ForS2ForOneFileExtractor(trainOneRecord,trained_mu,
trained_Sigma,trained_mixmat,trained_prior,trained_transmat
)
%FEATURE1FORS2FORONEFILEEXTRACTOR Summary of this function
goes here
%   Detailed explanation goes here

addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\HMM')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\KPMstats')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\KPMtools')
addpath('C:\Users\tjdtk\Desktop1\EE519\Homework5\HMM\HMMall
\netlab3.3')

%
[obslik_r1,~] =
mixgauss_prob(trainOneRecord,trained_mu,trained_Sigma,train
ed_mixmat);
path_r1 =
viterbi_path(trained_prior,trained_transmat,obslik_r1);

numOfFrame = size(path_r1,2); %e.g. 99
%Compute number of appearances of each state
```

```
numOfS1 = 0;
numOfS2 = 0;
numOfS3 = 0;
numOfS4 = 0;
numOfS5 = 0;
for frameIndex = 1:numOfFrame
    if(path_r1(1,frameIndex)==1)
        numOfS1 = numOfS1 + 1;
    elseif(path_r1(1,frameIndex)==2)
        numOfS2 = numOfS2 + 1;
    elseif(path_r1(1,frameIndex)==3)
        numOfS3 = numOfS3 + 1;
    elseif(path_r1(1,frameIndex)==4)
        numOfS4 = numOfS4 + 1;
    elseif(path_r1(1,frameIndex)==5)
        numOfS5 = numOfS5 + 1;
    end
end
lastIndexForS2 = numOfS1+numOfS2;
indexForS2 = [(numOfS1+1):lastIndexForS2];

feature1ForS2ForOneFile = trainOneRecord(1,indexForS2);



end
```

```
************frameLengthFilter.m
function reducedTotalData = frameLengthFilter(totalData)
%FRAMELENGTHFILTER Summary of this function goes here
%Obtain only data samples with 99 frames
%and reconstruct data set
%   Detailed explanation goes here

oneSample = totalData{1,1};  %13x99 size

tensorHeight = size(oneSample,1); %e.g. 13
tensorWidth = size(oneSample,2); %e.g. 99
tensorDepth = size(totalData,2); %e.g. 1733
```

```
numOfReducedSamples = 0;
for orderOfDepth = 1:tensorDepth
    currentSample = totalData{1,orderOfDepth};
    widthOfCurrentSample = size(currentSample,2);
    if(widthOfCurrentSample == tensorWidth)
        numOfReducedSamples = numOfReducedSamples + 1;
    end
end

reducedTensorDepth = numOfReducedSamples;

reducedTotalData =
zeros(tensorHeight,tensorWidth,reducedTensorDepth);

indexOfReducedDepth = 1;
for orderOfDepth = 1:tensorDepth
    currentSample = totalData{1,orderOfDepth};
    widthOfCurrentSample = size(currentSample,2);
    if(widthOfCurrentSample == tensorWidth)
        reducedTotalData(:,:,indexOfReducedDepth) =
totalData{1,orderOfDepth};
        indexOfReducedDepth = indexOfReducedDepth+1;
    end
end


end
```

```
************HMM4Predictor.m
function y_predicted =
HMM4Predictor(test_cat,test_dog,test_tree,test_three,traine
d_prior_cat,trained_transmat_cat,trained_mu_cat,trained_Sig
ma_cat,trained_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
```

```
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three )
%HMM4PREDICTOR Summary of this function goes here
%   Detailed explanation goes here

numOfSamples_cat = size(test_cat,3);
numOfSamples_dog = size(test_dog,3);
numOfSamples_tree = size(test_tree,3);
numOfSamples_three = size(test_three,3);

y_predicted =
zeros(1,numOfSamples_cat+numOfSamples_dog+numOfSamples_tree
+numOfSamples_three);

overallIndex = 1;
%cat
for index_cat = 1:numOfSamples_cat
    crrRecording_cat = test_cat(:,:,index_cat);
    crrPredicedLabel =
HMM4PredictorForOneRecord(crrRecording_cat,trained_prior_ca
t,trained_transmat_cat,trained_mu_cat,trained_Sigma_cat,tra
ined_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three );
    y_predicted(1,overallIndex) = crrPredicedLabel;
    overallIndex = overallIndex+1;
end

%dog
for index_dog = 1:numOfSamples_dog
    crrRecording_dog = test_dog(:,:,index_dog);
    crrPredicedLabel =
HMM4PredictorForOneRecord(crrRecording_dog,trained_prior_ca
t,trained_transmat_cat,trained_mu_cat,trained_Sigma_cat,tra
ined_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
```

```
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three );
    y_predicted(1,overallIndex) = crrPredicedLabel;
    overallIndex = overallIndex+1;
end

%tree
for index_tree = 1:numOfSamples_tree
    crrRecording_tree = test_tree(:,:,index_tree);
    crrPredicedLabel =
HMM4PredictorForOneRecord(crrRecording_tree,trained_prior_c
at,trained_transmat_cat,trained_mu_cat,trained_Sigma_cat,tr
ained_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three );
    y_predicted(1,overallIndex) = crrPredicedLabel;
    overallIndex = overallIndex+1;
end

%three
for index_three = 1:numOfSamples_three
    crrRecording_three = test_three(:,:,index_three);
    crrPredicedLabel =
HMM4PredictorForOneRecord(crrRecording_three,trained_prior_
cat,trained_transmat_cat,trained_mu_cat,trained_Sigma_cat,t
rained_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three );
    y_predicted(1,overallIndex) = crrPredicedLabel;
    overallIndex = overallIndex+1;
end
```

```
***********HMM4PredictorForOneRecord.m
function crrPredicedLabel =
HMM4PredictorForOneRecord(crrRecording,trained_prior_cat,tr
ained_transmat_cat,trained_mu_cat,trained_Sigma_cat,trained
_mixmat_cat,
trained_prior_dog,trained_transmat_dog,trained_mu_dog,train
ed_Sigma_dog,trained_mixmat_dog ,
trained_prior_tree,trained_transmat_tree,trained_mu_tree,tr
ained_Sigma_tree,trained_mixmat_tree ,
trained_prior_three,trained_transmat_three,trained_mu_three
,trained_Sigma_three,trained_mixmat_three )
%HMM4PREDICTORFORONERECORD Summary of this function goes
here
%   Detailed explanation goes here

[loglik_byHMM1ForCat,~] =
mhmm_logprob(crrRecording,trained_prior_cat ,
trained_transmat_cat , trained_mu_cat , trained_Sigma_cat,
trained_mixmat_cat );
[loglik_byHMM2ForDog,~] =
mhmm_logprob(crrRecording,trained_prior_dog ,
trained_transmat_dog , trained_mu_dog , trained_Sigma_dog,
trained_mixmat_dog );
[loglik_byHMM3ForTree,~] =
mhmm_logprob(crrRecording,trained_prior_tree ,
trained_transmat_tree , trained_mu_tree ,
trained_Sigma_tree, trained_mixmat_tree );
[loglik_byHMM4ForThree,~] =
mhmm_logprob(crrRecording,trained_prior_three ,
trained_transmat_three, trained_mu_three ,
trained_Sigma_three, trained_mixmat_three );

loglikStorage = [loglik_byHMM1ForCat, loglik_byHMM2ForDog,
loglik_byHMM3ForTree , loglik_byHMM4ForThree];


[~,location ] = max(loglikStorage);

crrPredicedLabel = location - 1;
end
```

```
************muCombinder.m
function mu4AllStates = muCombinder(mu4S1, mu4S2, mu4S3,
mu4S4, mu4S5, numOfFeatures, numOfStates, numOfMixtures)
%MUCOMBINDER Summary of this function goes here
%combine mu 1, 2, 3, 4, 5
%   Detailed explanation goes here

mu4AllStates =
zeros(numOfFeatures,numOfStates,numOfMixtures);


mu4AllStates(:,1,:) = mu4S1(:,:);
mu4AllStates(:,2,:) = mu4S2(:,:);
mu4AllStates(:,3,:) = mu4S3(:,:);
mu4AllStates(:,4,:) = mu4S4(:,:);
mu4AllStates(:,5,:) = mu4S5(:,:);

end
```

```
************sigmaCombinder.m
function Sigma4AllStates =
sigmaCombinder(Sigma4S1,Sigma4S2,Sigma4S3,Sigma4S4,Sigma4S5
,numOfFeatures, numOfStates, numOfMixtures)
%SIGMACOMBINDER Summary of this function goes here
%Combine Sigma 1,2,3,4,5
%   Detailed explanation goes here

Sigma4AllStates =
zeros(numOfFeatures,numOfFeatures,numOfStates,numOfMixtures
);

Sigma4AllStates(:,:,1,:) = Sigma4S1(:,:,:);
Sigma4AllStates(:,:,2,:) = Sigma4S2(:,:,:);
Sigma4AllStates(:,:,3,:) = Sigma4S3(:,:,:);
Sigma4AllStates(:,:,4,:) = Sigma4S4(:,:,:);
Sigma4AllStates(:,:,5,:) = Sigma4S5(:,:,:);
```

```
end
```