18-698 / 42-632
Neural Signal Processing
Problem Set 3 Solutions

1.)

ii.) Full Gaussian, class covariance: $\boldsymbol{x}|\boldsymbol{C_k} \sim \mathcal{N}(\boldsymbol{\mu_k}, \Sigma_k)$. We want to find the following:

$$(\pi_k^*, \boldsymbol{\mu_k^*}, \Sigma_k^*) = \arg\max_{\pi_k, \boldsymbol{\mu_k}, \Sigma_k} P\left(\left\{\boldsymbol{x_n}, C_{k_n}\right\} \middle| \pi_k, \boldsymbol{\mu_k}, \Sigma_k\right) \ subject\ to\ \sum_{k=1}^{K} \pi_k = 1$$

$$= \arg\max_{\pi_k, \boldsymbol{\mu_k}, \Sigma_k} \prod_{k=1}^{K} \prod_{n \in C_k} \pi_k \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \Sigma_k) \ subject\ to\ \sum_{k=1}^{K} \pi_k = 1$$

Instead, we can maximize the log-likelihood, utilizing the fact that $\ln x$ is a monotonically increasing function. Also, the constraint can be incorporated into the maximization via a Lagrangian term:

$$\arg\max_{\pi_k, \boldsymbol{\mu_k}, \Sigma_k, \lambda} \mathcal{L} = \arg\max_{\pi_k, \boldsymbol{\mu_k}, \Sigma_k, \lambda} \left\{ \ln\left[\prod_{k=1}^{K} \prod_{n \in C_k} \pi_k \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \Sigma_k)\right] - \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)\right\}$$

$$= \arg\max_{\pi_k, \boldsymbol{\mu_k}, \Sigma_k, \lambda} \left\{ \sum_{k=1}^{K} \sum_{n \in C_k} \left[\ln \pi_k - \frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \Sigma_k^{-1} (\boldsymbol{x_n} - \boldsymbol{\mu_k})\right] - \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)\right\}$$

We can find the global maximum by setting the partial derivatives of $\mathcal{L}$ to zero:

$$\frac{d\mathcal{L}}{d\pi_k} = \sum_{n \in C_k} \frac{1}{\pi_k} - \lambda = set\ 0 \Rightarrow \pi_k^* = \frac{N_k}{\lambda}$$

$$\frac{d\mathcal{L}}{d\lambda} = 1 - \sum_{k=1}^{K} \pi_k = set\ 0 \Rightarrow \sum_{k=1}^{K} \pi_k^* = \sum_{k=1}^{K} \frac{N_k}{\lambda} = \frac{N}{\lambda} = 1 \Rightarrow \lambda = N \Rightarrow \pi_k^* = \frac{N_k}{N}$$

To find $\frac{d\mathcal{L}}{d\boldsymbol{\mu_k}}$, we use these facts: $\frac{d}{d\boldsymbol{z}} A\boldsymbol{z} = A$ and $\frac{d}{d\boldsymbol{z}} \boldsymbol{z}^T A \boldsymbol{z} = \boldsymbol{z}^T(A^T + A)$.

$$\frac{d\mathcal{L}}{d\boldsymbol{\mu_k}} = \sum_{n \in C_k} (\boldsymbol{x_n^T} \Sigma_k^{-1} - \boldsymbol{\mu_k^T} \Sigma_k^{-1}) = set\ 0 \Rightarrow \sum_{n \in C_k} \left(\boldsymbol{x_n^T} - \boldsymbol{\mu_k^*}^T\right) = 0 \Rightarrow \boldsymbol{\mu_k^*} = \frac{1}{N_k} \sum_{n \in C_k} \boldsymbol{x_n}$$

To find Σ_k^*, we set $\frac{d\mathcal{L}}{d\Lambda_k} = 0$, where $\Lambda_k = \Sigma_k^{-1}$. We also use the fact: $|A^{-1}| = |A|^{-1}$.
First, these are the terms of $\mathcal{L}$ that depend on Σ_k, expressed in terms of Λ_k:

$$-\frac{1}{2} \sum_{k=1}^{K} \sum_{n \in C_k} [-\ln|\Lambda_k| + (\boldsymbol{x_n} - \boldsymbol{\mu_k})\Lambda_k(\boldsymbol{x_n} - \boldsymbol{\mu_k})]$$

To simplify $\frac{d\mathcal{L}}{d\Lambda_k}$, we use this fact: $\frac{d}{dA}\ln|A| = (A^{-1})^T$.

$$\frac{d\mathcal{L}}{d\Lambda_k} = -\frac{1}{2}\sum_{n\in C_k}\left[(\boldsymbol{x_n}-\boldsymbol{\mu_k})(\boldsymbol{x_n}-\boldsymbol{\mu_k})^T - (\Lambda_k^{-1})^T\right] = set\ 0$$

$$\Rightarrow {\Lambda_k^*}^{-1} = \Sigma_k^* = \frac{1}{N_k}\sum_{n\in C_k}(\boldsymbol{x_n}-\boldsymbol{\mu_k})(\boldsymbol{x_n}-\boldsymbol{\mu_k})^T = S_k$$

To summarize:

$$\pi_k^* = \frac{N_k}{N}, \boldsymbol{\mu_k^*} = \frac{1}{N_k}\sum_{n\in C_k}\boldsymbol{x_n}, \Sigma_k^* = \frac{1}{N_k}\sum_{n\in C_k}(\boldsymbol{x_n}-\boldsymbol{\mu_k})(\boldsymbol{x_n}-\boldsymbol{\mu_k})^T = S_k$$

iii.) Poisson: $x_i|C_k \sim Poisson(\lambda_{ki})$. We want to find the following:

$$(\pi_k^*, \lambda_{ki}^*) = \arg\max_{\pi_k,\lambda_{ki}} P\left(\{x_{ni}, C_{k_n}\}\middle|\pi_k, \lambda_{ki}\right)\ subject\ to\ \sum_{k=1}^{K}\pi_k = 1$$

$$= \arg\max_{\pi_k,\lambda_{ki}}\prod_{k=1}^{K}\prod_{n\in C_k}\prod_{i=1}^{D}\pi_k\frac{\lambda_{ki}^{x_{ni}}e^{-\lambda_{ki}}}{x_{ni}!}\ subject\ to\ \sum_{k=1}^{K}\pi_k = 1$$

Instead, we can maximize the log-likelihood, utilizing the fact that $\ln x$ is a monotonically increasing function. Also, the constraint can be incorporated into the maximization via a Lagrangian term:

$$\arg\max_{\pi_k,\lambda_{ki},\lambda}\mathcal{L} = \arg\max_{\pi_k,\lambda_{ki},\lambda}\left[\ln\left(\prod_{k=1}^{K}\prod_{n\in C_k}\prod_{i=1}^{D}\pi_k\frac{\lambda_{ki}^{x_{ni}}e^{-\lambda_{ki}}}{x_{ni}!}\right) - \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)\right]$$

$$= \arg\max_{\pi_k,\lambda_{ki},\lambda}\left\{\sum_{k=1}^{K}\sum_{n\in C_k}\sum_{i=1}^{D}\left[\ln\pi_k + x_{ni}\ln\lambda_{ki} - \lambda_{ki} - \ln(x_{ni}!)\right] - \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)\right\}$$

We can find the global maximum by setting the partial derivatives of $\mathcal{L}$ to zero:

$$\frac{d\mathcal{L}}{d\pi_k} = \sum_{n\in C_k}\frac{1}{\pi_k} - \lambda = set\ 0 \Rightarrow \pi_k^* = \frac{N_k}{\lambda}$$

$$\frac{d\mathcal{L}}{d\lambda} = 1 - \sum_{k=1}^{K}\pi_k = set\ 0 \Rightarrow \sum_{k=1}^{K}\pi_k^* = \sum_{k=1}^{K}\frac{N_k}{\lambda} = \frac{N}{\lambda} = 1 \Rightarrow \lambda = N \Rightarrow \pi_k^* = \frac{N_k}{N}$$

$$\frac{d\mathcal{L}}{d\lambda_{ki}} = \sum_{n\in C_k}\left(\frac{x_{ni}}{\lambda_{ki}} - 1\right) = set\ 0 \Rightarrow \lambda_{ki}^* = \frac{1}{N_k}\sum_{n\in C_k}x_{ni}$$

2.)

ii.) Full Gaussian, class covariance: $\boldsymbol{x}|C_k \sim \mathcal{N}(\boldsymbol{\mu_k}, \Sigma_k)$

For two different classes i and j, we classify $\boldsymbol{x}$ as from class i over j iff:

$$P(\boldsymbol{x}|C_i)P(C_i) > P(\boldsymbol{x}|C_j)P(C_j) \Rightarrow \ln P(\boldsymbol{x}|C_i)P(C_i) > \ln P(\boldsymbol{x}|C_j)P(C_j)$$

$$\Rightarrow \ln \pi_i - \frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu_i})$$
$$> \ln \pi_j - \frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_j| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_j})^T\Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})$$

So the decision boundary is:

$$\ln\frac{\pi_i}{\pi_j} - \frac{1}{2}\ln\frac{|\Sigma_i|}{|\Sigma_j|} - \frac{1}{2}\left[(\boldsymbol{x}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu_i}) - (\boldsymbol{x}-\boldsymbol{\mu_j})^T\Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})\right] = 0$$

This boundary is nonlinear—there are terms quadratic in $\boldsymbol{x}$.

iii.) Poisson: $x_i|C_k \sim Poisson(\lambda_{ki})$

For two different classes i and j, we classify $\boldsymbol{x}$ as from class i over j iff:

$$P(\boldsymbol{x}|C_i)P(C_i) > P(\boldsymbol{x}|C_j)P(C_j) \Rightarrow \ln P(\boldsymbol{x}|C_i)P(C_i) > \ln P(\boldsymbol{x}|C_j)P(C_j)$$

$$\Rightarrow \sum_{k=1}^{D}[\ln \pi_i + x_k \ln \lambda_{ik} - \lambda_{ik} - \ln(x_k!)] > \sum_{k=1}^{D}[\ln \pi_j + x_k \ln \lambda_{jk} - \lambda_{jk} - \ln(x_k!)]$$

So the decision boundary is:

$$D\ln\frac{\pi_i}{\pi_j} + \sum_{k=1}^{D}\left[x_k \ln\frac{\lambda_{ik}}{\lambda_{jk}} - (\lambda_{ik} - \lambda_{jk})\right] = 0$$

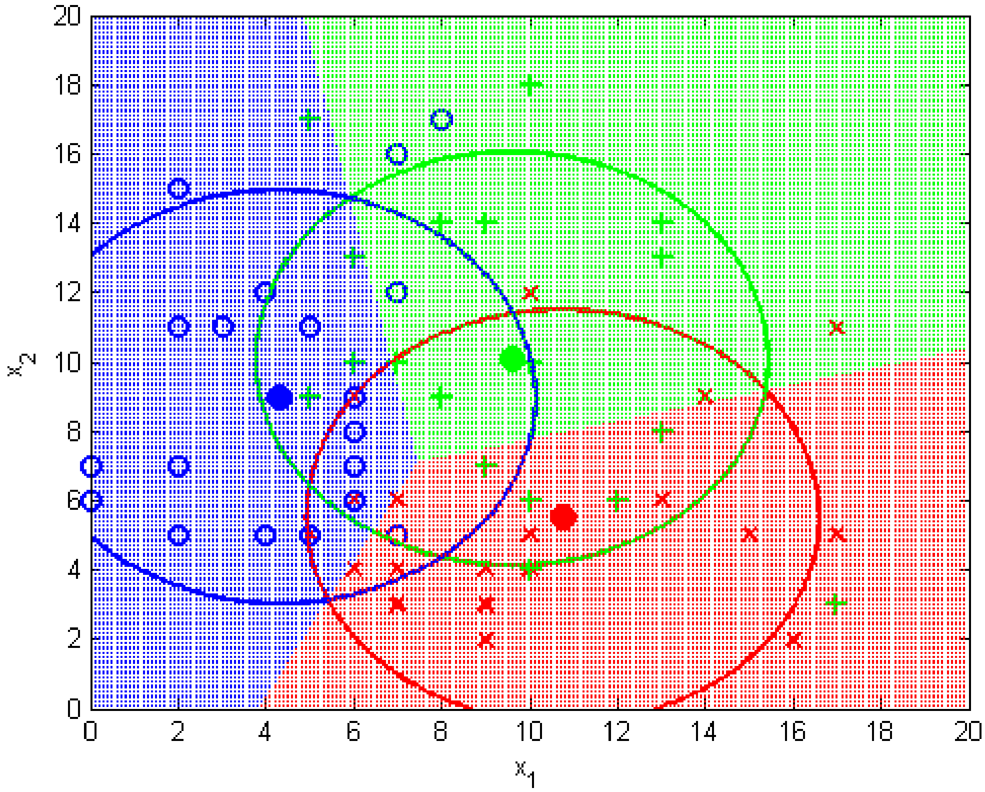
This boundary is linear—all terms involving $\boldsymbol{x}$ are linear.

Problem 3

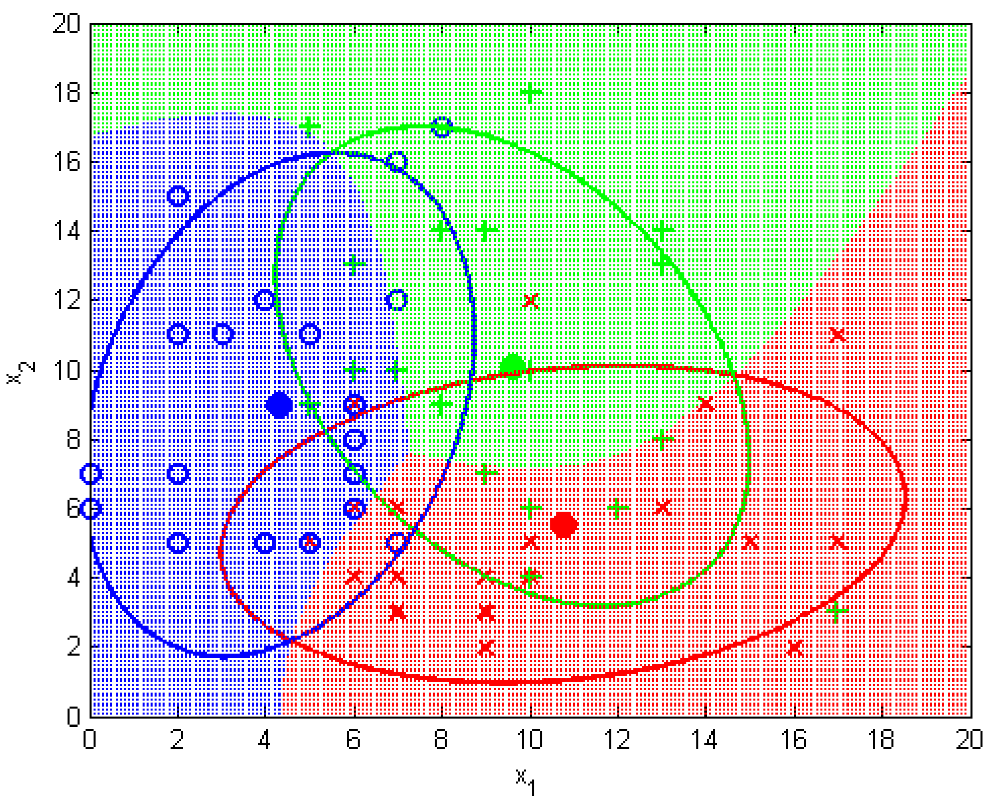
```
function PS3_3

load('ps3_simdata.mat','-mat');
[NumData NumClass]=size(trial);
for classIX=1:NumClass
    for dataIX=1:NumData
        dataArr(classIX,dataIX,:)=trial(dataIX,classIX).x;
    end
end
NumFea=size(dataArr,3);

%% %%%%%%%%%%%%%%%%%%%%%%%%% Part (b) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NumModel=3;

% For model (1)
modParam{1}.mean=squeeze(mean(dataArr,2));
% Remove the mean of each class
dataArrRemMean=reshape(repmat(modParam{1}.mean,1,NumData),[NumClass NumFea...
    NumData]);
dataArrRemMean=dataArr-permute(dataArrRemMean,[1 3 2]);
% Shared covariance matrix
modParam{1}.cov{1}=cov(reshape(dataArrRemMean,[],size(dataArr,3)));

% For model (2)
modParam{2}.mean=squeeze(mean(dataArr,2));
for classIX=1:NumClass
    modParam{2}.cov{classIX}=cov(squeeze(dataArr(classIX,:,:)));
end

% For model (3)
modParam{3}.mean=squeeze(mean(dataArr,2));

for modelIX=1:NumModel
    %% %%%%%%%%%%%%%%%%%%%%%%%%% Part (a) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    MarkerPat={'rx','g+','bo'};
    figure(modelIX);
    for classIX=1:NumClass
        plot(squeeze(dataArr(classIX,:,1)),squeeze(dataArr(classIX,:,2)),...
            MarkerPat{classIX},'LineWidth',2,'MarkerSize',8);
        hold on;
    end
    axis([0 20 0 20]);
    xlabel('x_1');
    ylabel('x_2');

    %% %%%%%%%%%%%%%%%%%%%%%%%%% Part (c) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    MarkerCol={'r','g','b'};
    for classIX=1:NumClass
       plot(modParam{modelIX}.mean(classIX,1),modParam{modelIX}.mean(classIX,2),...
            'o','MarkerEdgeColor',MarkerCol{classIX},'MarkerFaceColor',...
            MarkerCol{classIX},'MarkerSize',10)
        hold on;
    end

    %% %%%%%%%%%%%%%%%%%%%%%%%%% Part (d) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Skip this part if using model (3)
    if modelIX<3
```

```
        NumCov=length(modParam{modelIX}.cov);
        [X,Y] = meshgrid(0:.1:20,0:.1:20);
        feaVec=cat(3,X,Y);
        feaVec=reshape(feaVec,[],size(feaVec,3));
        for classIX=1:NumClass
            currMean=modParam{modelIX}.mean(classIX,:);
            covIX=min(classIX,NumCov);
            currCov=modParam{modelIX}.cov{covIX};

            % For each f=(x,y) calculate:
            % z=exp(-(x-f)*inv(cov)*(x f)/2)/sqrt(det(cov))
            Z=sum(((feaVec-repmat(currMean,size(feaVec,1),1))*inv(currCov))...
                .*(feaVec-repmat(currMean,size(feaVec,1),1)),2);
            Z=exp(-Z/2)/sqrt(det(currCov));

            Z=reshape(Z,size(X));

            isoThr=0.02;
            contour(X,Y,Z,isoThr,MarkerCol{classIX},'LineWidth',2);
            hold on;
        end
    end

    %% %%%%%%%%%%%%%%%%%%%%%%%%% Part (e) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Generate dense samples
    [X,Y] = meshgrid(0:.1:20,0:.1:20);
    feaVec=cat(3,X,Y);
    feaVec=reshape(feaVec,[],size(feaVec,3));
    NumGrid=size(feaVec,1);

    switch modelIX
        % If using model (1)
        case {1}
            for classIX=1:NumClass
                tmp=feaVec-repmat(modParam{modelIX}.mean(classIX,:),NumGrid,1);
                logP(:,classIX)=sum(tmp*inv(modParam{modelIX}.cov{1}).*tmp,2);
            end
            [minVal classLabel]=min(logP,[],2);
        % If using model (2)
        case {2}
            for classIX=1:NumClass
                tmp=feaVec-repmat(modParam{modelIX}.mean(classIX,:),NumGrid,1);
                logP(:,classIX)=sum(tmp*inv(modParam{modelIX}.cov{classIX}).*...
                    tmp,2)+log(det(modParam{modelIX}.cov{classIX}));
            end
            [minVal classLabel]=min(logP,[],2);
        % If using model (3)
        case {3}
            for classIX=1:NumClass
                currLambda=modParam{modelIX}.mean(classIX,:);
                logP(:,classIX)=-feaVec*log(currLambda')+sum(currLambda);
            end
            [minVal classLabel]=min(logP,[],2);
    end
    h=gscatter(X(:),Y(:),classLabel,'rgb','.',[],'off');
    set(h, 'Markersize', 1);
    hold on;
end
return;
```

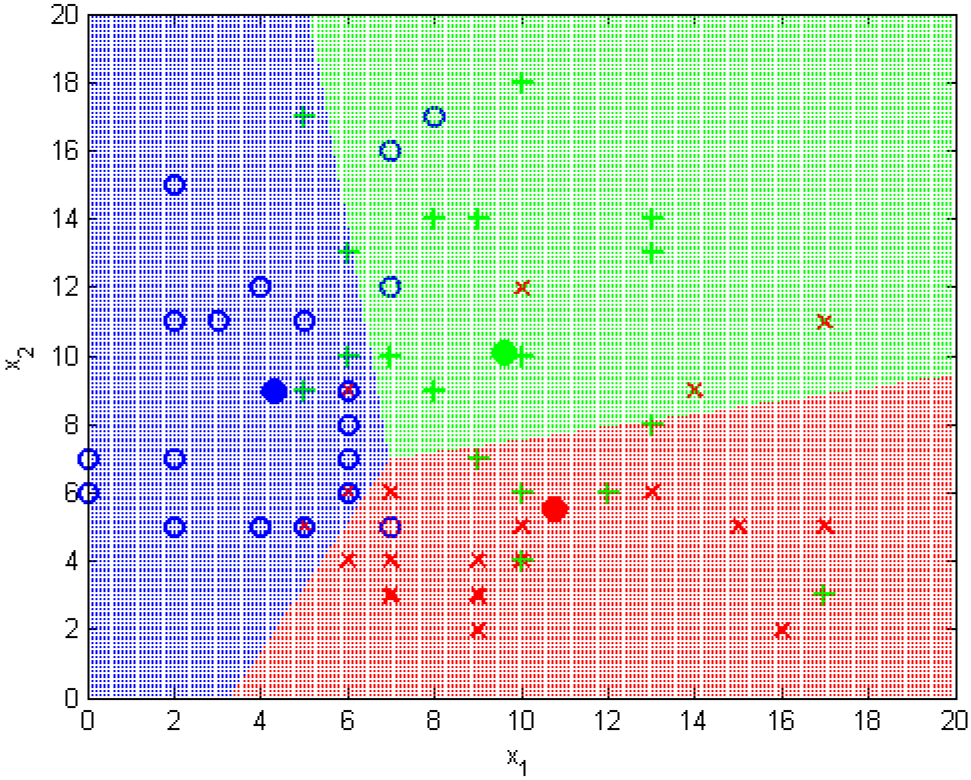
Model (i)



Model (ii)



Model (iii)



```matlab
function PS3_4

%% %%%%%%%%%%%%%% Feature Extraction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
load('ps3_realdata.mat','-mat');
[NumTrainData NumClass]=size(train_trial);
NumTestData=size(test_trial,1);
for classIX=1:NumClass
for trainDataIX=1:NumTrainData
currData=train_trial(trainDataIX,classIX).spikes(:,351:550);
trainDataArr(classIX,trainDataIX,:)=sum(currData,2);
end
for testDataIX=1:NumTestData
currData=test_trial(testDataIX,classIX).spikes(:,351:550);
testDataArr(classIX,testDataIX,:)=sum(currData,2);
end
end
NumFea=size(trainDataArr,3);
% For test data
actLabel=repmat([1:NumClass]',1,NumTestData);
testData=reshape(testDataArr,[],NumFea);
%% %%%%%%%%%%%%%%%%%%%%%%%%% Part (a) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Fit the parameters of model(1)
modMean=squeeze(mean(trainDataArr,2));
% Remove the mean of each class
trainDataArrRemMean=reshape(repmat(modMean,1,NumTrainData),[NumClass
NumFea...
NumTrainData]);
trainDataArrRemMean=trainDataArr-permute(trainDataArrRemMean,[1 3 2]);
% Shared covariance matrix
modCov{1}=cov(reshape(trainDataArrRemMean,[],NumFea));
% Test
for classIX=1:NumClass
tmp=testData-repmat(modMean(classIX,:),NumTestData*NumClass,1);
logP(:,classIX)=sum(tmp*inv(modCov{1}).*tmp,2);
end
[minVal predLabel]=min(logP,[],2);
corrPred=find((predLabel-actLabel(:))==0);
corrRatio_a=length(corrPred)/(NumTestData*NumClass);
%% %%%%%%%%%%%%%%%%%%%%%%%%% Part (b) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Fit the parameters of model(2)
% The covariance matrices are singular
modMean=squeeze(mean(trainDataArr,2));
for classIX=1:NumClass
modCov{classIX}=cov(squeeze(trainDataArr(classIX,:,:)));
end
% Test
for classIX=1:NumClass
tmp=testData-repmat(modMean(classIX,:),NumTestData*NumClass,1);
logP(:,classIX)=sum(tmp*inv(modCov{classIX}).*tmp,2)+log(det(modCov{cla
ssIX}));
end
[minVal predLabel]=min(logP,[],2);
```

```
corrPred=find((predLabel-actLabel(:))==0);
corrRatio_b=length(corrPred)/(NumTestData*NumClass);
%% %%%%%%%%%%%%%%%%%%%%%%%%% Part (c) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Fit the parameters of model (3)
modMean=squeeze(mean(trainDataArr,2));
% Test
for classIX=1:NumClass
currLambda=modMean(classIX,:);
logP(:,classIX)=-testData*log(currLambda')+sum(currLambda);
end
[minVal predLabel]=min(logP,[],2);
corrPred=find((predLabel-actLabel(:))==0);
corrRatio_c=length(corrPred)/(NumTestData*NumClass);
%% %%%%%%%%%%%%%%%%%%%%%%%%% Part (d) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% Fit the parameters of model (3) with min variance set
modMean=squeeze(mean(trainDataArr,2));
regCoef=0.01;
modMean=max(modMean,regCoef);
% Test
for classIX=1:NumClass
currLambda=modMean(classIX,:);
logP(:,classIX)=-testData*log(currLambda')+sum(currLambda);
end
[minVal predLabel]=min(logP,[],2);
corrPred=find((predLabel-actLabel(:))==0);
corrRatio_d=length(corrPred)/(NumTestData*NumClass);
%% %%%%%%%%%%%% Print out the classification results %%%%%%%%%%%%%%%%%
%%%
fprintf('(a) Classification accuracy of model(1): %2.2f%
%\n',corrRatio_a*100);
fprintf('(b) Classification accuracy of model(2): %2.2f%
%\n',corrRatio_b*100);
fprintf(' Not enough training data to fit a full covariance matrix
(random chance level)\n');
fprintf('(c) Classification accuracy of model(3): %2.2f%
%\n',corrRatio_c*100);
fprintf(' Poisson distribution has 0 var for \mu=0 (random chance
level)\n');
fprintf('(d) Classification accuracy of model(3) with minimum variance
set: %2.2f%%\n',corrRatio_d*100);
return;

(a) Classification accuracy of model(1): 96.02%
(b) Classification accuracy of model(2): 12.5%
      Not enough training data to fit a full covariance matrix (random
chance level)
(c) Classification accuracy of model(3): 12.5%
      Poisson distribution has 0 var for \mu=0 (random chance level)
(d) Classification accuracy of model(3) with minimum variance set:
94.09%
```