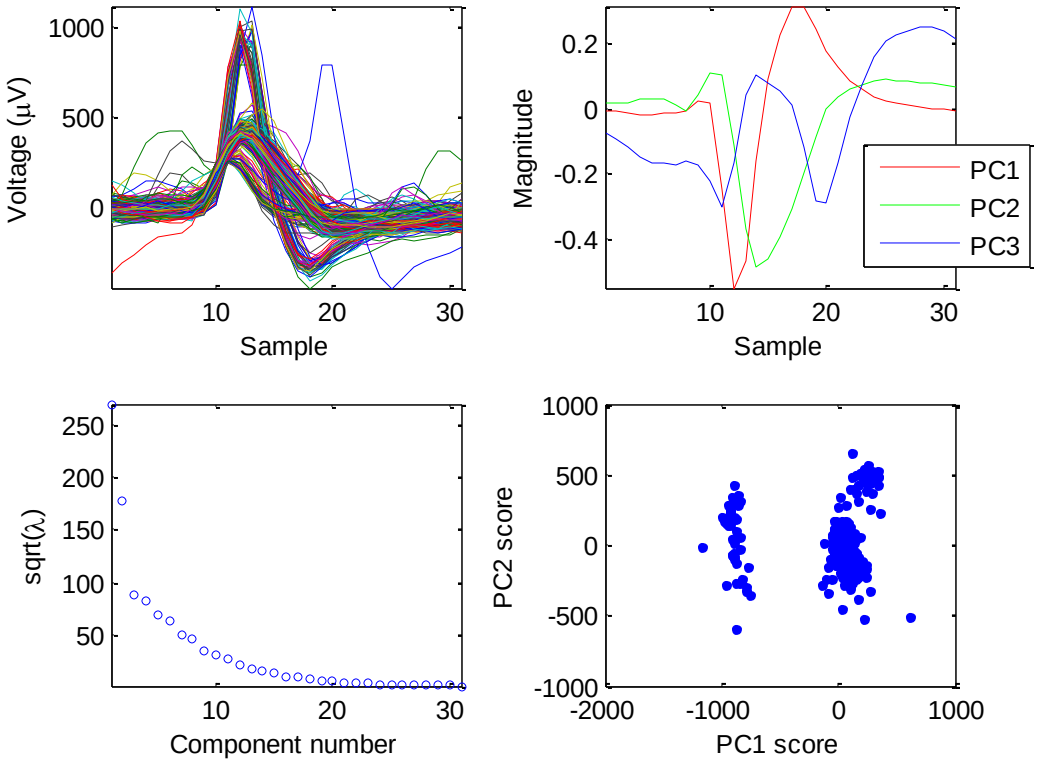
18-698 / 42-632
Neural Signal Processing
Problem Set 7 Solutions

1.) Code:

```
%% Problem 7.1
clear; close all;
load('ps7_data.mat');
[NTime, NSpikes] = size(Spikes);
%% ############################## Part (a) ##############################
% Plot the raw spikes
subplot(2,2,1); plot(Spikes);
xlabel('Sample'); ylabel('Voltage (\muV)');
axis tight
%% ############################## Part (b) ##############################
% Apply PCA
[comp, score, eigVal] = princomp(Spikes');
comp = -comp;
score = -score;
% Plot the first three principal components
subplot(2, 2, 2); plot(comp(:, 1), '-r');
hold on; plot(comp(:, 2), '-g'); hold on; plot(comp(:, 3), '-b');
xlabel('Sample'); ylabel('Magnitude'); legend('PC1', 'PC2', 'PC3');
axis tight
%% ############################## Part (c) ##############################
% Plot the square-rooted eigenvalue spectrum
subplot(2, 2, 3); plot(sqrt(eigVal), 'o', 'MarkerSize', 2);
xlabel('Component number'); ylabel('sqrt(\lambda)');
axis tight
%% ############################## Part (d) ##############################
% Create a scatter plot of PC1 score versus the PC2 score
subplot(2, 2, 4); plot(score(:, 1), score(:, 2), '.', 'MarkerSize', 5);
xlabel('PC1 score'); ylabel('PC2 score');
return;
```

c.) There are approximately 2 dominant eigenvalues, as seen by the elbow.

d.) I see 3 distinct clusters in the plot.

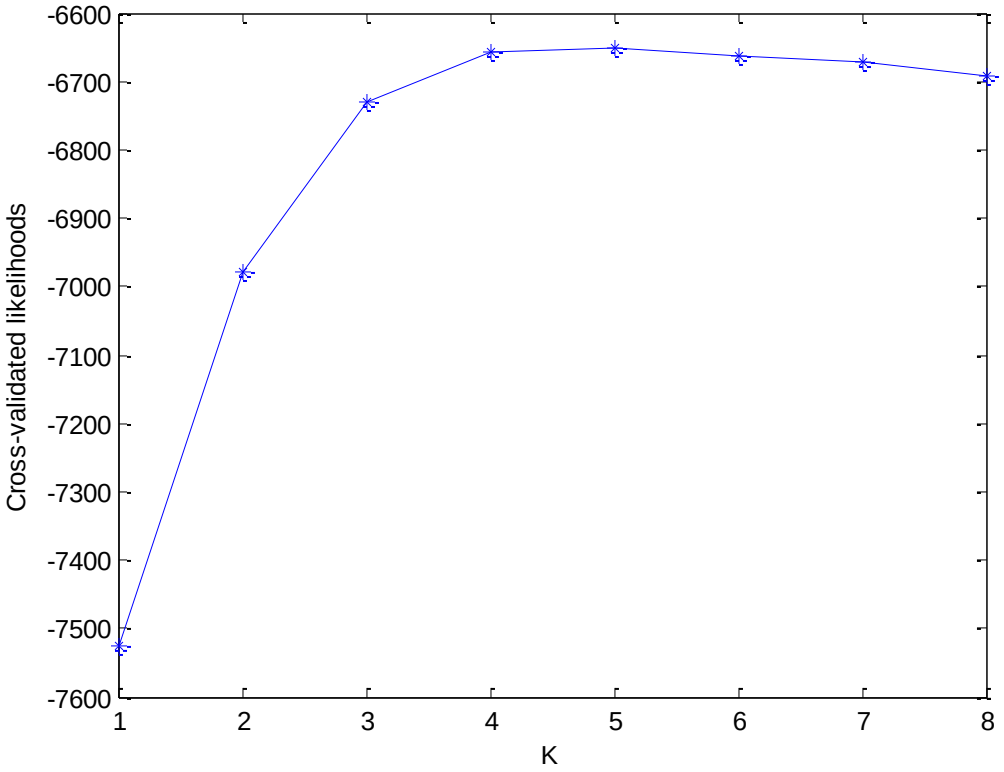

1000
500
0
Voltage (μV)
10
20
30
Sample
0.2
0
-0.2
-0.4
Magnitude
PC1
PC2
PC3
10
20
30
Sample
250
200
150
100
50
sqrt(λ)
10
20
30
Component number
1000
500
0
-500
-1000
PC2 score
-2000
-1000
0
1000
PC1 score

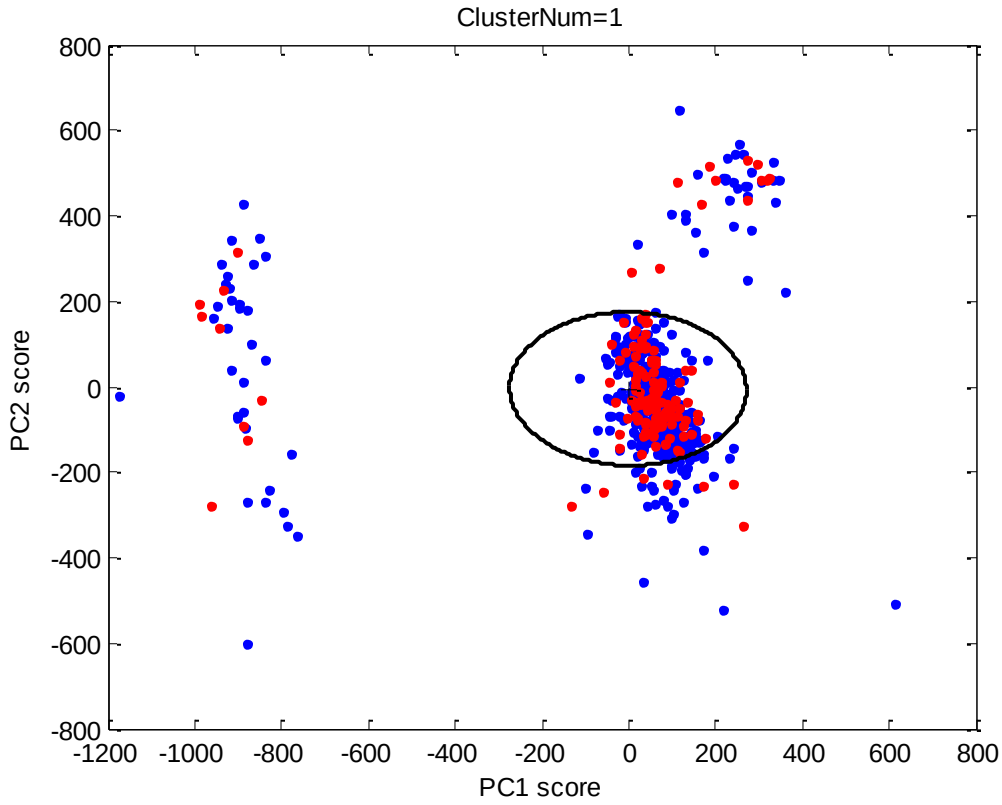
2.) Code:

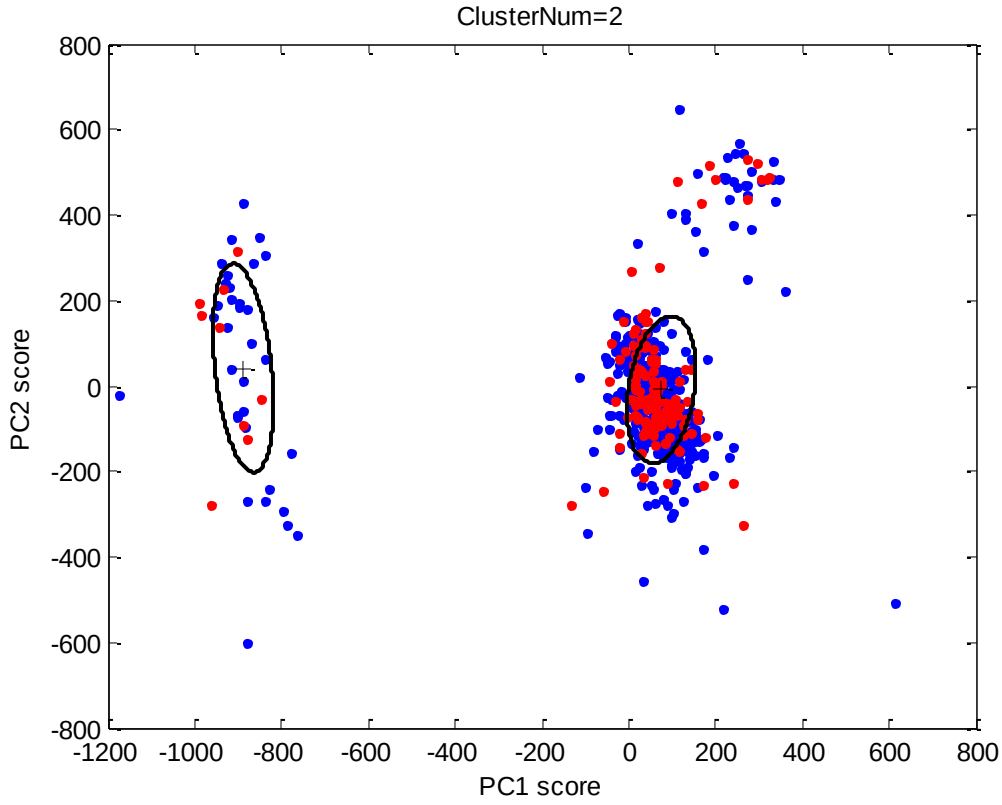
```
close all; clear all; clc;
load('ps7_data.mat');
[NTime, NSpikes]=size(Spikes);
% Project the spikes to a two-dimensional space using PCA
[comp, score]=princomp(Spikes');
% negate the directios
comp = -comp;
score = -score;

NumFea=2;
selFea=score(:,1:NumFea);
% Compute mean of the spikes
meanSpikes=mean(Spikes,2);
%% GMM model selection
% Set the K= 1, ..., 8
NumClusterList=1:8;
plotCount=0;
for clusterIX=1:length(NumClusterList)
    NumCluster=NumClusterList(clusterIX);
    % Initialize the GMM parameters
    currInitParams.mu=InitParams.mu(:,1:NumCluster);
    for k=1:NumCluster
        currInitParams.Sigma(:,:,k)=InitParams.Sigma;
        currInitParams.pi(k)=1/NumCluster;
    end
    % Corss-validation
    NumFold=4;
    NumPerFold=NSpikes/NumFold;
    for foldIX=1:NumFold
        % Separate the training set and the test set
        testIX=((foldIX-1)*NumPerFold+1):(foldIX*NumPerFold);
        trainIX=setdiff(1:NSpikes,testIX);
        testData=selFea(testIX,:);
        trainData=selFea(trainIX,:);
        % Training
        [mu, sigma, ppi]=func_GMM(currInitParams, trainData');
        % Test (compute the likelihood of each test fold)
        const = -0.5 * NumFea * log(2*pi);
        logMat = nan(NumCluster, length(testIX));
        for k = 1:NumCluster
            S = sigma(:,:,k);
            xdif = bsxfun(@minus, testData', mu(:,k));
            term1 = -0.5 * sum((xdif' * inv(S)) .* xdif', 2); % N x 1
            term2 = const - 0.5 * log(det(S)) + log(ppi(k)); % scalar
            logMat(k,:) = term1' + term2;
        end
        astar = max(logMat, [], 1);
        adif = bsxfun(@minus, logMat, astar);
        nLL = log(sum(exp(adif), 1)) + astar; % 1 x N
        LL(foldIX) = sum(nLL);
        %% %%%%%%%%%%%%%%%%%%%%%%%% Part (b) %%%%%%%%%%%%%%%%%%%%%%%%
        % Plot the Gaussian distribution from the first CV fold
        if (foldIX==1)
            plotCount=plotCount+1;
```

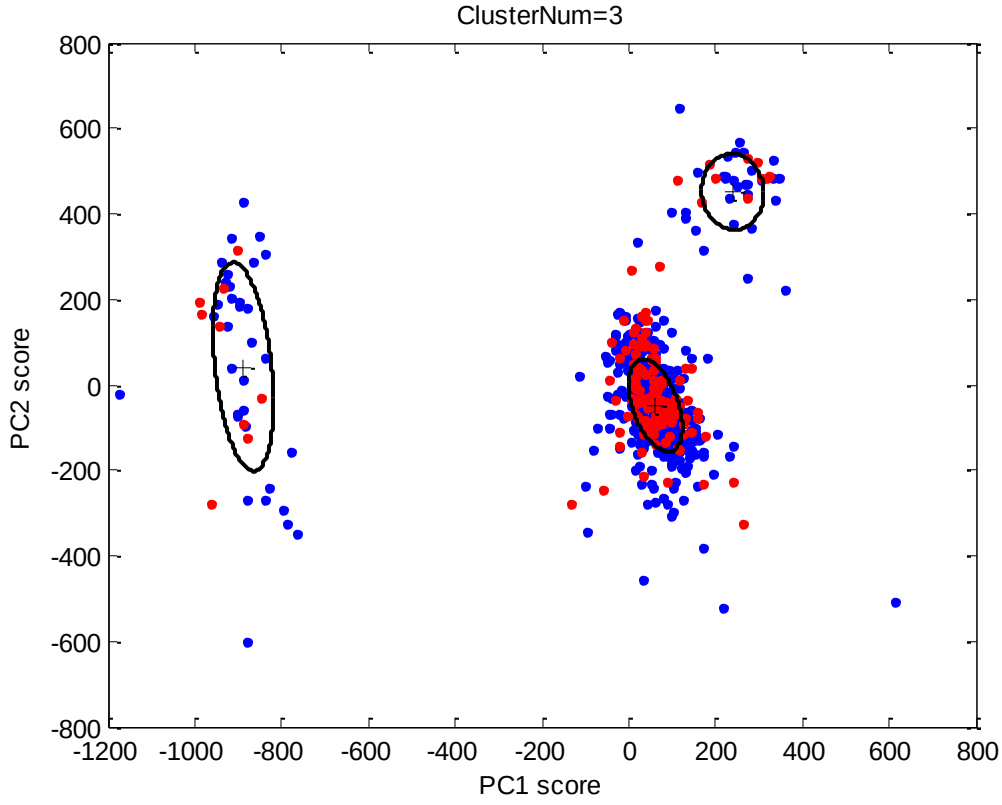
```
            figure(plotCount);
            % Plot the training data in blue
            plot(trainData(:,1),trainData(:,2),'b.');hold on;
            % Plot the training data in red
            plot(testData(:,1),testData(:,2),'r.'); hold on;
            for k=1:NumCluster
                func_plotEllipse(mu(:,k),sigma(:,:,k));
                hold on;
            end
            xlabel('PC1 score'); ylabel('PC2 score');
            titleStr=sprintf('ClusterNum=%d',NumCluster);
            title(titleStr);
        end
        %% %%%%%%%%%%%%%%%%%%%%%%%% Part (c) %%%%%%%%%%%%%%%%%%%%%%%%
        % For K=3, plot the waveforms corresponding to GMM cluster
centers
        if (NumCluster==3 && foldIX==1)
            recCenter=comp(:,1:2)*mu+repmat(meanSpikes,1,3);
            plotCount=plotCount+1;
            figure(plotCount); plot(recCenter);
            xlabel('Sample'); ylabel('Voltage (\muV)');
            axis tight
            title ('K=3: Waveforms corresponding to GMM cluster
centers');
        end
    end
    % The CV likelihoods for K=NumClusterList(clusterIX)
    totLL(clusterIX)=sum(LL);
end
%% %%%%%%%%%%%%%%%%%%%%%%%% Part (a) %%%%%%%%%%%%%%%%%%%%%%%%
% Plot the CV likelihoods versus K
plotCount=plotCount+1;
figure(plotCount); plot(totLL,'-*')
xlabel('K'); ylabel('Cross-validated likelihoods');
```

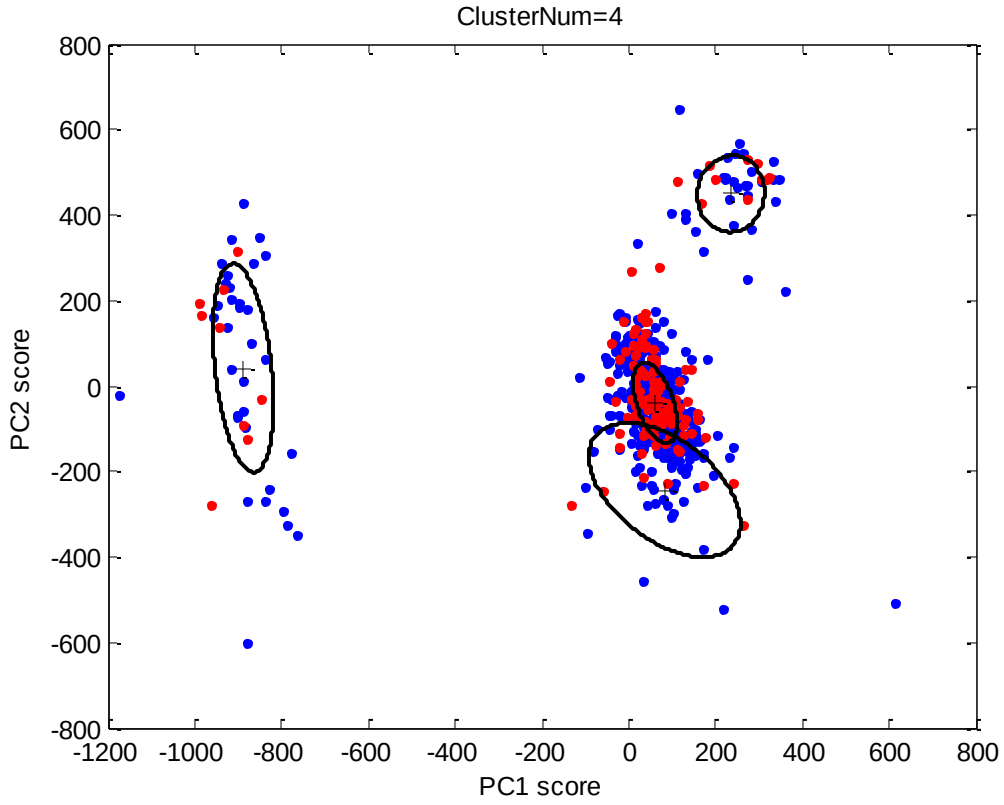
a.) The optimal value of K (the one with highest cross-validated likelihood) is 5 clusters.

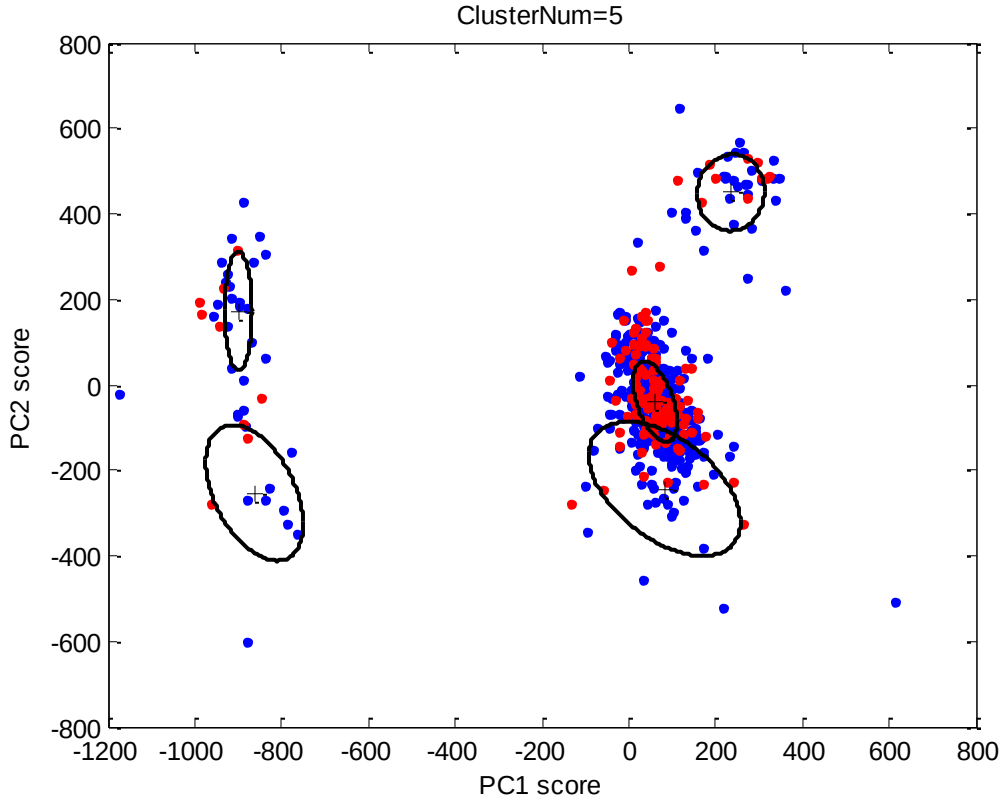

-6600
-6700
-6800
-6900
-7000
-7100
-7200
-7300
-7400
-7500
-7600
Cross-validated likelihoods
1
2
3
4
5
6
7
8
K

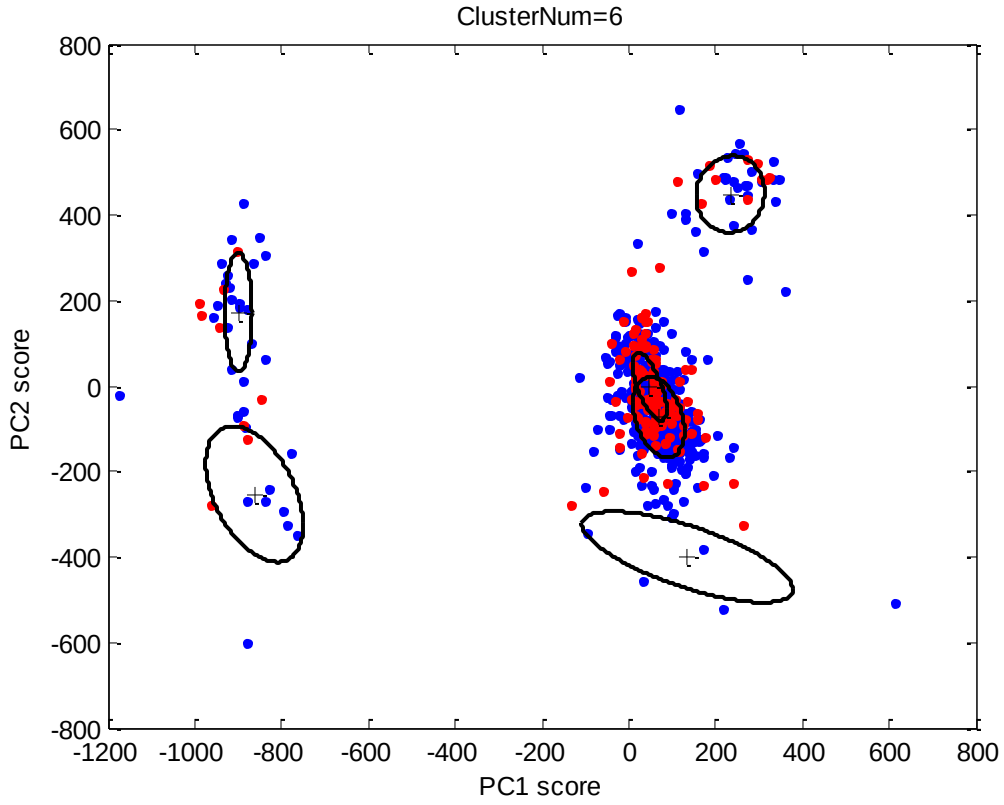

ClusterNum=1
800
600
400
200
0
-200
-400
-600
-800
PC2 score
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800
PC1 score

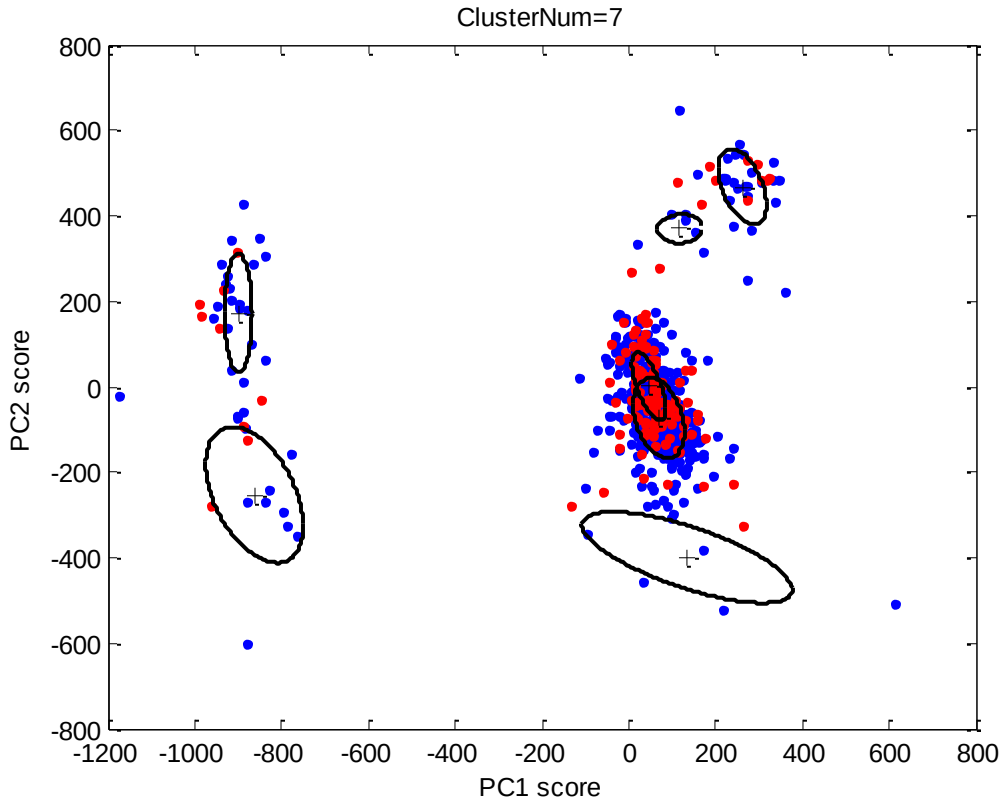

ClusterNum=2
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800

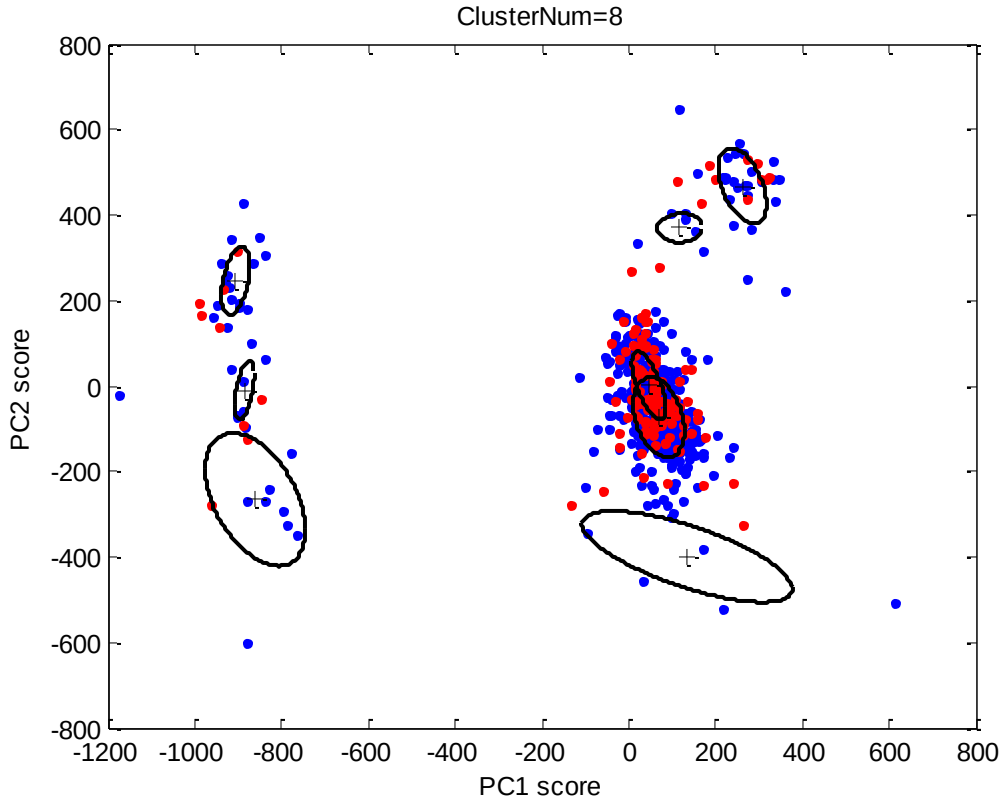

ClusterNum=3
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800

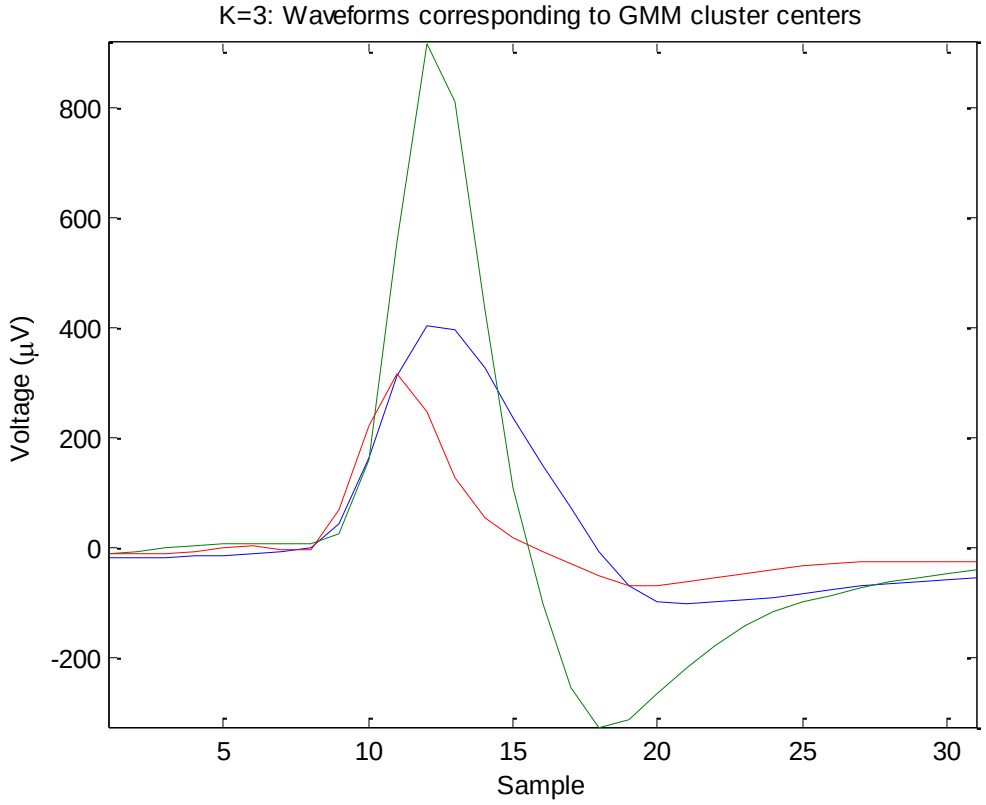

ClusterNum=4
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800


ClusterNum=5
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800


ClusterNum=6
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800


ClusterNum=7
PC2 score
PC1 score
800
600
400
200
0
-200
-400
-600
-800
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800


ClusterNum=8
800
600
400
200
0
-200
-400
-600
-800
PC2 score
-1200
-1000
-800
-600
-400
-200
0
200
400
600
800
PC1 score


K=3: Waveforms corresponding to GMM cluster centers
800
600
400
200
0
-200
Voltage (μV)
5
10
15
20
25
30
Sample

Appendix:

```
function func_plotEllipse(M,V)
%
% func_plotEllipse(M,V)
%
% Plot a one-standard-deviation ellipse for a N-dimensional Gaussian
distribution
%
% INPUTS:
%    M - covariance matrix of the Gaussian (N x N)
%    V - mean of the Gaussian (N x 1)

% Code adapted from EM_GM.m by Patrick P. C. Tsui.

[Ev,D] = eig(V);
d = length(M);
if V(:,:)==zeros(d,d),
    V(:,:) = ones(d,d)*eps;
end
iV = inv(V);
% Find the larger projection
P = [1,0;0,0];  % X-axis projection operator
P1 = P * 2*sqrt(D(1,1)) * Ev(:,1);
P2 = P * 2*sqrt(D(2,2)) * Ev(:,2);
if abs(P1(1)) >= abs(P2(1)),
    Plen = P1(1);
else
    Plen = P2(1);
end
count = 1;
step = 0.001*Plen;
Contour1 = zeros(2001,2);
Contour2 = zeros(2001,2);
for x = -Plen:step:Plen,
    a = iV(2,2);
    b = x * (iV(1,2)+iV(2,1));
    c = (x^2) * iV(1,1) - 1;
    Root1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
    Root2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
    if isreal(Root1),
        Contour1(count,:) = [x,Root1] + M';
        Contour2(count,:) = [x,Root2] + M';
        count = count + 1;
    end
end
Contour1 = Contour1(1:count-1,:);
Contour2 = [Contour1(1,:);Contour2(1:count-1,:);Contour1(count-1,:)];
plot(M(1),M(2),'k+');
plot(Contour1(:,1),Contour1(:,2),'k-','LineWidth',2);
plot(Contour2(:,1),Contour2(:,2),'k-','LineWidth',2);
```

```
function [mu, Sigma, ppi, gam, LL]=func_GMM(InitParams,Spikes)
%
% [mu, Sigma, ppi]=func_GMM(InitParams,Spikes)
%
% EM algorithm for Gaussian Mixture Model estimation
%
% xDim: data dimensionality
% zDim: number of mixture components
% N: number of data points
%
% INPUTS:
% InitParams - a 1x1 structure containing two fields
% InitParams.mu - initialization of mean vectors of GMs (xDim x zDim)
% InitParams.Sigma - initialization of covariance matrices of GMs (xDim
x xDim x
% zDim)
% Spikes - input data (xDim x N)
%
% OUTPUTS:
% mu - estimated mean vectors of GMs (xDim x zDim)
% Sigma - estimated covariance matrices of GMs (xDim x xDim x zDim)
% ppi - estimated weights of GMs
% gam - estimated responsibilities of each cluster to each data point
(zDim x xDim)
% LL - estimated log-likelihood at each iteration
mu = InitParams.mu;
ppi = InitParams.pi;
K = size(mu, 2);
[D, N] = size(Spikes);
Sigma = InitParams.Sigma;
const = -0.5 * D * log(2*pi);
for i = 1:100
% === E-step ===
logMat = nan(K, N);
for k = 1:K
S = Sigma(:,:,k);
xdif = bsxfun(@minus, Spikes, mu(:,k));
term1 = -0.5 * sum((xdif' * inv(S)) .* xdif', 2); % N x 1
% term2 = const - 0.5 * logDete(S) + log(ppi(k)); % scalar
term2 = const - 0.5 * log(det(S)) + log(ppi(k)); % scalar
logMat(k,:) = term1' + term2;
end
% Evaluate log P({x})
astar = max(logMat, [], 1);
adif = bsxfun(@minus, logMat, astar);
nLL = log(sum(exp(adif), 1)) + astar; % 1 x N
LL(i) = sum(nLL);
gam = exp(bsxfun(@minus, logMat, nLL)); % K x N (responsibilities)
gam = bsxfun(@rdivide, gam, sum(gam, 1)); % for numerical stability
% === M-step ===
Neff = sum(gam, 2);
ppi = Neff' / N;
for k = 1:K
mu(:,k) = (Spikes * gam(k,:)') / Neff(k);
xdif = bsxfun(@minus, Spikes, mu(:,k));
S = bsxfun(@times, xdif, gam(k,:)) * xdif' / Neff(k);
Sigma(:,:,k) = (S + S') / 2; % for numerical stability
```

```
end
end
return;
```