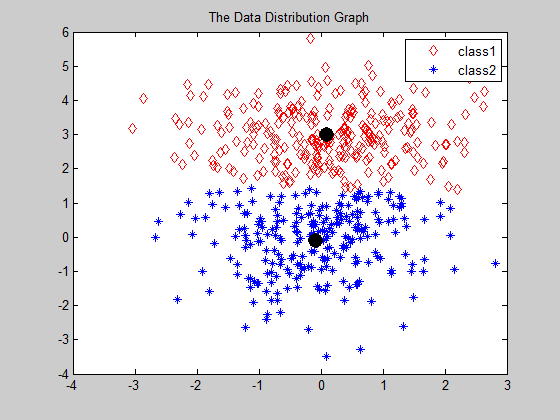
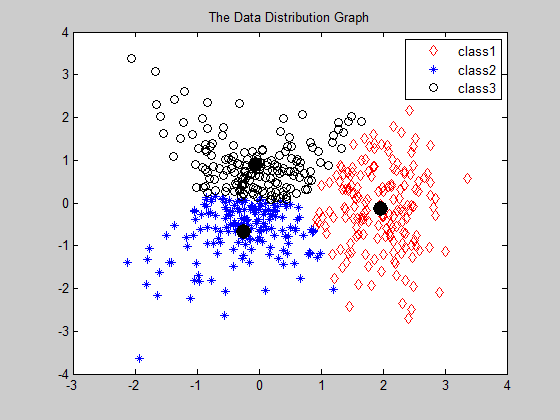
**Algorithm 1: K-Means Clustering**

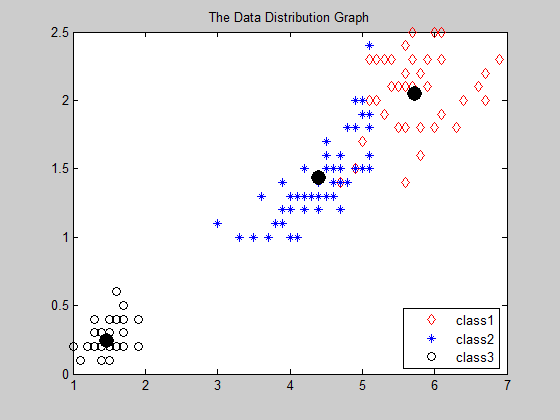
1. **For each data set, using the true K value for each one, showing the following:**
2. The K-means solution and the location of the solution (the cluster means), and plotting the data from different clusters with different symbols.
3. **Data set 1:**



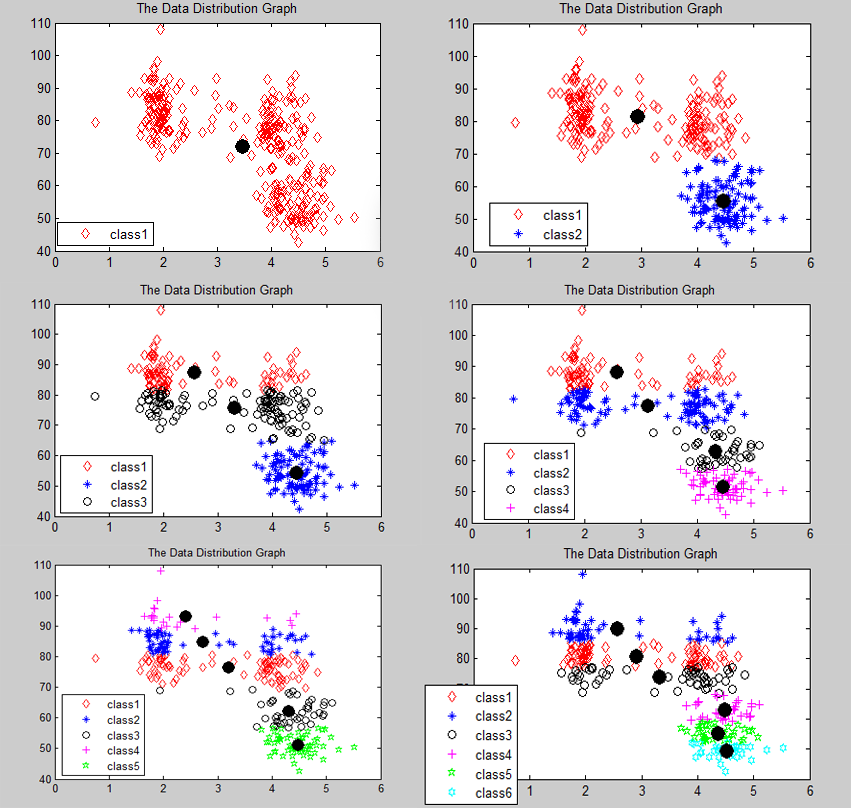
1. **Data set 2**



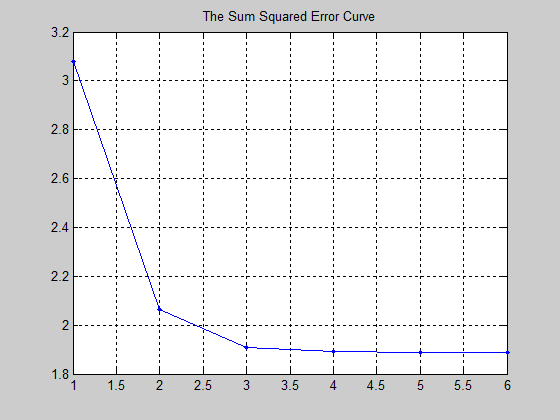
1. **Data set 3**



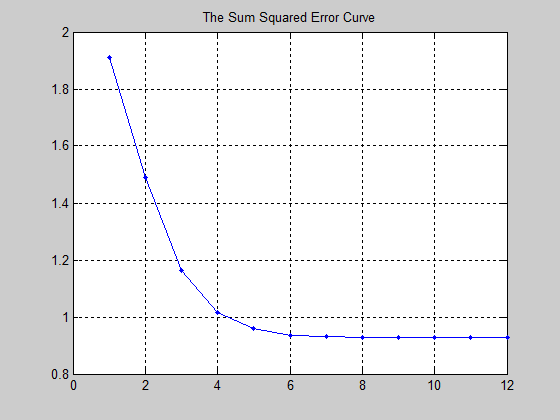
1. **Data set 4 (K from 1 to 6)**



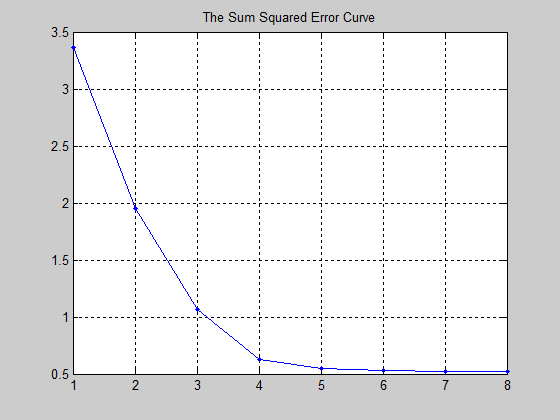
1. A plot of the sum-square-error as a function of iteration number in the K-means algorithm.
2. **Data set 1**



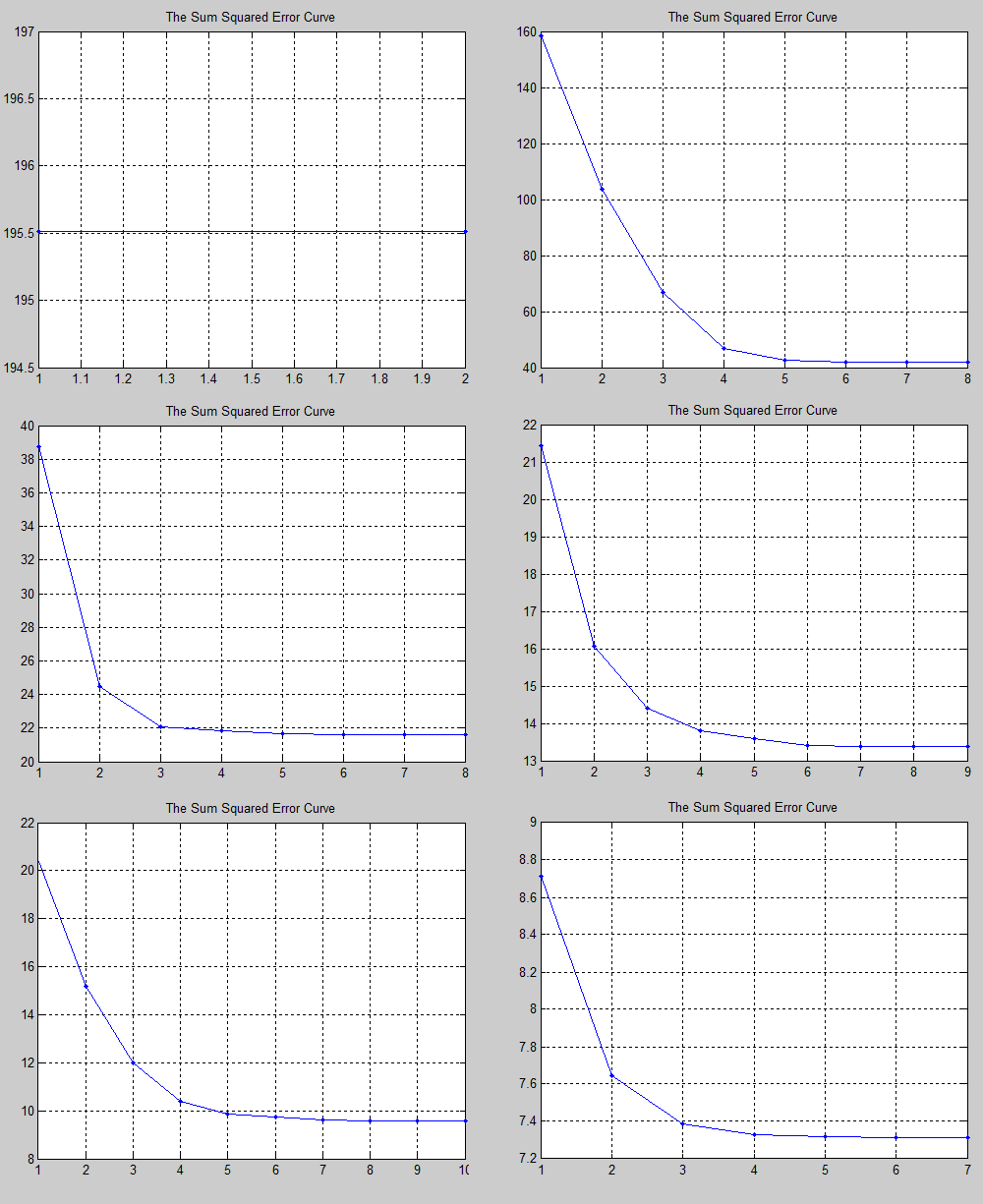
1. **Data set 2**



1. **Data set 3**



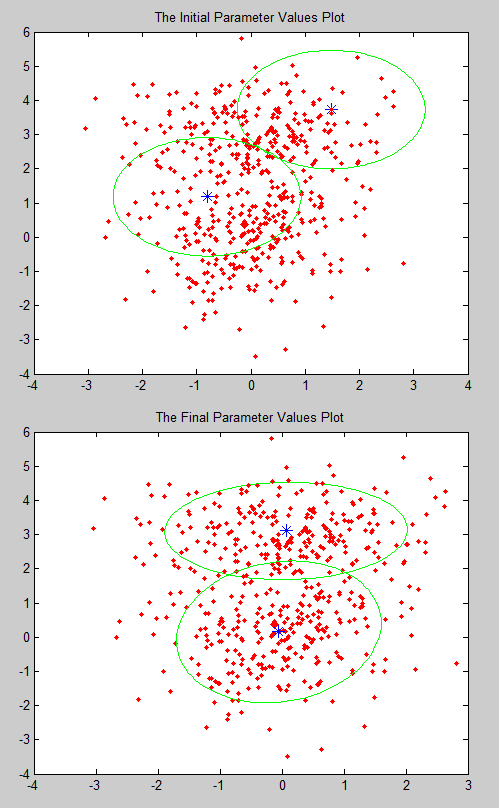
1. **Data set 4 (K from 1 to 6)**



BIC MAXIMUM WHEN K=4

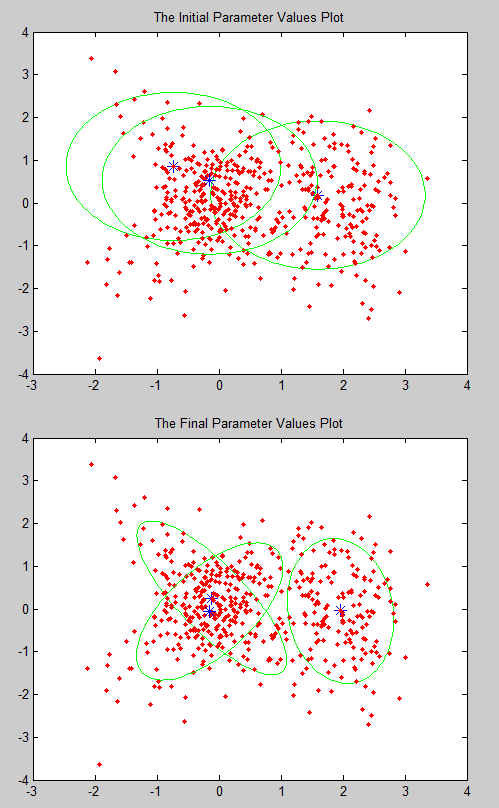
**Algorithm 2 Gaussian Mixture Clustering**

1. The initial parameter values and the final parameter values for the EM/Gaussian mixtures code for the highest-likelihood solution.
2. **Data set 1**



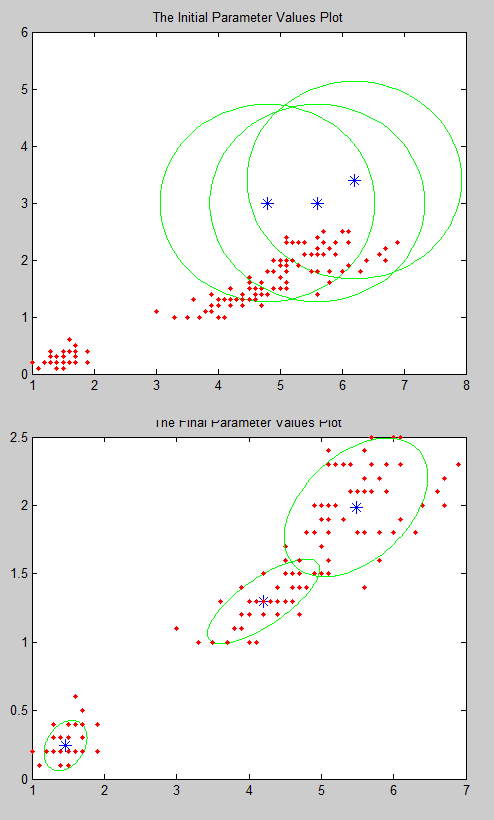
Comment: From the different between K-mean and EM clustering in data set 1, we can get the EM clustering is much more precise than K-mean clustering.

1. **Data set 2**



Comment: From the different between K-mean and EM clustering in data set 2, we can get the EM clustering can distinguish the overlap data which the K-mean is just a kind of hard clustering.

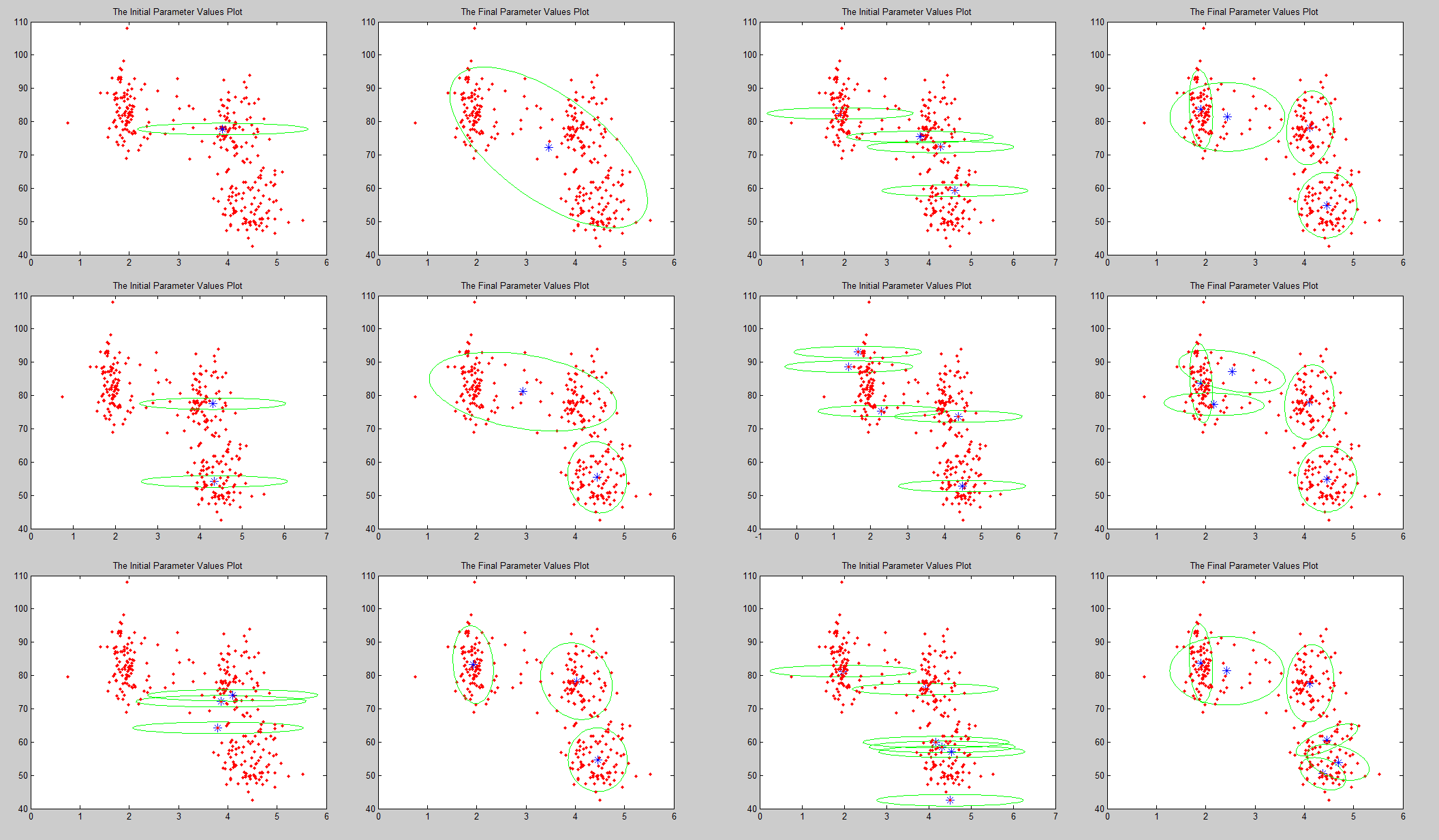
1. **Data set 3**



Comment: From the different between K-mean and EM clustering in data set 3, we can get the EM clustering can also have a good clustering of the separate data which is the same as k-mean clustering.

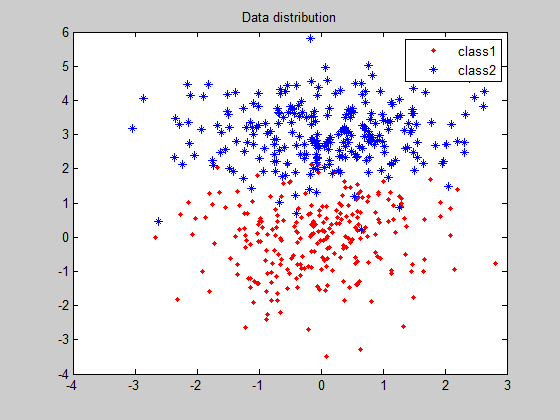
1. **Data set 4(K From 1 to6)**

BIC MAXIMUM WHEN K=4

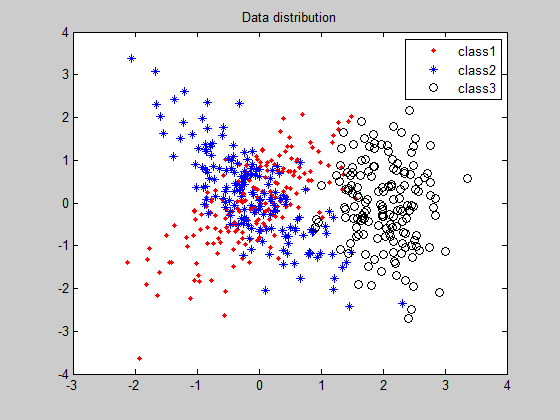


Comment: From the different between K-mean and EM clustering in data set 4, we can know that when K is increasing, the EM clustering also can distinguish the overlap data which the K-mean is just a kind of hard clustering.

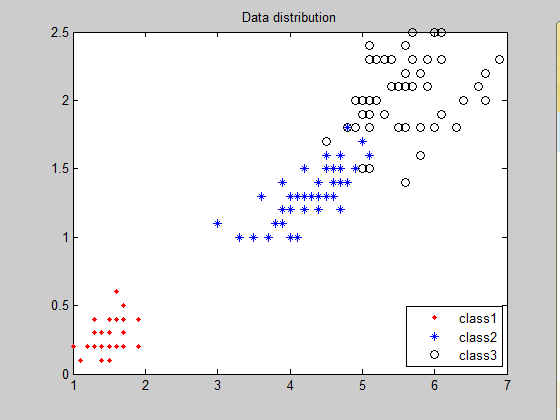
1. Show a scatter plot of the data and display the data coming from different classes (using the true labeled classes) with different colors and/or symbols.
2. **Data set 1**



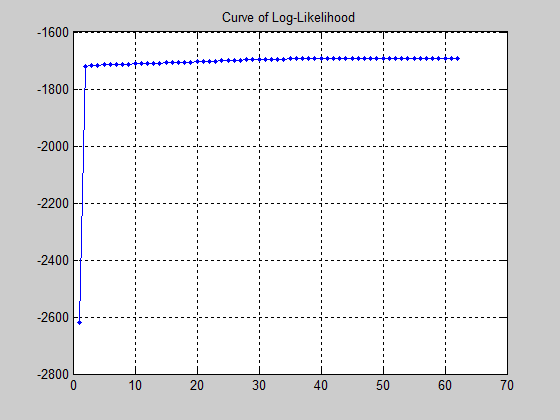
1. **Data set 2**



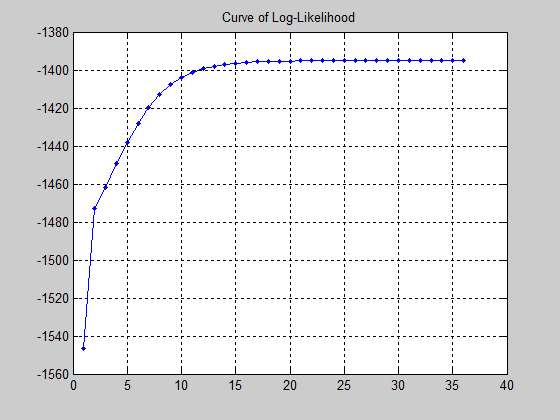
1. **Data set 3**



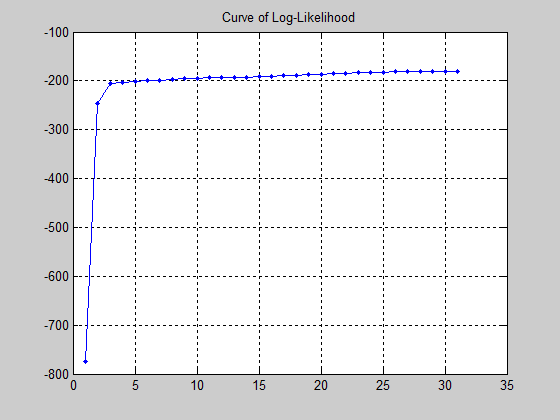
1. A plot of the log-likelihood as a function of iteration number during EM.
2. **Data set 1**



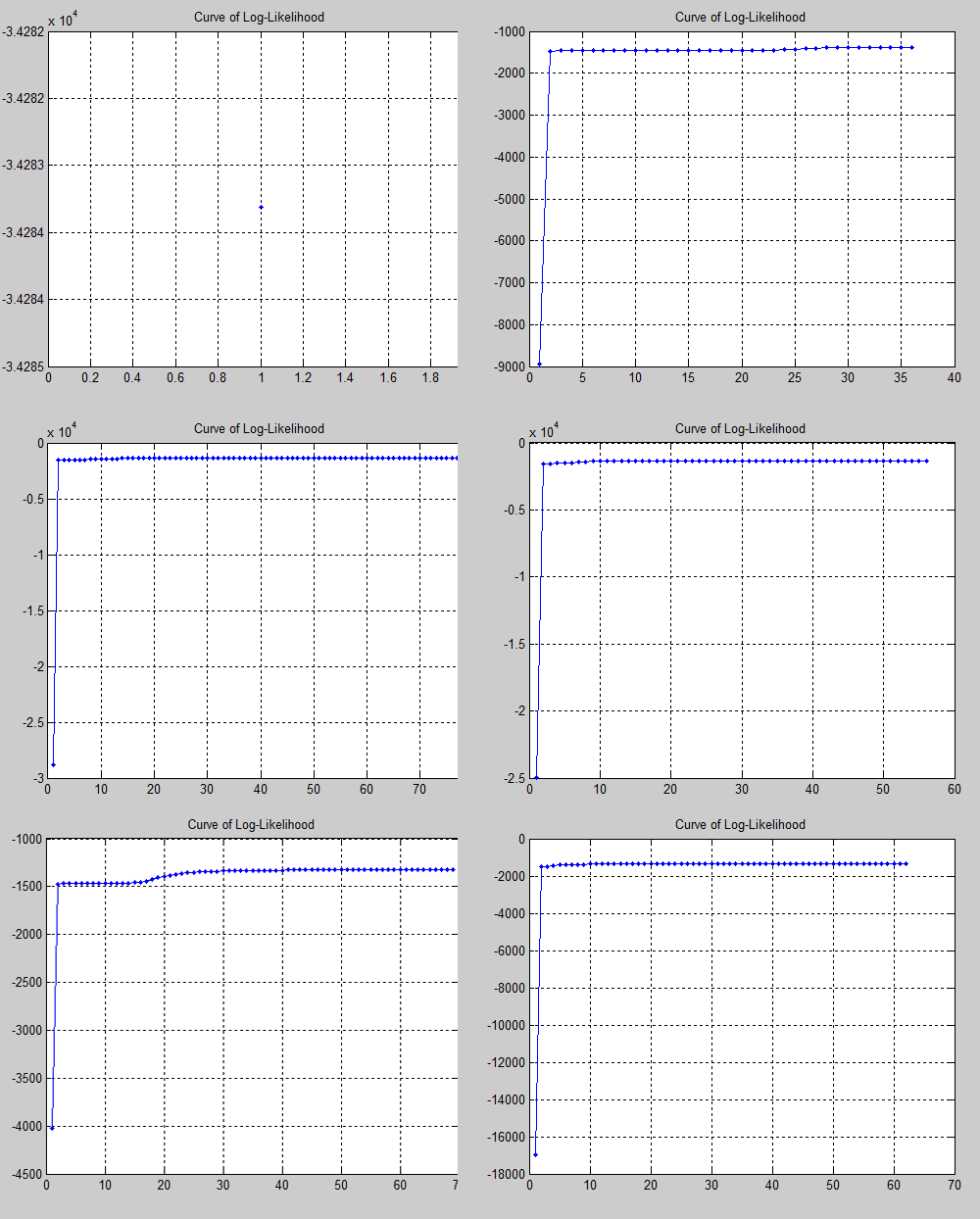
1. **Data set 2**



1. **Data set 3**

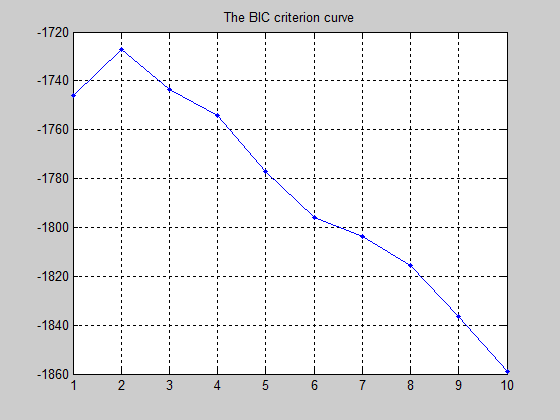


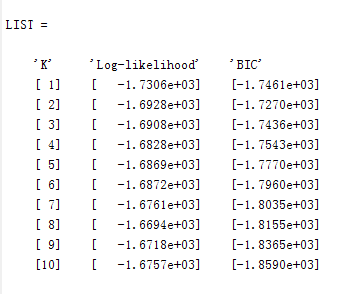
1. **Data set 4(K from 1 to 6)**



BIC MAXIMUM WHEN K=4

1. For each data set, generate a table of log-likelihood and BIC scores for K going from K=1 to some maximum value. Comment briefly on the results.
2. **Data set 1**



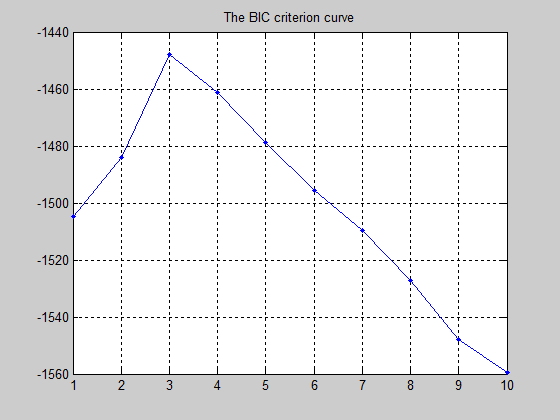


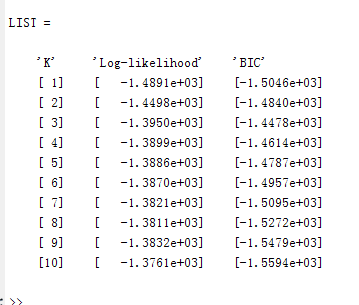




Comment: From the Graph we can get that the BIC criterion prevents the log-likelihood from being increasing with K. And the largest BIC score point is when K equal to 2, which means the best way to cluster the data set 1 is to separate them into 2 different classes.

1. **Data set 2**



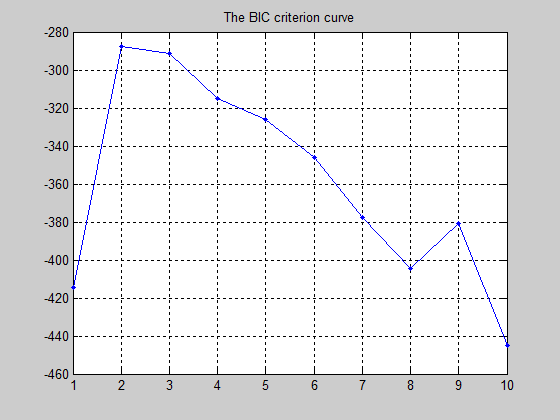


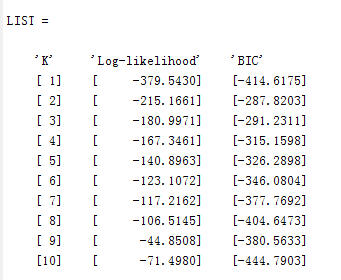




Comment: From the Graph we can get that the BIC criterion prevents the log-likelihood from being increasing with K. And the largest BIC score point is when K equal to 3, which means the best way to cluster the data set 2 is to separate them into 3 different classes.

1. **Data set 3**



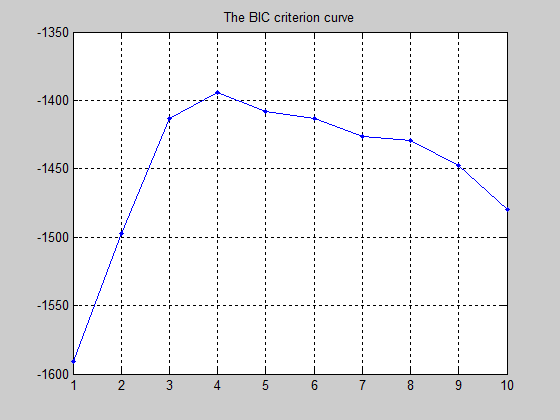


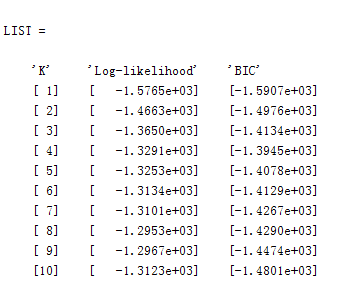




Comment: From the Graph we can get that the BIC criterion prevents the log-likelihood from being increasing with K. However, because the data set 3 might not be a mixture Gaussian data sets, the biggest BIC score point is located in K=2, Which is a little bit different with the true K (K=3). And it also tells us that the best way to cluster the data set 1 is to separate them into 2 different classes.

1. **Data set 4**









Comment: From the Graph we can get that the BIC criterion prevents the log-likelihood from being increasing with K. Because we haven’t got any labels in the original data set, and we cannot decided what kinds of clustering is the best. But from the BIC criterion we can get the biggest BIC score point is when K equal to 4, which means the best way to cluster the data set 4 is to separate them into 4 different classes.

1. **Appendix of the MALTAB codes**
2. **Algorithm 1: K-MEANS Clustering**

**function[]=k\_mean\_clustering(r,K,mdata)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the main function of running the K-Mean-Clustering Algorithm. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[observations,variables]=size(mdata); % Measure the size of data**

**for i=1:variables-1; % Clean the labels of original data**

**data(:,i)=mdata(:,i);**

**end**

**figure; % Plot the Original data set**

**plot(data(:,1),data(:,2),'b\*');**

**hold on;**

**for i=1:r % Define a r times loop that the**

**% total algorithm rums**

**[randmu,dimension]=mean\_initialization(data);**

**% Recall Mean\_initialization function to measure the range of data**

**times=0;n=1;mx=1;my=1; % Initialized some parameters**

**while mx<=K**

**%%%%%%%%%%% Initialized-the-first-K-points-randomly-Process %%%%%%%%%%**

**while n<=dimension**

**u(mx,my)=(randmu(2\*n-1)-randmu(2\*n))\*rand(1,1)+randmu(2\*n);**

**n=n+1;**

**my=my+1;**

**end**

**mx=mx+1;**

**n=1;**

**my=1;**

**end % Initialization processing end**

**mx=1; % Reset some parameters**

**a=1;**

**%%%%%%%%% Enter the main loop to find out the final K mean points %%%%%%%%%**

**while 1**

**times=times+1; % Total times that get u-convergence**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%% Find-the-Shortest-Distance-Process %%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for dx=1:observations % For every data row**

**while a<=K**

**X=[data(dx,:);u(a,:)];**

**d(a)=pdist(X); % Computed the Euclidean distance**

**a=a+1;**

**end**

**[minimum,classnum]=min(d); % Find out the shortest distance**

**sorted\_data(dx,:)=[data(dx,:),classnum];% Labeled data**

**d=[]; % Clean the parameter matrix**

**a=1; % Reset Parameter**

**end % End process**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%% Updated-K-mean-points-Process %%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**sorted\_data=sortrows(sorted\_data,variables);**

**% Sorted new data matrix by new labels**

**exu=u;t\_exu=transform(u);**

**u=data\_update(sorted\_data,K); % Recall function**

**t\_u=transform(u);**

**%%%%%%%%%%%%%%%%%%%%% Sum-Squared-Error-Process %%%%%%%%%%%%%%%%%%%%%**

**e=sum\_squared\_error(times,K,u,sorted\_data);**

**err(times,:)=e;**

**%%%%%%%%%%%%%%%%%%%%%%%% Reach Convergence? %%%%%%%%%%%%%%%%%%%%%%%%%%**

**if(pdist2(t\_exu,t\_u)<=0.000001)**

**%%%%%%%%%%%%%%%%%%%% Plot with different clusters %%%%%%%%%%%%%%%%%%%%%%**

**saperation\_plot(sorted\_data,K,u);**

**break; % Break the main loop**

**end**

**end % If we haven't reach the convergence, we do the it again**

**times=0; % Reset parameter**

**%%%%%%%%%%%%%%%%%%%%%%%%% Plot SSE Curve Part %%%%%%%%%%%%%%%%%%%%%%%%%**

**figure**

**plot(err(:,1),err(:,2),'b-');**

**hold on;**

**plot(err(:,1),err(:,2),'b.','MarkerSize',10);**

**hold on;**

**title('The Sum Squared Error Curve');**

**hold on;**

**grid on;**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**end % End a full time algorithm**

**end % End function**

**function[ui,dim]=mean\_initialization(data)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of finding out the ranges of the data. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data);**

**for i=1:colomn**

**u(2\*i-1)=round(max(data(:,i))); % Put the maximum number in each odd item**

**u(2\*i)=round(min(data(:,i))); % Put the maximum number in each even item**

**end**

**ui=u; % Output the mu vector**

**dim=colomn; % Output the dimension of data**

**end**

**function [u]=data\_update(data,k)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of computing the new K mean points. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data);**

**sum\_data=zeros(1,colomn); % Define a vector to compute the summation**

**num=0;**

**for i=1:k**

**for j=1:row**

**if length(data)~=0**

**if data(j,colomn)==i % Added the data for each cluster**

**sum\_data(1,:)=sum\_data(1,:)+data(j,:);**

**num=num+1; % Compute the data # of each cluster**

**end**

**end**

**end**

**sum\_data(:,colomn)=[];**

**umean=sum\_data/num; % Generate a new K mean points**

**sum\_data=zeros(1,colomn); % Reset the zero-vector**

**for x=1:num % Clean the same cluster data**

**data(1,:)=[]; % in order to computer next cluster**

**end % much more easily**

**num=0;**

**[row,colomn]=size(data);**

**u(i,:)=umean; % Updated the K mean points**

**end % End function**

**function[a]=transform(b)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of transform the matrix forms. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(b);**

**a=[]; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:row % Change the matrix forms as below: %**

**a=[a,b(i,:)]; % From: [\*\*\*\*\*\*] | To: [\*\*\*\*\*\*, ######] %**

**end % [######] | %**

**end %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**function[e]=sum\_squared\_error(times,k,u,data)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of computing the new K mean points. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data);**

**classes=cell(1,k);a=1;error=0;**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%% Process of Separating data from different clusters %%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:k**

**while data(1,colomn)==i % Put the data which belong the**

**% same cluster into one matrix**

**C(a,:)=data(1,:);**

**a=a+1;**

**data(1,:)=[]; % Clean the current row data**

**if isempty(data) % Whether the matrix still exist**

**break; % Break the loop**

**end**

**end**

**C(:,colomn)=[]; % Clean the label**

**classes{i}=C; % Put the matrix into cell**

**C=[]; % Reset parameters**

**a=1;**

**end**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%% Sum Squared Error Computation %%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:k**

**X=classes{i};sum=0;**

**while 1**

**sum=sum+(norm((X(1,:)-u(i,:))))^2; % Computed ||yi-mi||^2**

**X(1,:)=[];**

**if isempty(X)**

**break; % Break the loop**

**end**

**end**

**error=error+sum; % Computed the summation of error**

**end**

**e=[times,error/row]; % Get the final SSES**

**function saperation\_plot(data,k,u)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of plotting the data in different clusters. %**

**% Start Date: 2013-3-4 Finished Date: 2013-3-6 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data);**

**if colomn>=4 % If the data is multi-dimensions£¬just**

**data=[data(:,3),data(:,4),data(:,colomn)]; % plot first 2 dimensions.**

**[row,colomn]=size(data);**

**end**

**classes=cell(1,k);a=1; % Define a store space**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%% Process of Separating data from different clusters %%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:k**

**while data(1,colomn)==i % Put the data which belong the**

**C(a,:)=data(1,:); % same cluster into one matrix**

**a=a+1;**

**data(1,:)=[]; % Clean the current row data**

**if isempty(data) % Whether the matrix still exist**

**break; % Break the loop**

**end**

**end**

**C(:,colomn)=[]; % Clean the label**

**classes{i}=C; % Put the matrix into cell**

**C=[]; % Reset parameters**

**a=1;**

**end**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot Part %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**figure**

**for i=1:k**

**CLA=classes{i}; % Plot different cluster in deferent color**

**if i==1;**

**plot(CLA(:,1),CLA(:,2),'rd');**

**hold on;**

**elseif i==2**

**plot(CLA(:,1),CLA(:,2),'b\*');**

**hold on;**

**elseif i==3**

**plot(CLA(:,1),CLA(:,2),'ko');**

**hold on;**

**elseif i==4**

**plot(CLA(:,1),CLA(:,2),'m+');**

**hold on;**

**elseif i==5**

**plot(CLA(:,1),CLA(:,2),'gp');**

**hold on;**

**else**

**plot(CLA(:,1),CLA(:,2),'ch');**

**hold on**

**end**

**legend('class1','class2','class3','class4','class5','class6');**

**hold on;**

**end**

**plot(u(:,1),u(:,2),'k.','MarkerSize',35,'MarkerFaceColor','k');**

**hold on; % Plot the final K mean points**

**title('The Data Distribution Graph');**

**hold on;**

**end**

1. **Algorithm 2: Gaussian Mixture Clustering**

**function[CC]=Gauss\_Mix\_Clustering(r,k,mdata)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the main function of running the EM Algorithm in mixture data %**

**% Start Date: 2013-3-7 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(mdata); % Measured the size of the input data**

**mu=cell(k,1); % Defined some store space**

**sigma=cell(k,1);**

**for i=1:colomn-1; % Cleaned the original labels**

**data(:,i)=mdata(:,i);**

**end**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%% Loop of R times that the total algorithm runs %%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:r**

**times=0; % Initialized parameter**

**figure**

**for j=1:k % Define k clusters ( Initialized part)**

**mu{j}=data(round((row-1)\*rand(1,1)+1),:);**

**% Picked up k points from original data randomly**

**sigma{j}=eye(colomn-1);**

**% Generated D dimensions identity matrix**

**p(j)=1/k; % Initialized the probability**

**tiny=0.00001\*fliplr(eye(colomn-1));**

**plot\_gauss(data,mu{j},(sigma{j}+tiny),3,4);**

**hold on;**

**title('The Initial Parameter Values Plot');**

**end**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%% Main loop part which contain E-step & M-step %%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**while 1**

**matrixz=Estep(k,p,mu,sigma,data);**

**% Recall the Estep function to generate the matrix E[Zij]**

**[np,nmu,nsigma]=Mstep(k,matrixz,data);**

**% Recall the Mstep function to compute the new parameters**

**A=log\_likelihood(k,p,mu,sigma,data);**

**% Recall the sub function to calculate the old log-likelihood**

**B=log\_likelihood(k,np,nmu,nsigma,data);**

**% Recall the sub function to calculate the new log-likelihood**

**times=times+1; % Add the times that the E,M-steps run**

**if (abs(B-A)<0.001)||(times==80)**

**% Judgment condition (Convergence)**

**CC=B; % Output the final log-likelihood**

**break;**

**else % If we haven't got convergence**

**p=np;mu=nmu;sigma=nsigma; % Update the parameters**

**LOG(times,:)=[times,A]; % Generate a table of log-likelihood**

**end % End judgment**

**end**

**figure % Plot the different Gaussian**

**for j=1:k % regions with different clusters**

**plot\_gauss(data,nmu{j},nsigma{j},3,4);**

**hold on;**

**end % End plot**

**%%%%%%%%%%%%%%%% Plot the original data distribution %%%%%%%%%%%%%%%%**

**figure**

**pdata=sortrows(mdata,colomn); % Plot the original**

**s\_plot(pdata,k); % data distributions**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**figure % Plot the Log-likelihood curve**

**axis([0 60 -1000 -700]);**

**plot(LOG(:,1),LOG(:,2),'b-')**

**hold on;**

**plot(LOG(:,1),LOG(:,2),'b.-','MarkerSize',6)**

**hold on;**

**grid on;**

**title('Curve of Log-Likelihood');**

**end % End a full algorithm**

**end % End function**

**function[Z]=Estep(k,p,mu,sigma,data)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of running E-step in the EM algorithm %**

**% Start Date: 2013-3-7 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data); % Measured the size of the input data**

**denominator=0; % Initialized the denominator of E[Zij]**

**for i=1:row**

**for j=1:k % Computed the denominator of E[Zij]**

**denominator=denominator+p(j)\*1/((2\*pi)^(colomn/2)\*...**

**sqrt(det(sigma{j})))\*exp((-1/2)\*(data(i,:)-mu{j})\*...**

**inv(sigma{j})\*(data(i,:)-mu{j})');**

**end**

**for j=1:k % Computed the numerator of E[Zij]**

**numerator=p(j)\*1/((2\*pi)^(colomn/2)\*sqrt(det(sigma{j})))\*...**

**exp((-1/2)\*(data(i,:)-mu{j})\*inv(sigma{j})\*(data(i,:)-mu{j})');**

**Z(i,j)=numerator/denominator;**

**end**

**denominator=0; % Clean the denominator for next computation**

**end**

**end**

**function[p,mu,sigma]=Mstep(k,z,data)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of running M-step in the EM algorithm %**

**% Start Date: 2013-3-7 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[drow,dcolomn]=size(data); % Measure the size of data**

**[row,colomn]=size(z); % Measure the size of E[Zij] matrix**

**mu=cell(k,1); % Define some store space**

**sigma=cell(k,1);**

**p=zeros(1,k);u=zeros(k,dcolomn);sig=0\*eye(dcolomn); % Initialization**

**for j=1:k**

**for i=1:row % Calculated the new probability πij**

**p(j)=p(j)+z(i,j);**

**end**

**p(j)=p(j)/row;**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:row % Calculated the new mean mu**

**u(j,:)=u(j,:)+z(i,j)\*data(i,:);**

**end**

**u(j,:)=u(j,:)/(row\*p(j));**

**mu{j}=u(j,:);**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:row % Calculated the new covariance matrix**

**sig=sig+z(i,j)\*(data(i,:)-mu{j})'\*(data(i,:)-mu{j});**

**end**

**sig=sig/(row\*p(j));**

**sigma{j}=sig;**

**sig=0\*eye(dcolomn); % Reset the parameter**

**end**

**end % End function**

**function[L]=log\_likelihood(k,p,mu,sigma,data)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of calculating log-likelihood of EM algorithm %**

**% Start Date: 2013-3-7 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(data); % Measured the size of input data**

**L=0; % Initialized parameter**

**for i=1:row % Computed the log-likelihood**

**Sum=0;**

**for j=1:k % Computed the norm 2 square for each data**

**Sum=Sum+(p(j)\*1/((2\*pi)^(colomn/2)\*...**

**sqrt(det(sigma{j})))\*exp((-1/2)\*(data(i,:)-mu{j})\*...**

**inv(sigma{j})\*(data(i,:)-mu{j})'));**

**end**

**L=L+log(Sum); % Got the final log-likelihood**

**end**

**end % End function**

**function [classes]=s\_plot(data,k)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the sub function of calculating log-likelihood of EM algorithm %**

**% Start Date: 2013-3-7 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% [row,colomn]=size(data)**

**classes=cell(1,k);a=1; % Define a store space**

**for i=0:k-1**

**while data(1,colomn)==i % Put the data which belong the**

**C(a,:)=data(1,:); % same cluster into one matrix**

**a=a+1;**

**data(1,:)=[]; % Clean the current row data**

**if isempty(data) % Whether the matrix still exist**

**break; % Break the loop**

**end**

**end**

**C(:,colomn)=[]; % Clean the label**

**classes{i+1}=C; % Put the matrix into store space**

**C=[]; % Reset parameters**

**a=1;**

**end**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot Part %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**for i=1:k**

**CLA=classes{i}; % Plot different cluster in deferent color**

**if i==1;**

**plot(CLA(:,1),CLA(:,2),'r.');**

**hold on;**

**elseif i==2**

**plot(CLA(:,1),CLA(:,2),'b\*');**

**hold on;**

**elseif i==3**

**plot(CLA(:,1),CLA(:,2),'ko');**

**hold on;**

**elseif i==4**

**plot(CLA(:,1),CLA(:,2),'cp');**

**hold on;**

**else**

**plot(CLA(:,1),CLA(:,2),'g+');**

**hold on;**

**end**

**legend('class1','class2','class3','class4','others');**

**hold on;**

**title('Data distribution');**

**end**

**function plot\_gauss(data, mean, covar,xaxis,yaxis)**

**% PLOT\_GAUSS: plot\_gauss(data, mean, covar,xaxis,yaxis)**

**%**

**% MATLAB function to plot a 2 dimensional scatter plot of**

**% sample data (using xaxis and yaxis as the column indices into**

**% an N x d data matrix) and superpose the mean of a Gaussian**

**% model and its "covariance ellipse" on this data.**

**% ICS 274 Demo Function**

**%**

**% INPUTS:**

**% data: N x d matrix of d-dimensional feature vectors**

**% means: 1 x d matrix: the d-dimensional mean of the Gaussian model**

**% covar: d x d matrix: the dxd covariance matrix of the Gaussian model**

**% xaxis: an integer between 1 and d indicating which of the features is**

**% to be used as the x axis**

**% yaxis: another integer between 1 and d for the y axis**

**%figure**

**plot(data(:,xaxis),data(:,yaxis),'r.');**

**hold on**

**plot(mean(xaxis),mean(yaxis),'b\*','Markersize',10);**

**% Calculate contours for the 2d normals at Mahalanobis dist = constant**

**mhdist = 3;**

**% Extract the relevant dimensions from the ith component matrix**

**covar2d = [covar(xaxis,xaxis) covar(xaxis,yaxis); covar(yaxis,xaxis) covar(yaxis,yaxis)];**

**% Use some results from standard geometry to figure out the ellipse**

**% equations from the covariance matrix. Probably other ways to**

**% do this, e.g., finding the principal component directions, etc.**

**% See Fraleigh, p.431 for details on rotating the ellipse, etc**

**icov = inv(covar2d);**

**a = icov(1,1);**

**c = icov(2,2);**

**% we don't check if this is zero: which occasionally causes**

**% problems when we divide by it later! needs to be fixed.**

**b = icov(1,2)\*2;**

**theta = 0.5\*acot( (a-c)/b);**

**sc = sin(theta)\*cos(theta);**

**c2 = cos(theta)\*cos(theta);**

**s2 = sin(theta)\*sin(theta);**

**a1 = a\*c2 + b\*sc + c\*s2;**

**c1 = a\*s2 - b\*sc + c\*c2;**

**th= 0:2\*pi/100:2\*pi;**

**x1 = sqrt(mhdist/a1)\*cos(th);**

**y1 = sqrt(mhdist/c1)\*sin(th);**

**x = x1\*cos(theta) - y1\*sin(theta) + mean(xaxis);**

**y = x1\*sin(theta) + y1\*cos(theta) + mean(yaxis);**

**% plot the ellipse**

**plot(x,y,'g')**

**title('The Final Parameter Values Plot');**

1. **Algorithm 3: Choosing K for Gaussian Mixture Clustering**

**function[BICMAX,list]=K\_BIC(r,K,mdata)**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**% This is the main function of Calculating the BIC Score in algorithm 3 %**

**% Start Date: 2013-3-8 Finished Date: 2013-3-8 Due Date: 2013-3-18 %**

**% Programmer: Tingshen Yan Location: Northeastern University %**

**%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**

**[row,colomn]=size(mdata);**

**colomn=colomn-1;**

**list=cell(K,3);**

**list{1,1}='K';list{1,2}='Log-likelihood';list{1,3}='BIC';**

**for i=1:K % For clustering i clusters from 1 to k**

**list{i+1,2}=Gauss\_Mix\_Clustering(r,i,mdata);**

**% Runing algorithm 2 for different k to get different log-likelihood**

**list{i+1,1}=i; % Print out the current k clusters**

**list{i+1,3}=list{i+1,2}-(i-1+i\*colomn+colomn\*(colomn+1)/2\*i)/2\*log(row);**

**% Computing the BIC Score for each different k clusters**

**BIC(i,1)=list{i+1,1}; % Rebuilt a matrix in order to plot**

**BIC(i,2)=list{i+1,3}; % the BIC curve**

**end**

**[BICMAX,I]=max(BIC);**

**figure**

**plot(BIC(:,1),BIC(:,2),'b-');**

**hold on;**

**plot(BIC(:,1),BIC(:,2),'b.','MarkerSize',8);**

**hold on;**

**grid on;**

**title('The BIC criterion curve');**

**end**