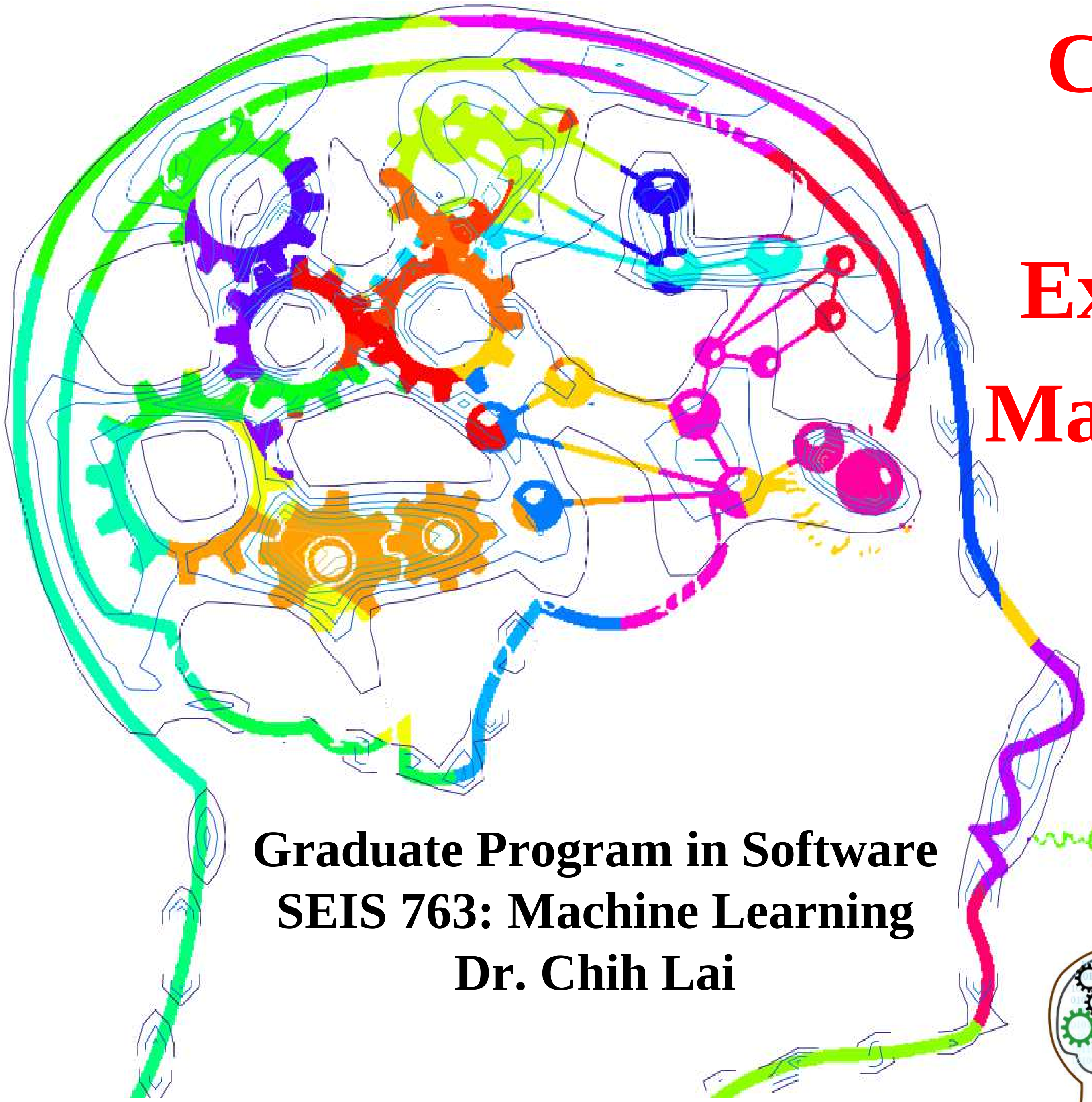
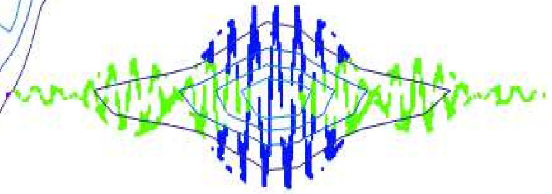
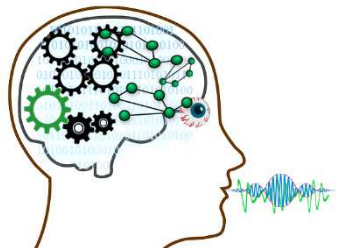


Clustering with Expectation Maximization



Graduate Program in Software
SEIS 763: Machine Learning
Dr. Chih Lai

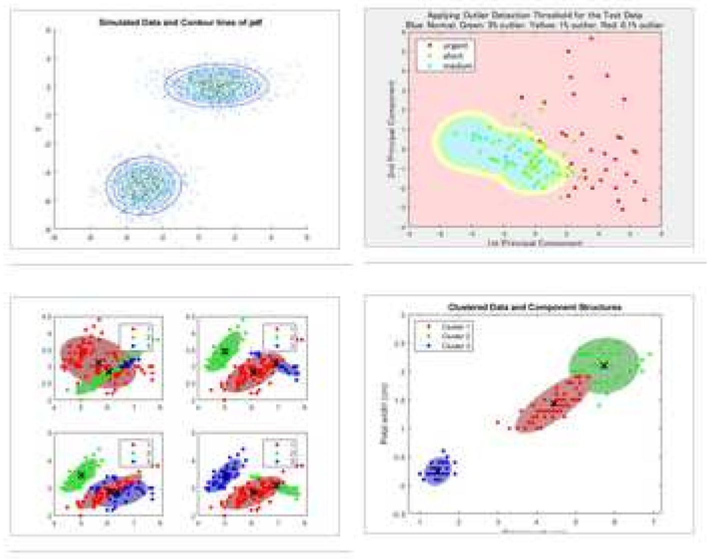


Outline

- Mean and covariance.
- Maximum likelihood, negative log likelihood.
- Gaussian Mixture Model (GMM).
- Expectation Maximization (EM) method.
  - EM regularization.
- EM vs. ***k***-means.
- Anomaly Detection using GMM and regularization.
- Other Anomaly Detection Issues.
  - Add predictors to separate outliers.
  - Make predictors become Gaussian.

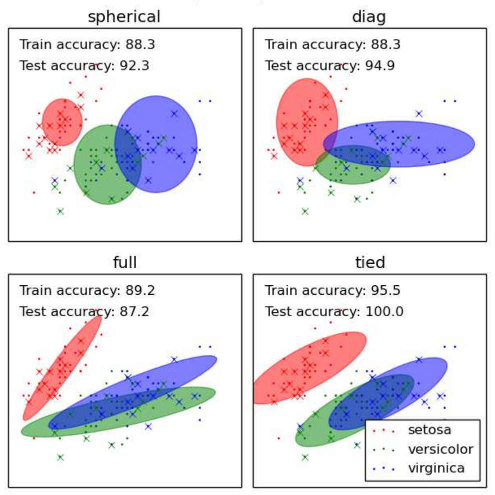
References

- Matlab
  - https://www.mathworks.com/help/stats/fitgmdist.html
  - https://www.mathworks.com/examples/search?q=fitgmdist

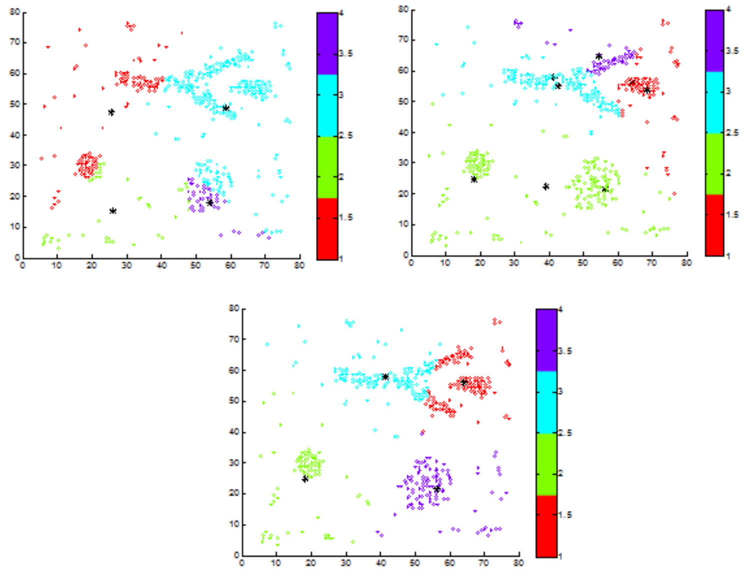


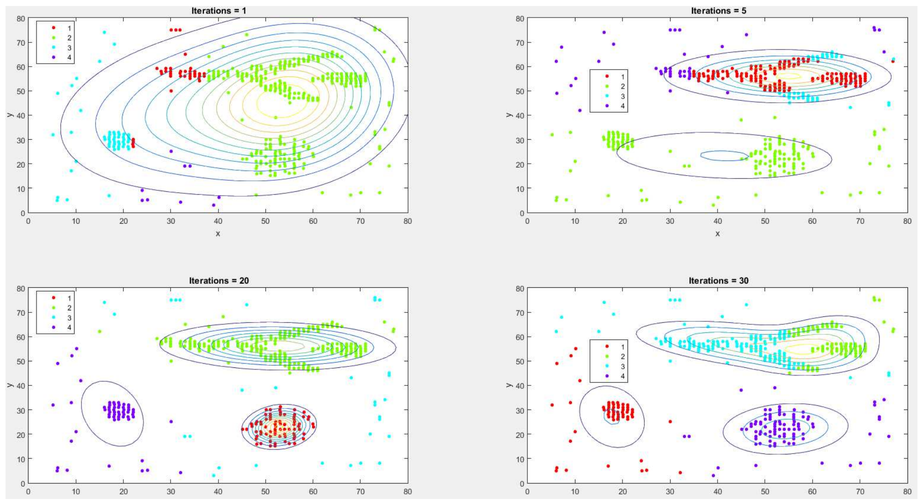
- sklearn
  - http://scikit-learn.org/stable/modules/mixture.html
  - http://scikit-learn.org/0.15/auto_examples/mixture/plot_gmm_classifier.html
  - http://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html

➢ **gmm = mixture.GaussianMixture(n_components=5, covariance_type='full').fit(X)**



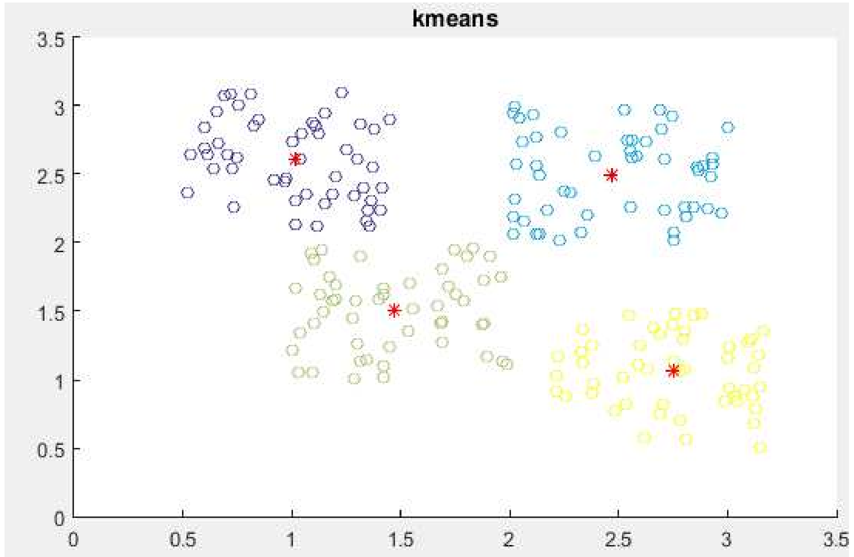
General Steps in *k*-means and GMM

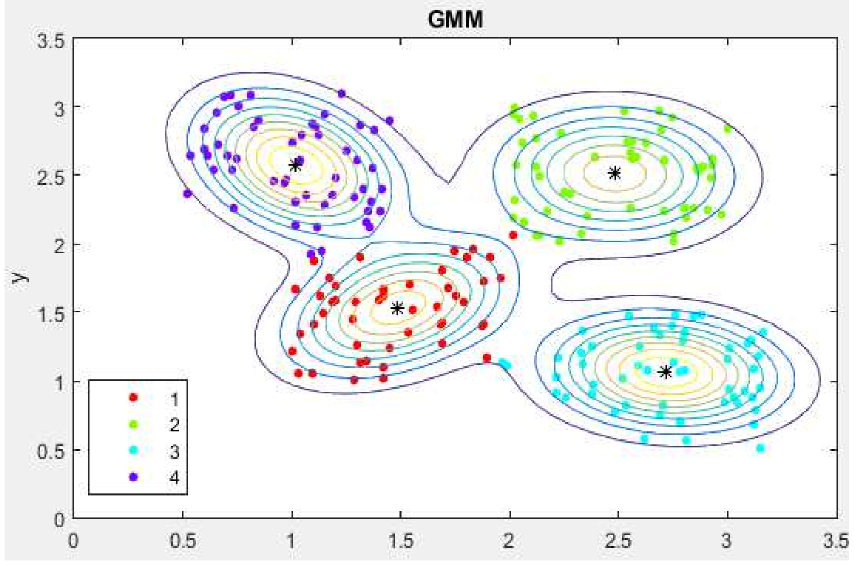


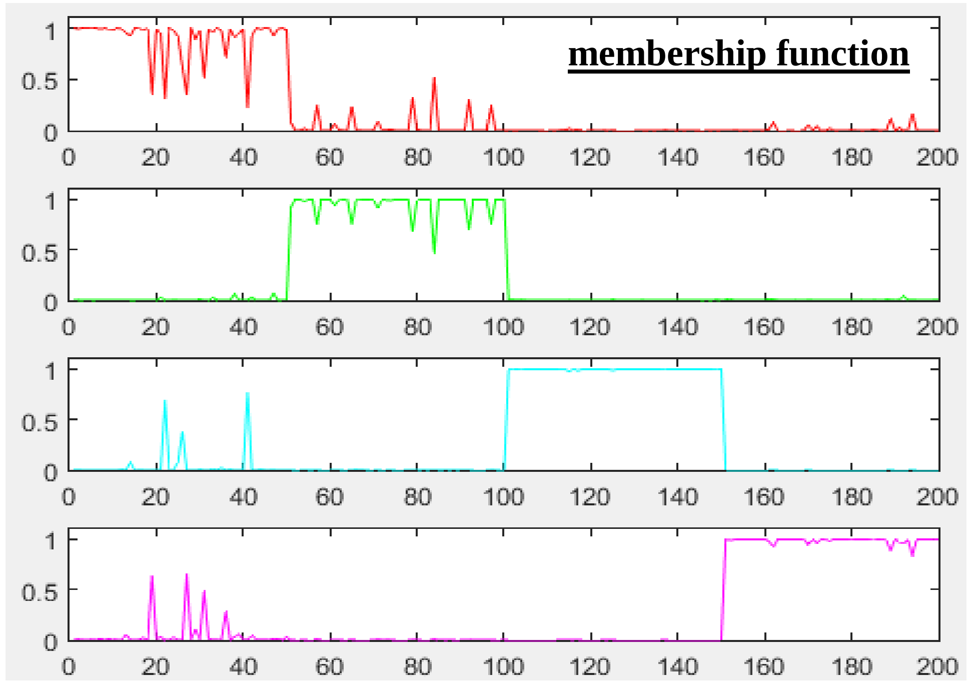


General Steps	**k-means**	**GMM**
1. initialization	pick random $\boldsymbol{k}$ points as centers $\boldsymbol{C_g}$ of each group $\boldsymbol{g}$	set $\boldsymbol{k}$ random $\boldsymbol{\mu_g}$ and $\boldsymbol{\Sigma_g}$ for each group $\boldsymbol{g}$
2. group assignment	assign each point to the closest center $\boldsymbol{C_g}$	assign each point to a group $\boldsymbol{g}$ with max probability computed against each $\boldsymbol{\mu_g}$ and $\boldsymbol{\Sigma_g}$
3. computation	re-compute $\boldsymbol{C_g}$ for $\boldsymbol{k}$ new groups based on the new assignment	re-compute $\boldsymbol{\mu_g}$ and $\boldsymbol{\Sigma_g}$ for $\boldsymbol{k}$ new groups based on the new assignment
4. iterations	go back to step 2	go back to step 2

GMM vs. ***k***-means



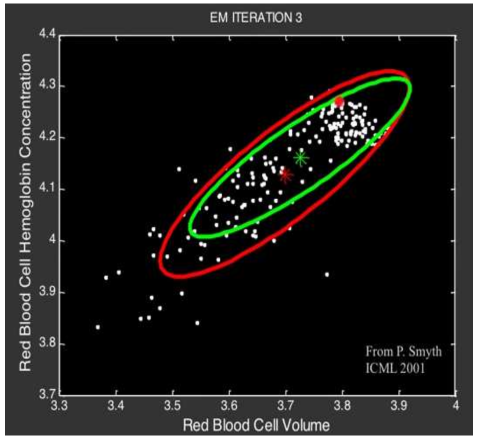


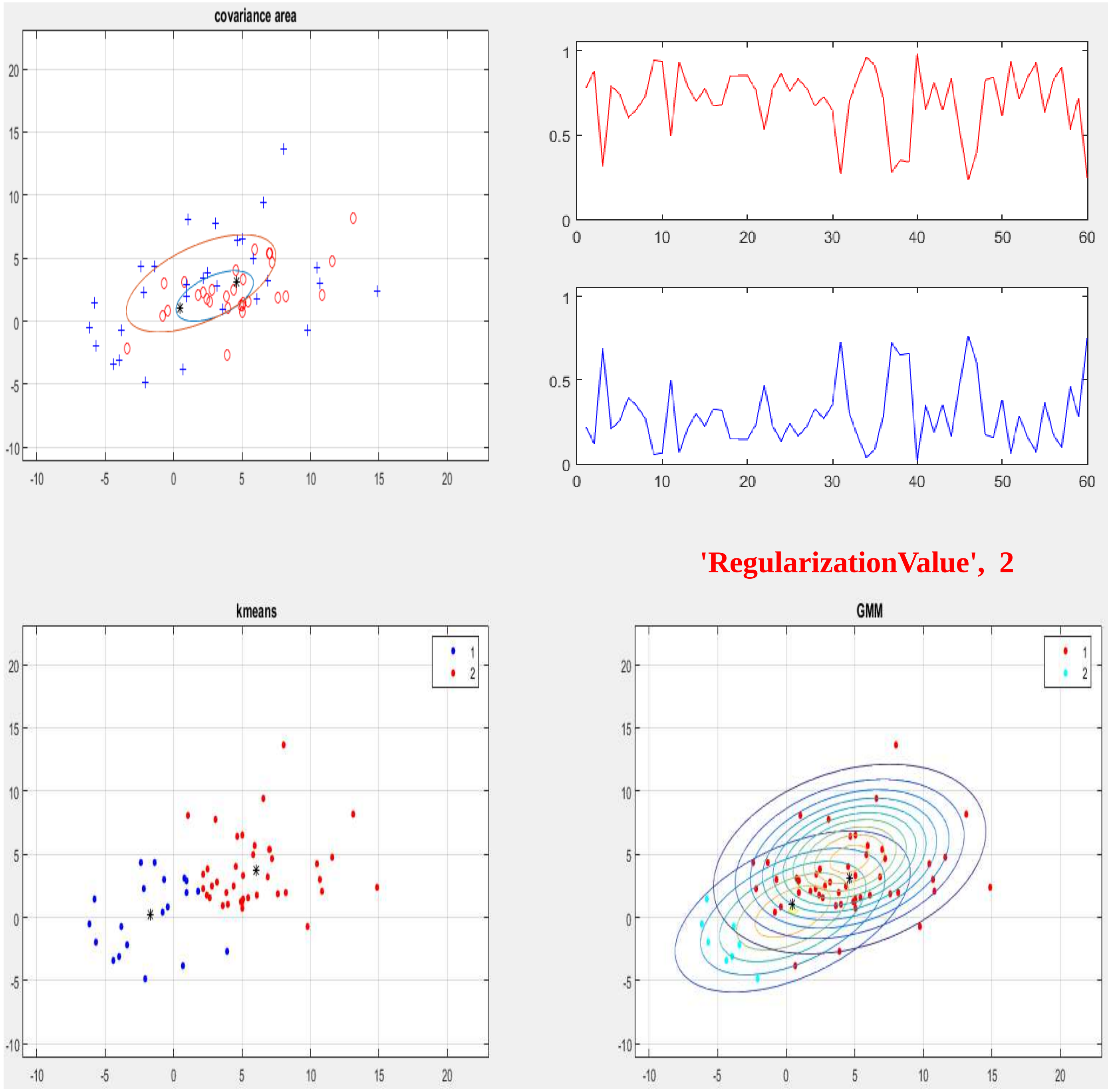


GMM on Overlapping Groups

with

Regularization



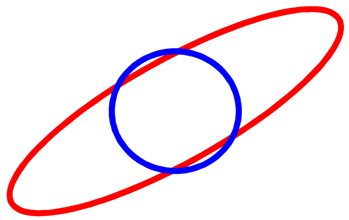


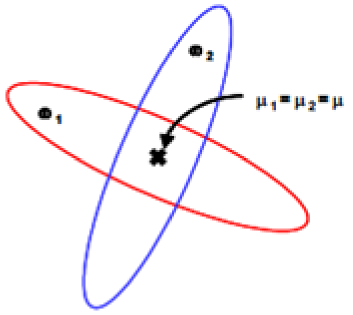
'RegularizationValue', 2

EM vs. ***K***-means

- Similar to ***K***-means.
  - Depends on starting components.
  - Keep changing means (centers).

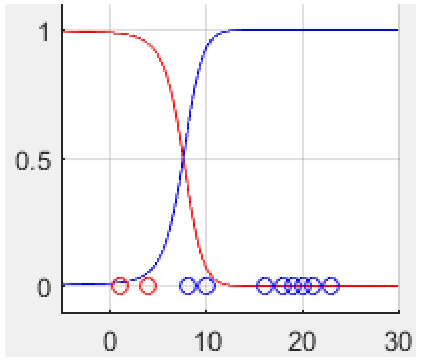
- Different from ***K***-means.
  - Soft clustering→ assign each point to a Gaussian component w/ probability.
  - One data point can belong to multiple components w/ probability.
  - Keep changing means (centers) and variance (or covariance).
  - K-means fail if non-sephere clusters center at the same locations.

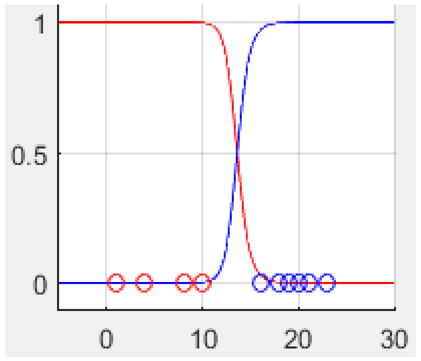


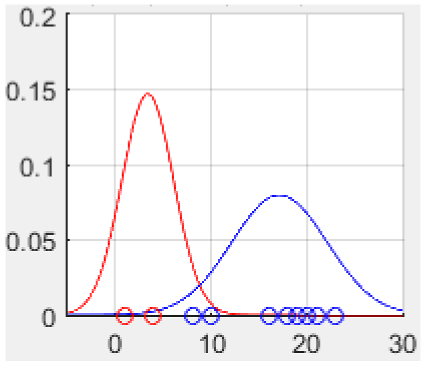


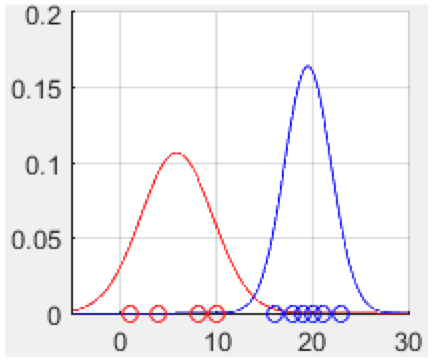
Maximum Likelihood

- How do we decide best cluster? maximize the ***joined probability*** of all points.
  - In other words, maximize the multiplications of probabilities from all points.
  - **maximize** $\boldsymbol{L} = \prod_{i=1}^{m} P_i$ (multiplications of probabilities from all points) **(“+”, bigger ≈ 1, better)**
  - Difficult to solve (chain rule), but we know log(a×b) = log(a) + log(b).
  - = **maximize** ***log-liklihood*** log($\boldsymbol{L}$) = $\log(\prod_{i=1}^{m} P_i) = \sum_{i=1}^{m} log(P_i)$ **(“—”, bigger ≈ 0, better)**
    - **NOTE: $0 \le P \le 1 \Rightarrow -\infty \le \log(P) \le 0$.**
  - **Same as to minimize *negative log likelihood.*** **(“+”, smaller ≈ 0, better)**


14 iterations
(0.5, 0.5, 10.5, 0.5)

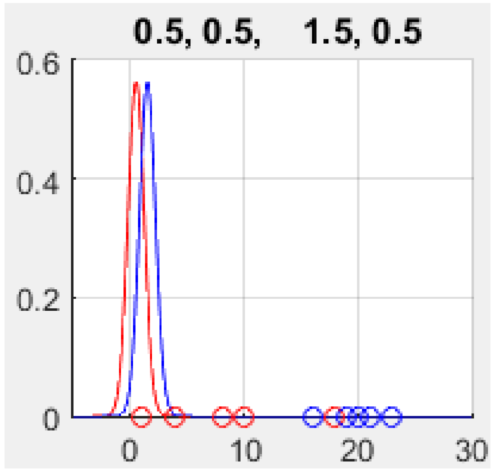

26 iterations
(0.5, 0.5, 10.5, 0.5)

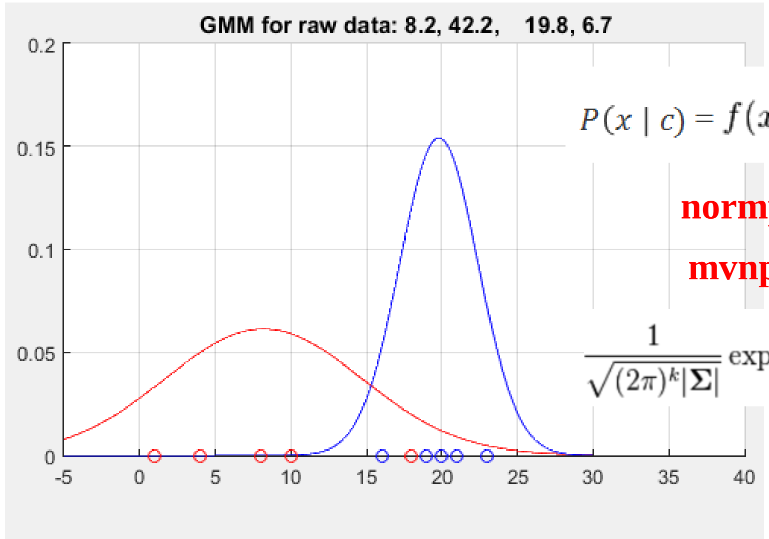

14 iterations
(0.5, 0.5, 10.5, 0.5)


26 iterations
(0.5, 0.5, 10.5, 0.5)

Maximize Log-Likelihood = Minimize Negative Log-Likelihood

- Maximize ***joined probability*** ➔ Maximize Log Likelihood ➔ Minimize Negative Log Likelihood.



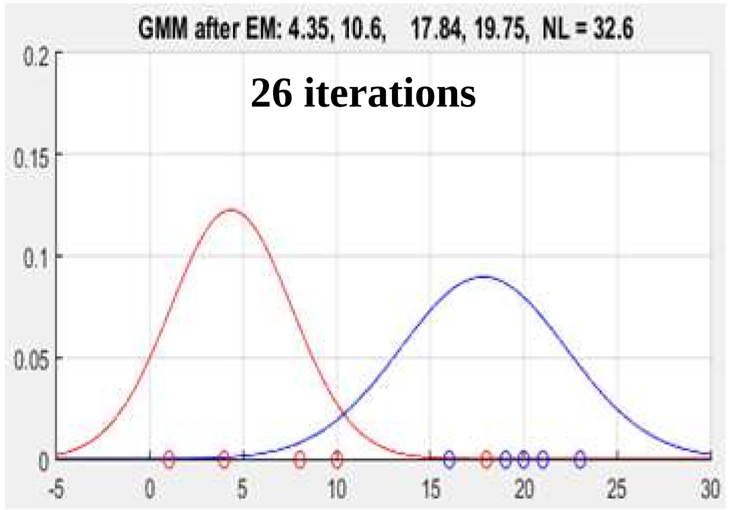


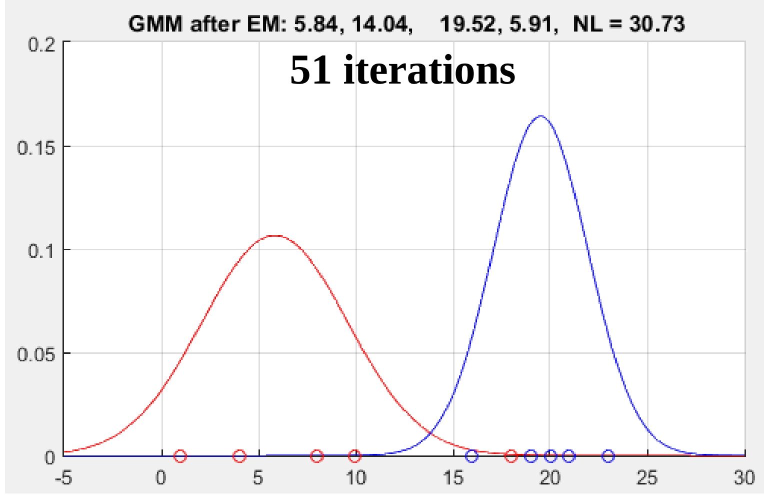
$$P(x \mid c) = f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

normpdf(data, ***μ***, ***σ***)

mvnpdf(data, ***μ***, Σ)

$$\frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

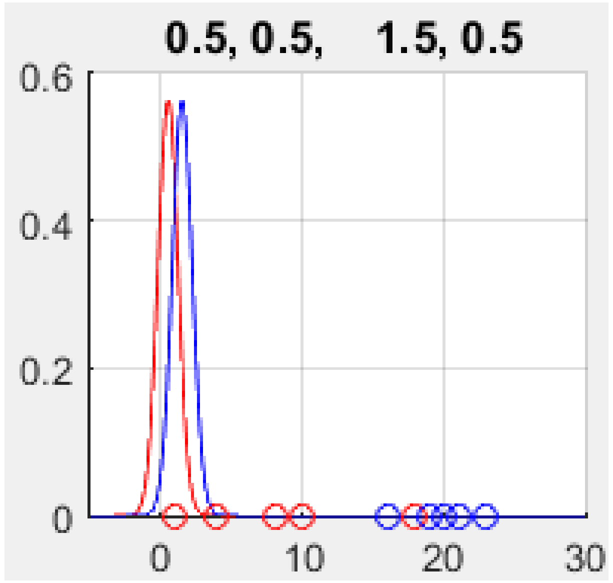


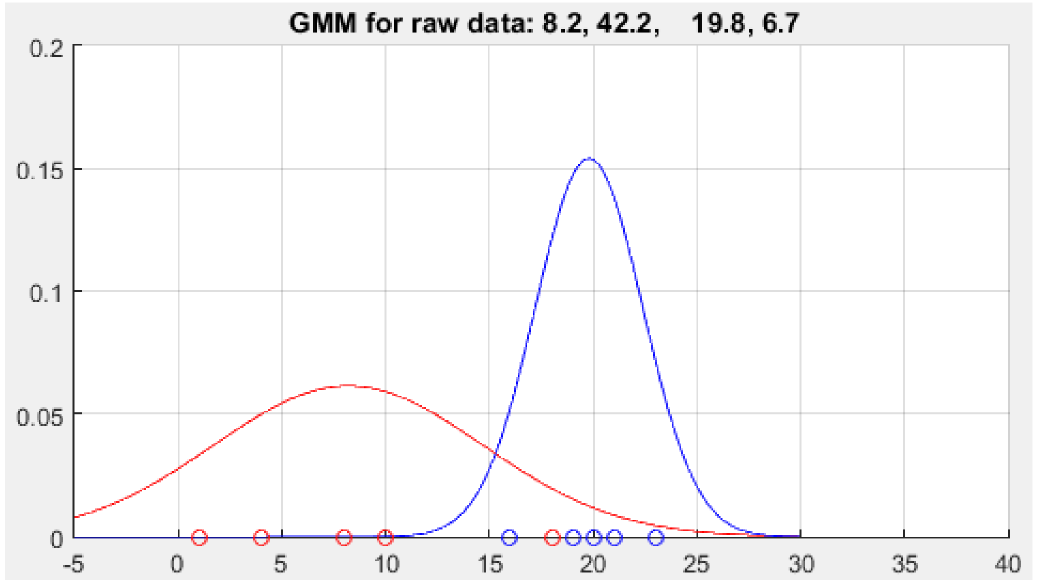


Gaussian Mixture Model (GMM)

- GMM is a probabilistic model that assumes data points were generated from…
  - A mixture of a finite # of latent Gaussian distributions (components).
  - Each component has unknown parameters ***μ*** and **Σ** (covariance).
  - Mixture models ≈ ***k***-means clustering (***centers***) w/ additional covariance of data.

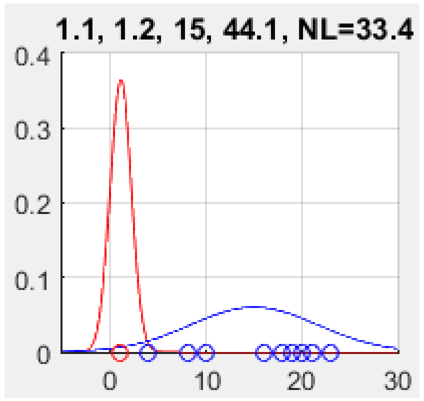
- ***Expectation-maximization*** (EM) fits mixture of ***k*** Gaussian models to data.
  - i.e. to estimate ***μ*** and **Σ** of each Gaussian model.

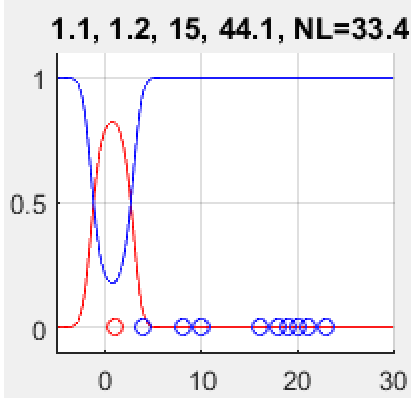


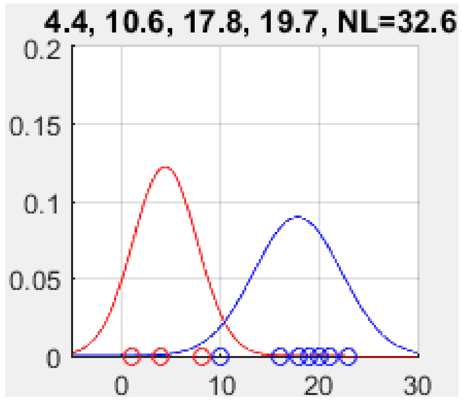


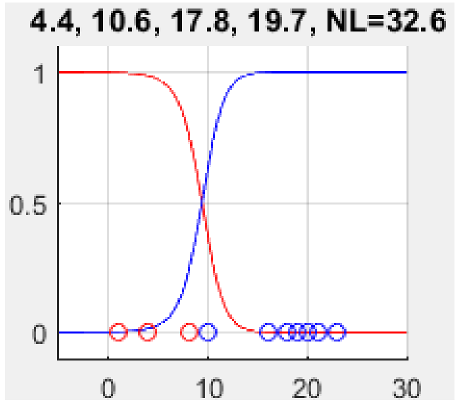
Iterate *E* and *M* until Convergence

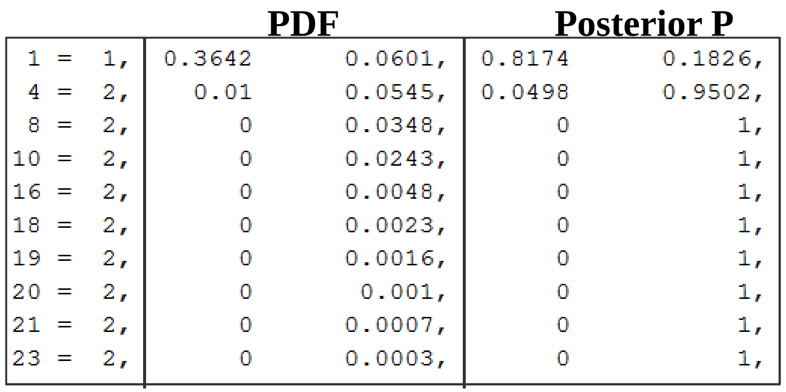
- **Randomly** set $\boldsymbol{\mu}_g$ & Σ_g to all GMM components.
- *E*xpectation: compute ***PDF*** of point ***x*** to all GMMs, & re-label ***x*** to GMM ***g*** based on max posterior probability. (i.e. does point ***x*** likely belong GMM$_g$?)
  - **normpdf**(***x***, ***μ***, *σ*) **posterior**(GMM_Model, ***x***)
- *M*aximization: adjust parameters $\boldsymbol{\mu}_g$ & Σ_g to minimize ***NLL*** (based on probability weights).



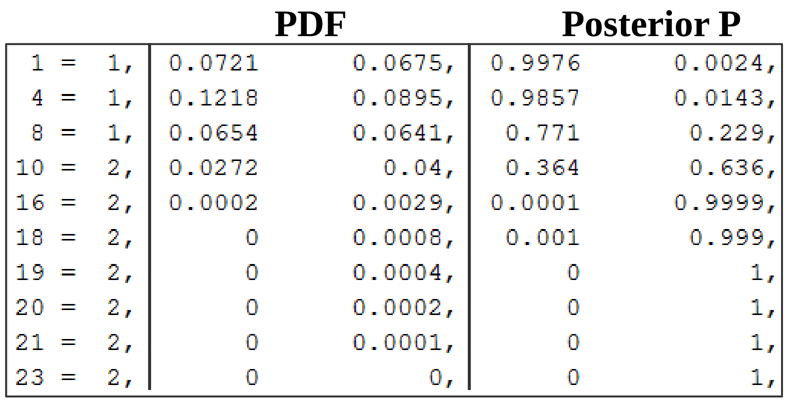








	PDF		Posterior P	
1 = 1,	0.3642	0.0601,	0.8174	0.1826,
4 = 2,	0.01	0.0545,	0.0498	0.9502,
8 = 2,	0	0.0348,	0	1,
10 = 2,	0	0.0243,	0	1,
16 = 2,	0	0.0048,	0	1,
18 = 2,	0	0.0023,	0	1,
19 = 2,	0	0.0016,	0	1,
20 = 2,	0	0.001,	0	1,
21 = 2,	0	0.0007,	0	1,
23 = 2,	0	0.0003,	0	1,

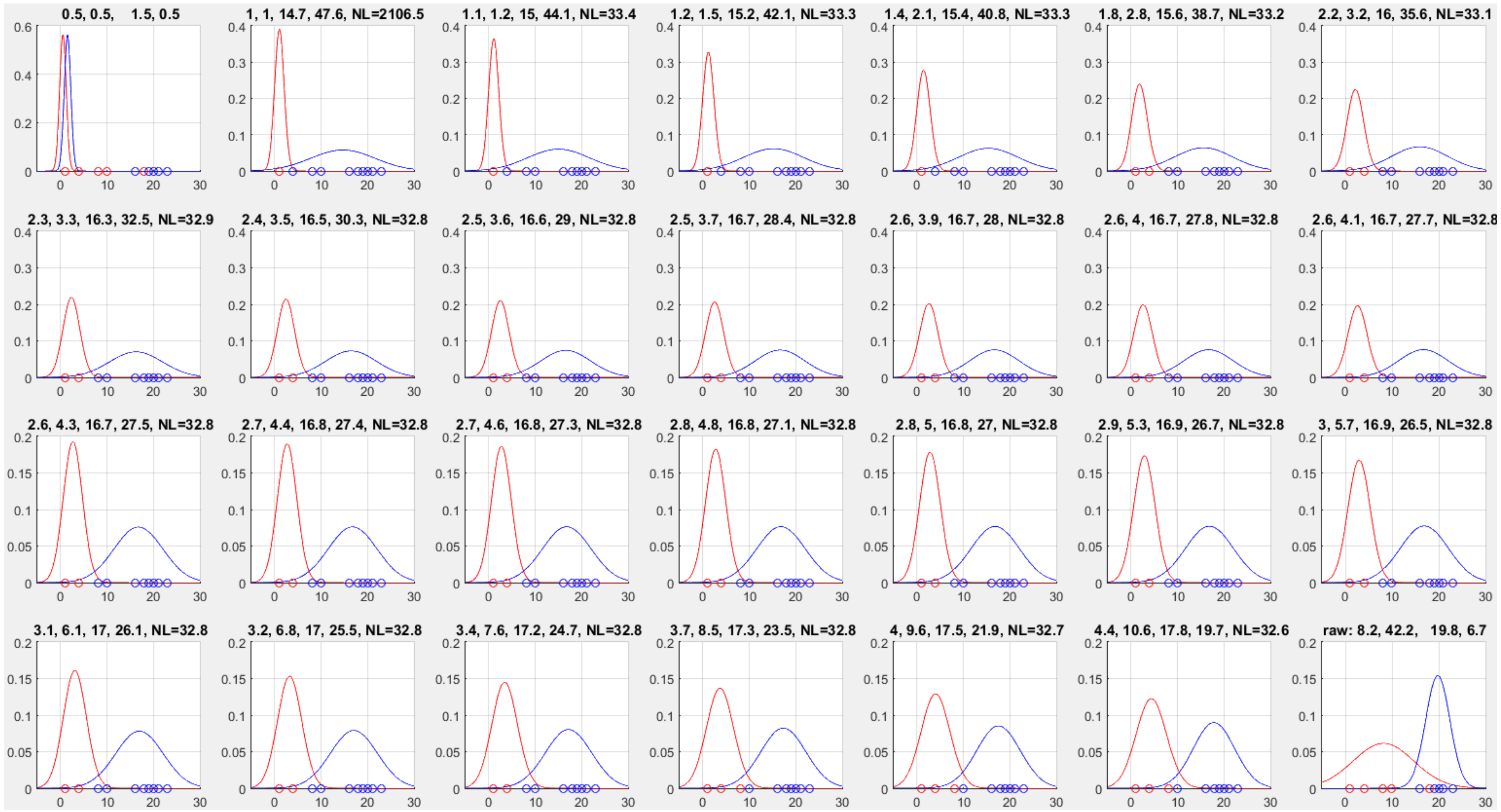


	PDF		Posterior P	
1 = 1,	0.0721	0.0675,	0.9976	0.0024,
4 = 1,	0.1218	0.0895,	0.9857	0.0143,
8 = 1,	0.0654	0.0641,	0.771	0.229,
10 = 2,	0.0272	0.04,	0.364	0.636,
16 = 2,	0.0002	0.0029,	0.0001	0.9999,
18 = 2,	0	0.0008,	0.001	0.999,
19 = 2,	0	0.0004,	0	1,
20 = 2,	0	0.0002,	0	1,
21 = 2,	0	0.0001,	0	1,
23 = 2,	0	0,	0	1,

EM Iteration 1 : 26 (PDF)

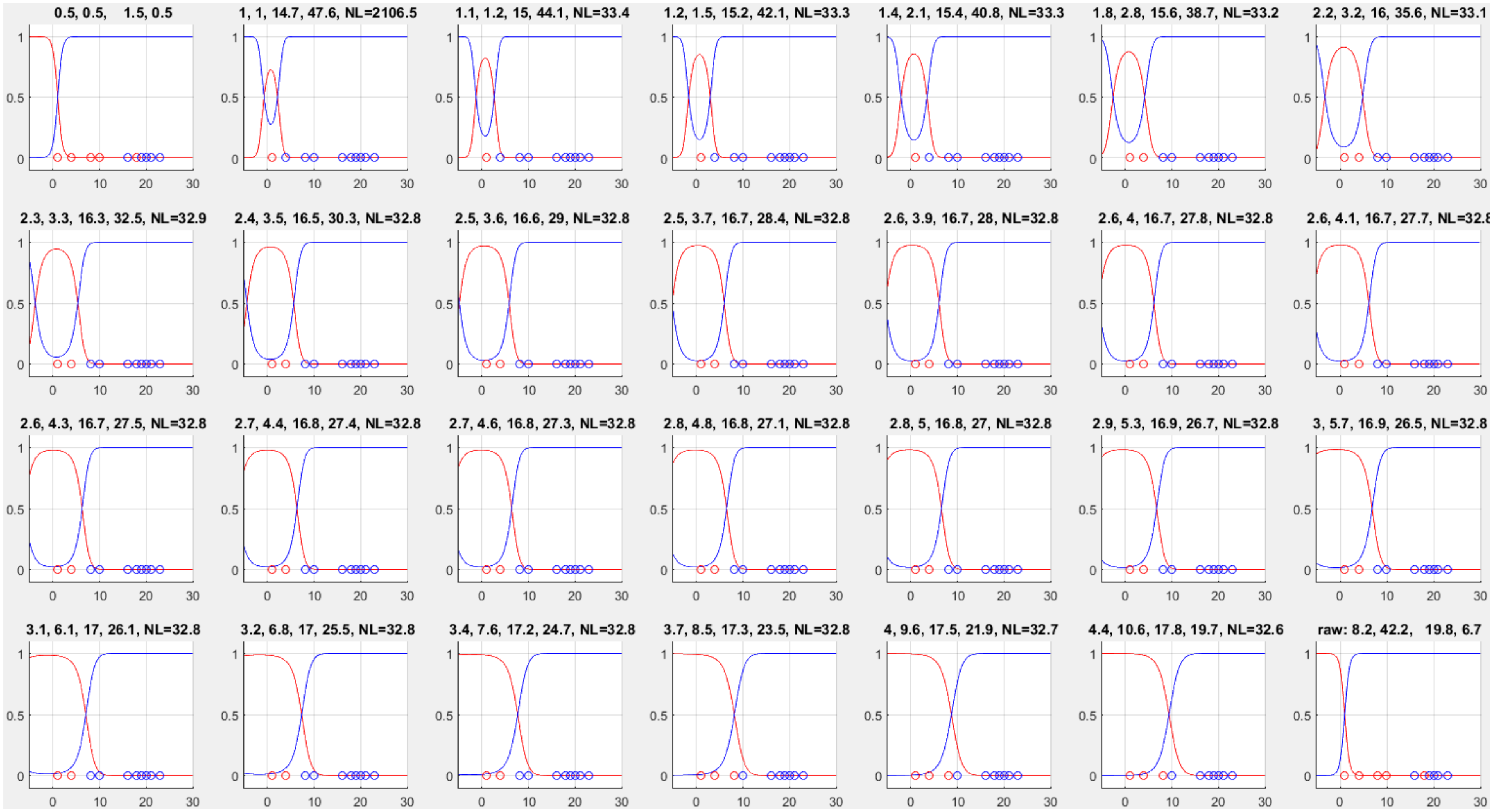
- ***Assume*** 2 GMMS (i.e. clusters).
  - Initialize $\mu_R = 0.5$, $\Sigma_R = 0.5$,
    $\mu_B = 1.5$, $\Sigma_R = 0.5$

```
GMM = fitgmdist(data, K, 'Start', S, 'RegularizationValue', 1);
% GMM.NlogL;
[idx, NLL, PostP, logpdf] = cluster(GMM, data);
```

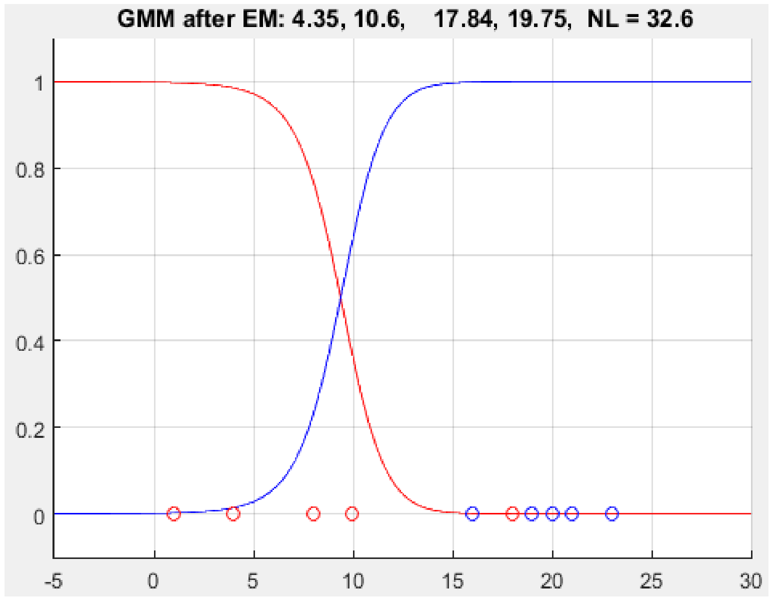


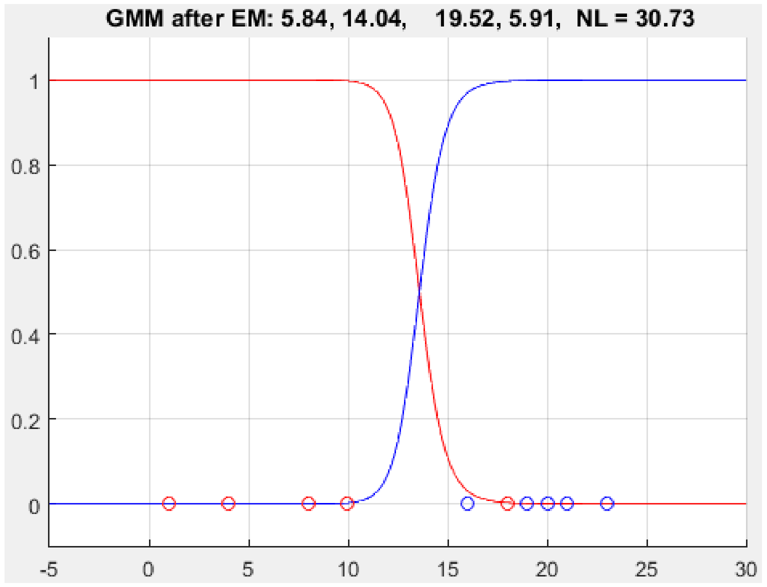
[23 2; 21 2; 20 2; 19 2; 18 1; 16 2; 10 1; 8 1; 4 1; 1 1]

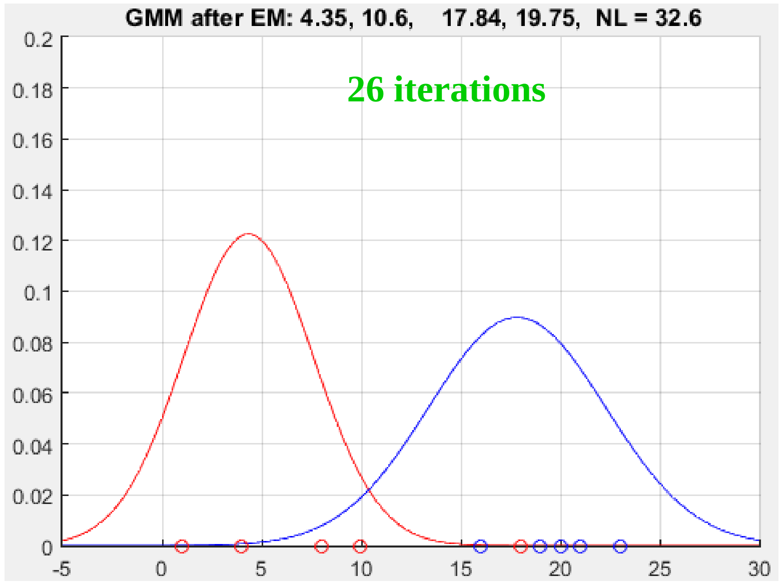
EM Iteration 1 : 26 (Posterior Probability)

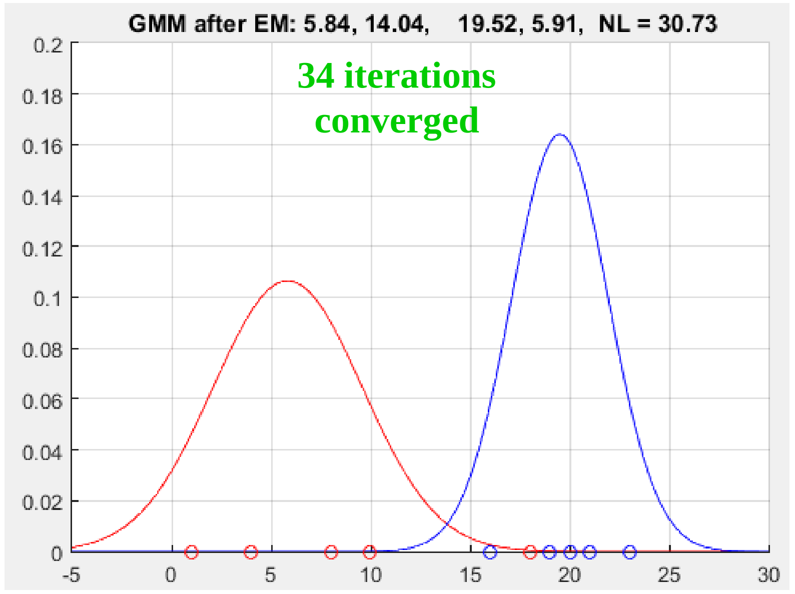


Converge at about 34 iterations

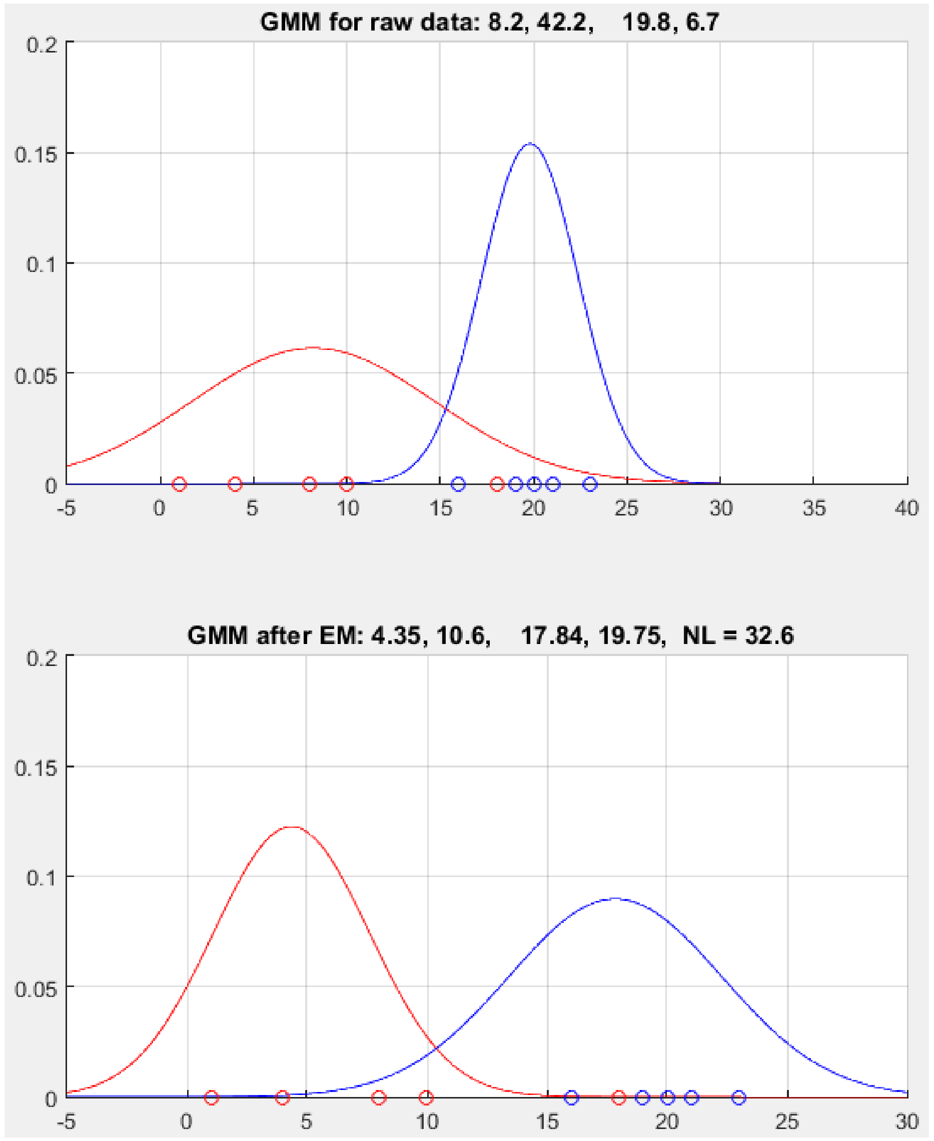


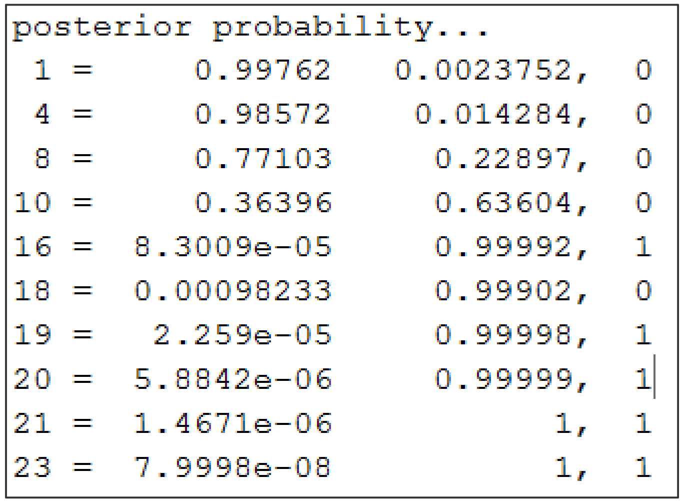




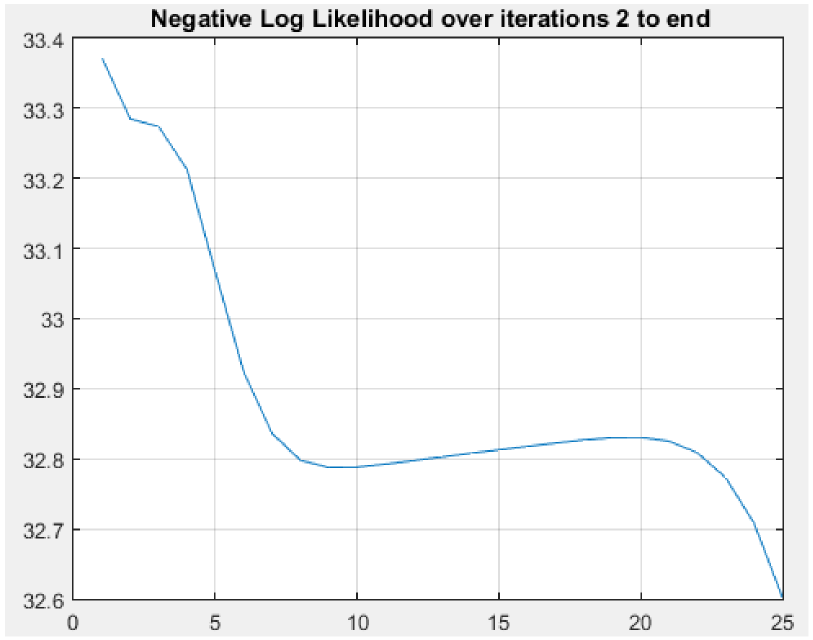


Negative Log Likelihood, After 26 Iterations

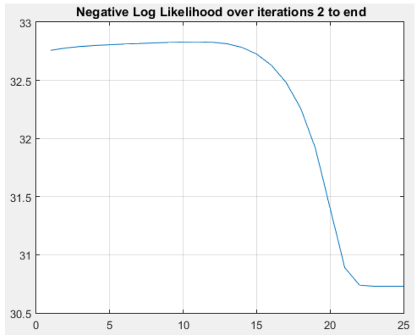


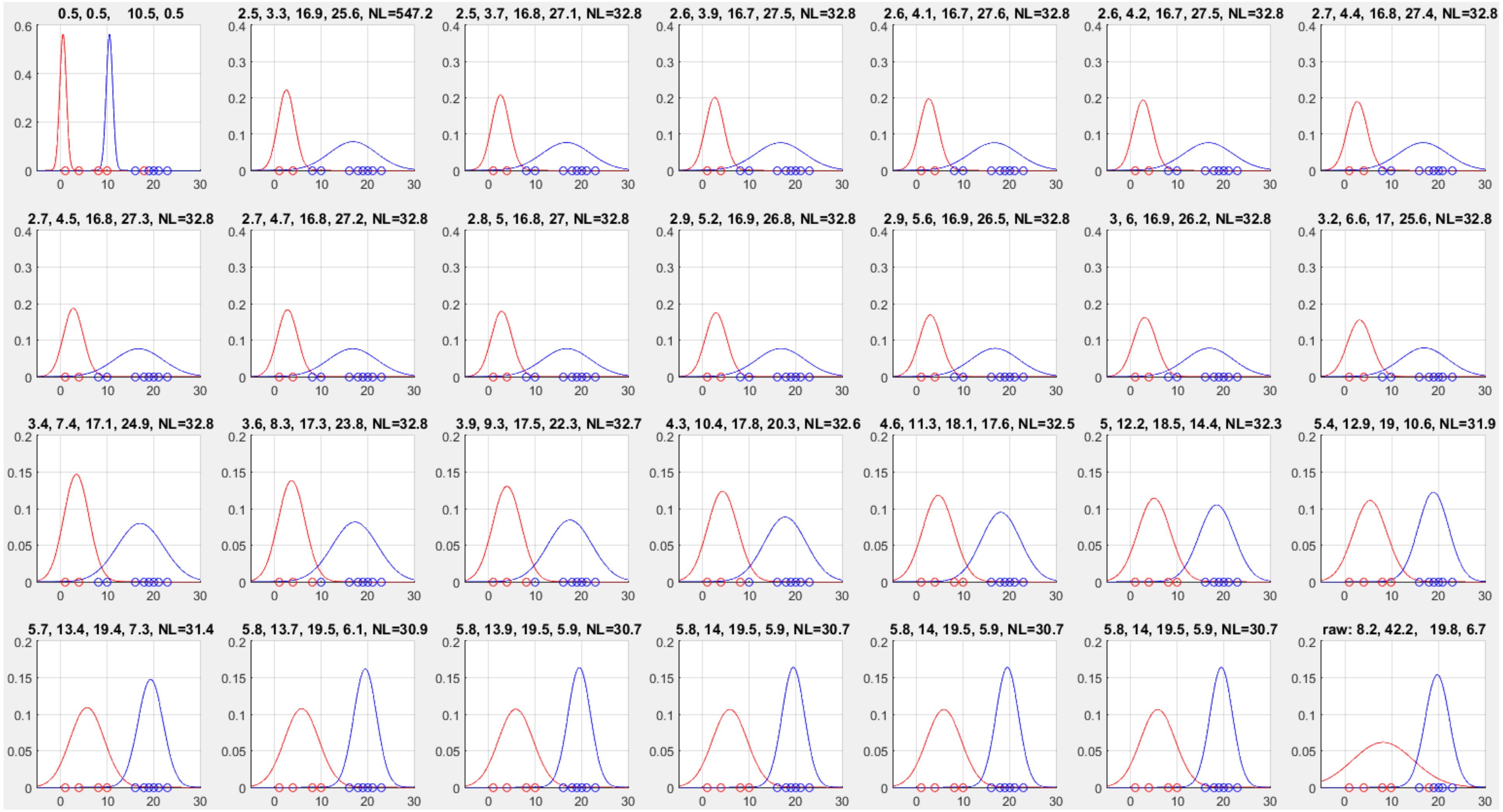


```
posterior probability...
 1 =      0.99762   0.0023752,  0
 4 =      0.98572    0.014284,  0
 8 =      0.77103     0.22897,  0
10 =      0.36396     0.63604,  0
16 =   8.3009e-05     0.99992,  1
18 =   0.00098233     0.99902,  0
19 =    2.259e-05     0.99998,  1
20 =   5.8842e-06     0.99999,  1
21 =   1.4671e-06           1,  1
23 =   7.9998e-08           1,  1
```

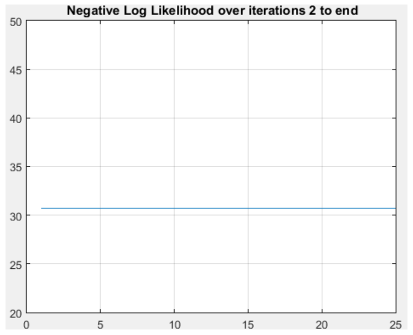


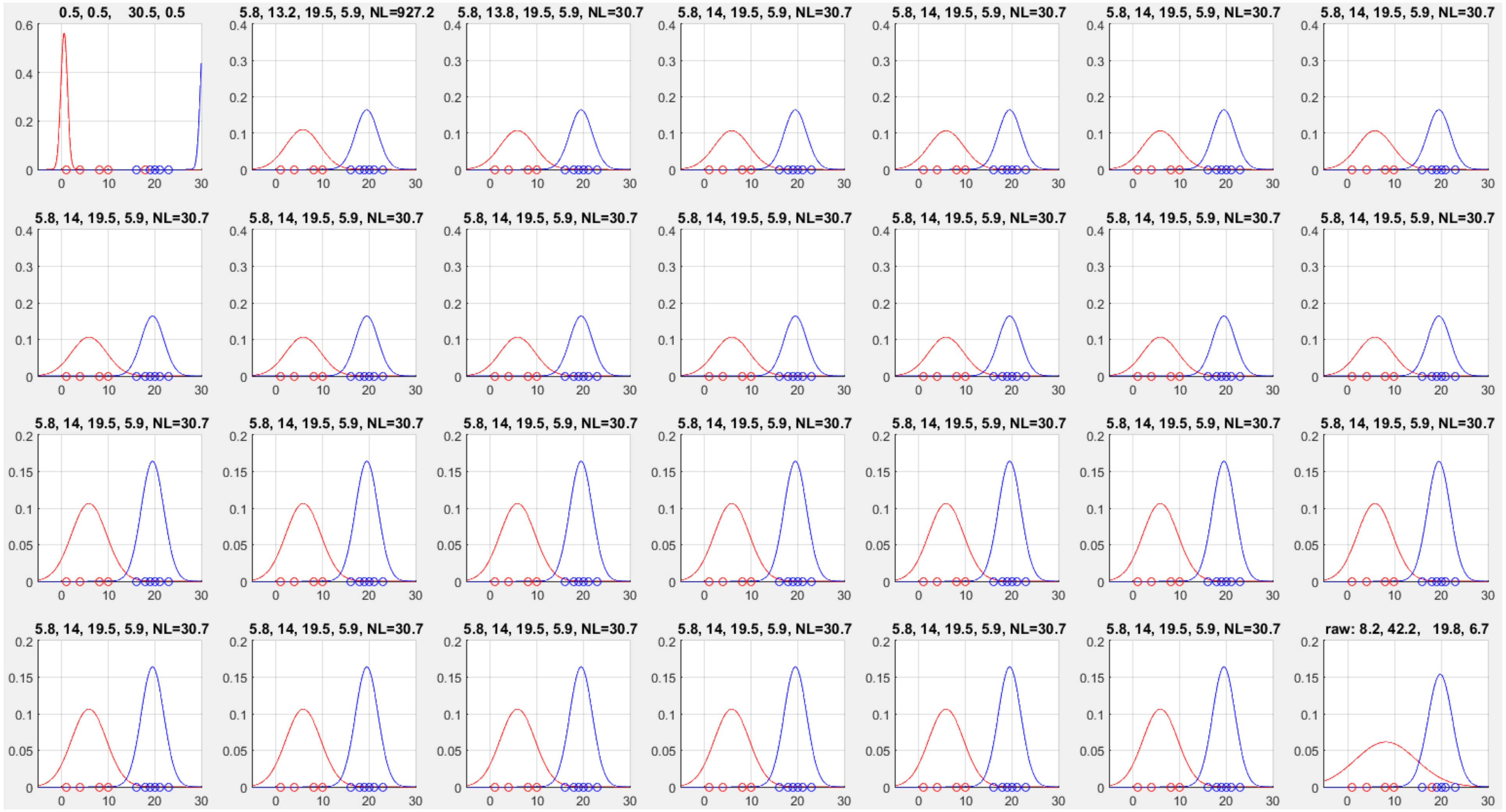
EM Iteration with Different Initial μ and Σ (2)



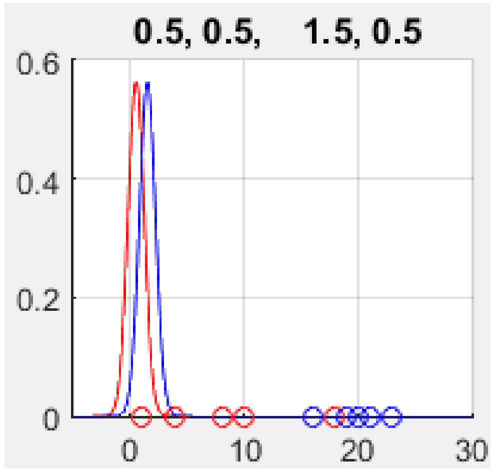


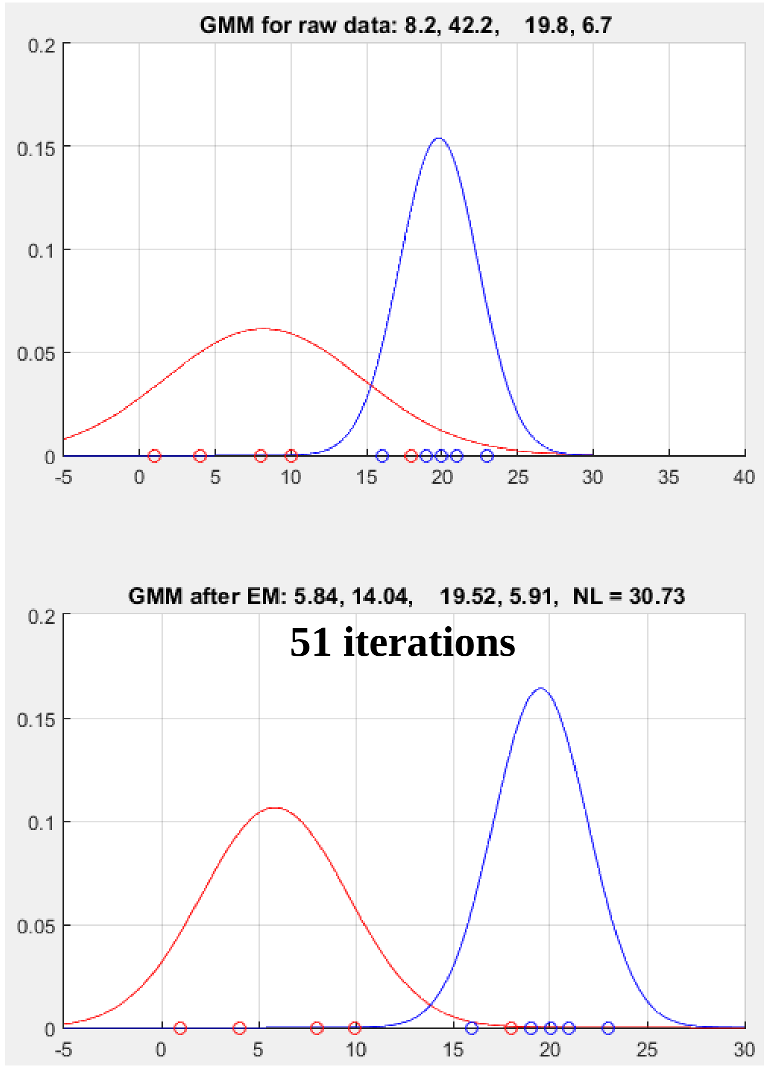
EM Iteration with Different Initial μ and Σ (3)

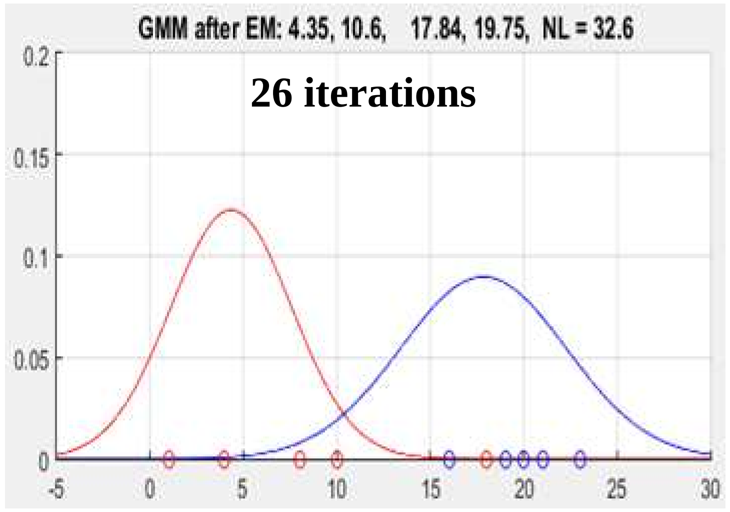




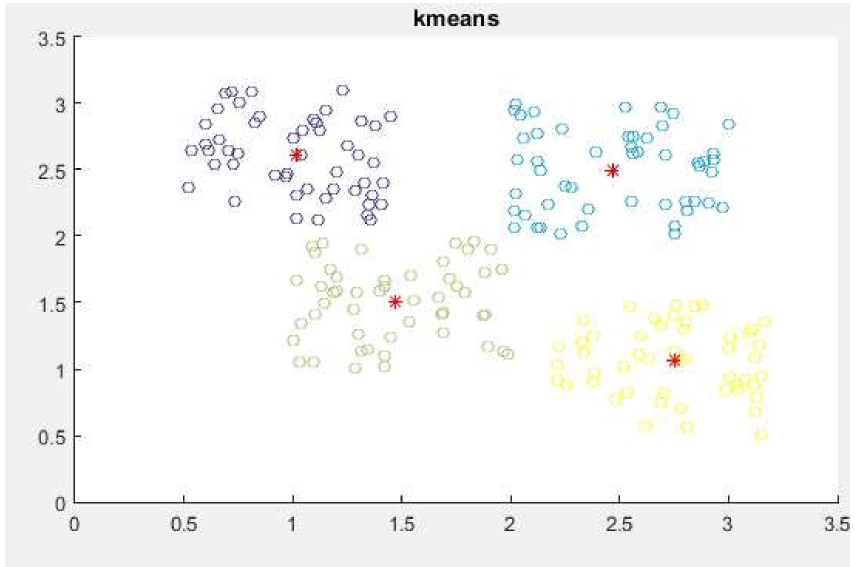
EM Iteration with Different Initial μ and Σ (1)

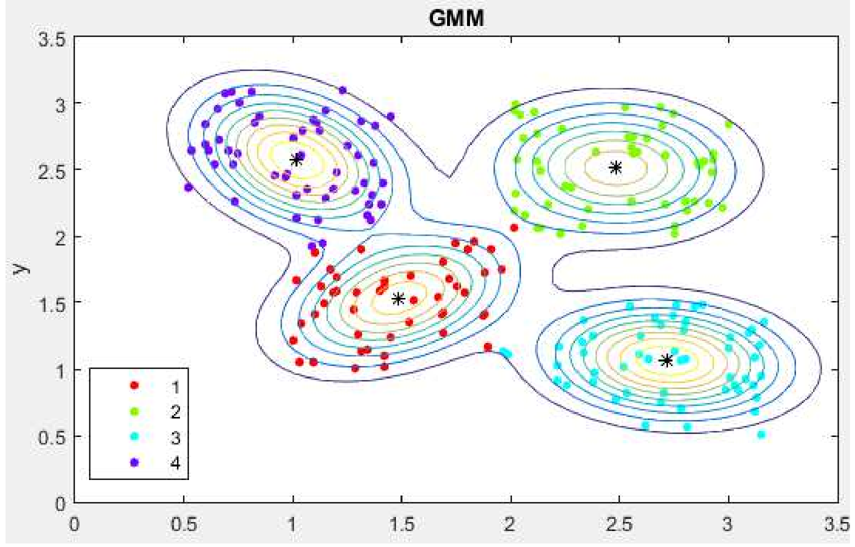


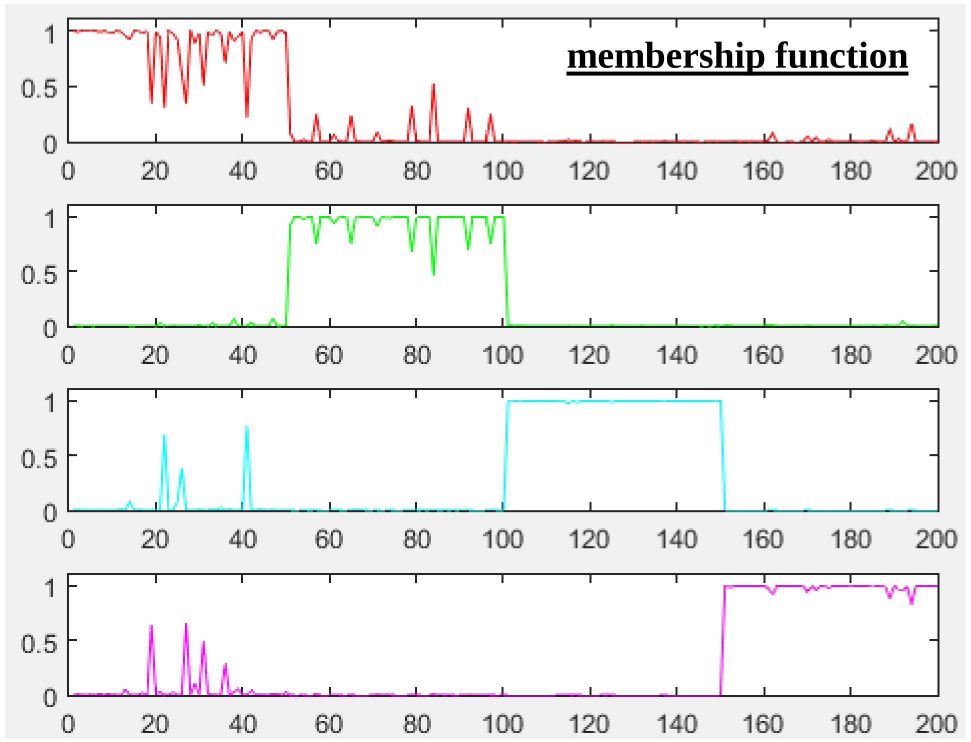




GMM vs. *k*-means



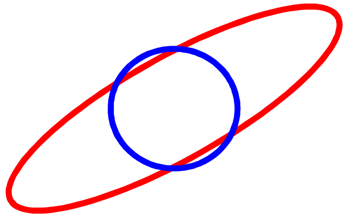


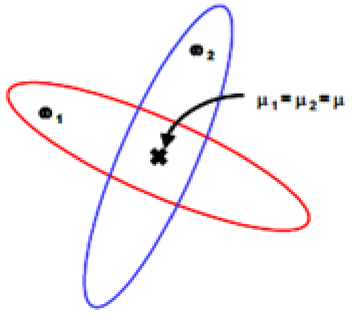


EM vs. ***K***-means

- Similar to ***K***-means.
  - Depends on starting components.
  - Objective is not convex➔ may find local minimum when converge.
  - Convergence is defined as no change or small change on ***NLL***.

- Different from ***K***-means.
  - Soft clustering→ assign each point to a Gaussian component w/ probability.
  - One data point can belong to multiple components w/ probability.
  - Keep changing means (centers) and variance (or covariance).
  - K-means fail if non-sephere clusters center at the same locations.

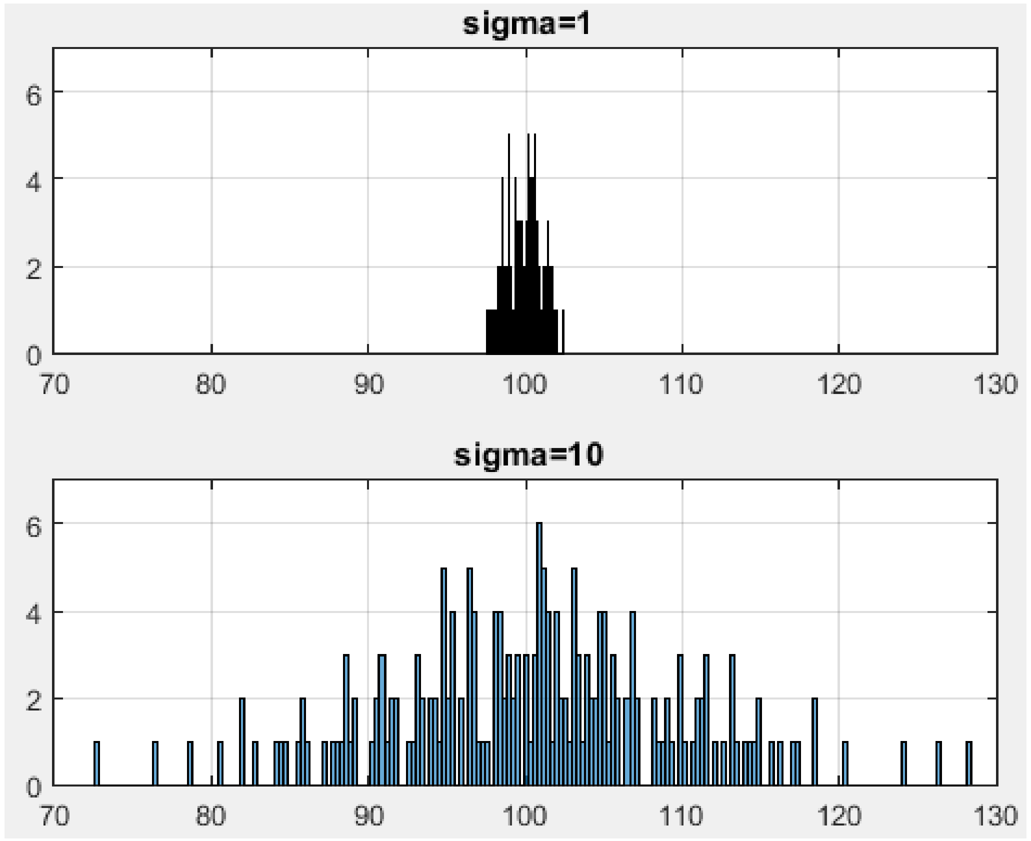




- Will ***EM*** = ***K***-means if we set variance = 0?
  - K-means is viewed as a special case of GMM.

GMM Regularization

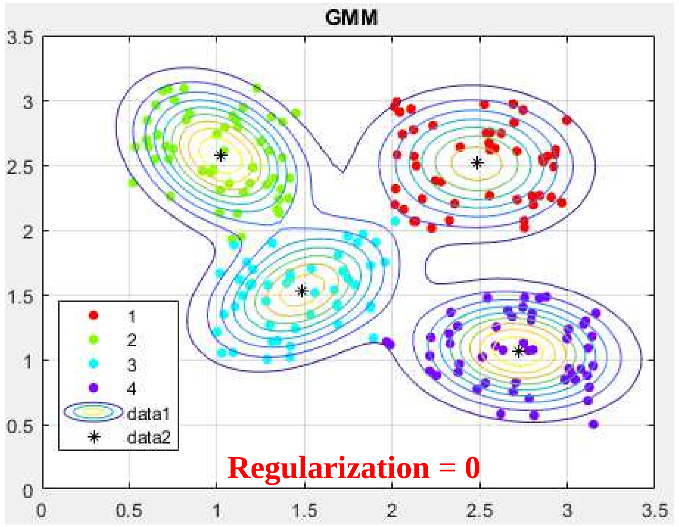
- Add a regularization number ($\geq$ **0**) (default 0) to the diagonal of covariance Σ.
  - Increase covariance in exchange for smaller estimation errors and **better stability**.
  - PDF spread out more smoothly.
  - Also improve "convergence" rates.

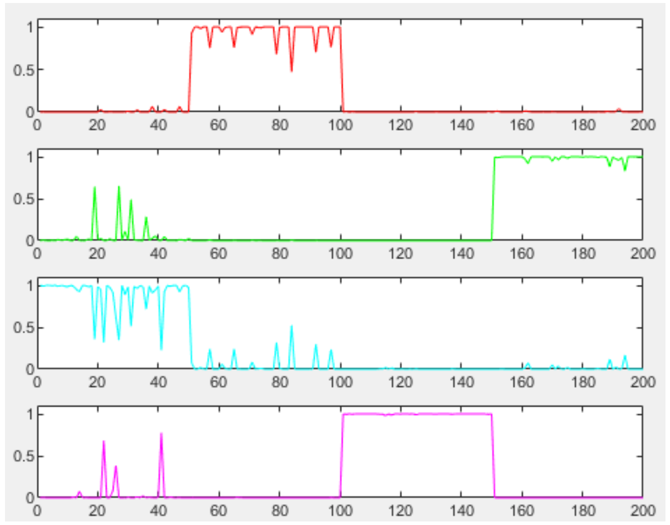


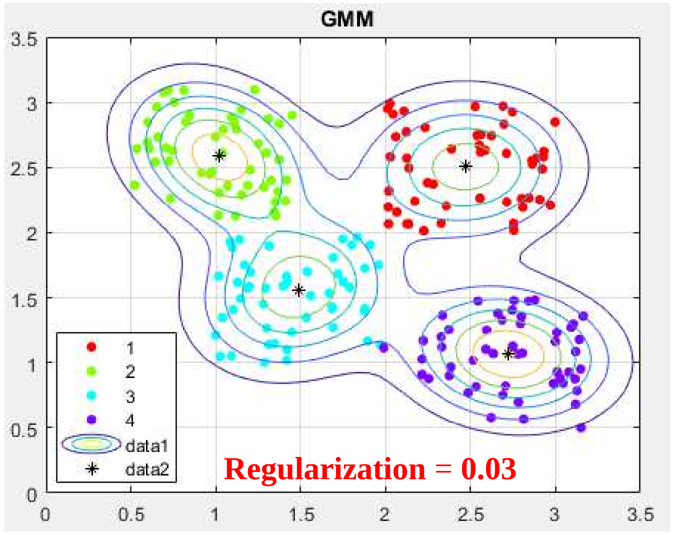
http://www.mathworks.com/help/stats/gmdistribution.fit.html

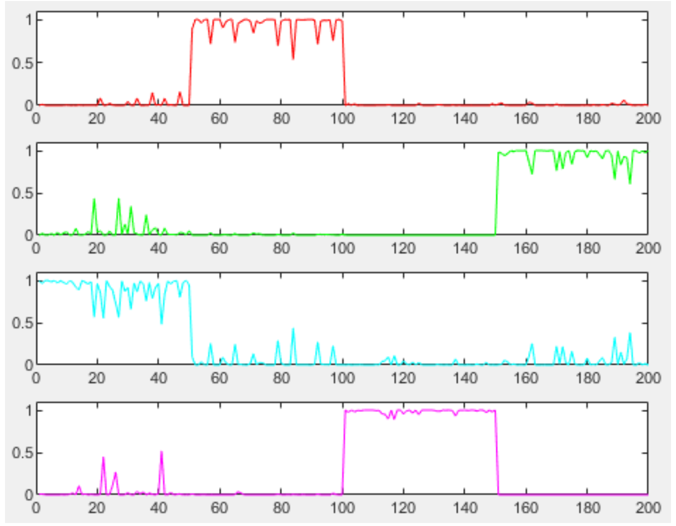
GMM Regularization

- Add a regularization number (**≥ 0**) (default 0) to the diagonal of covariance matrices.
  - Increase covariance in exchange for smaller estimation errors and **better stability**.





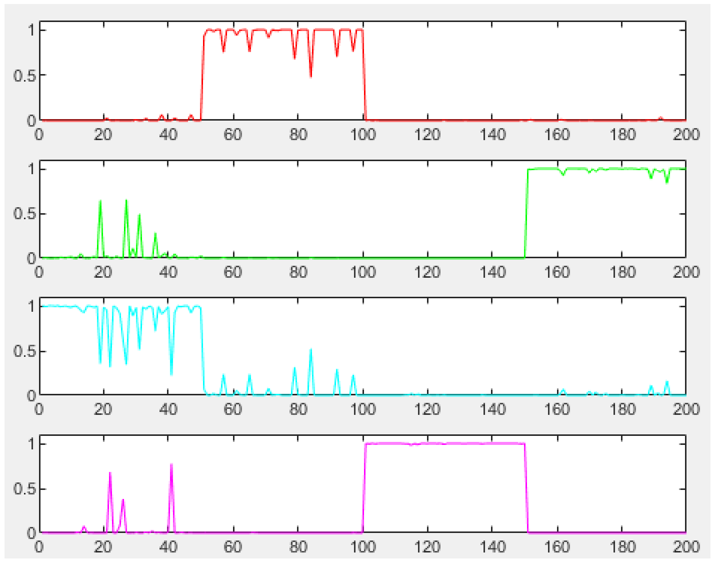


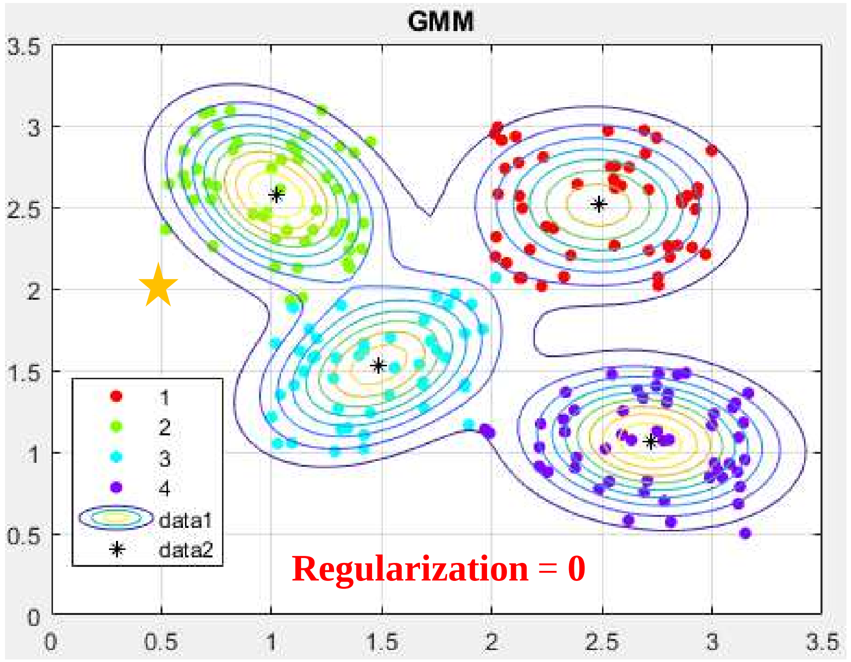


Anomaly Detection in GMM

- Compute ***PDF***(μ_g) against a GMM ***g***.
- Compute ***PDF***(**new**$_x$) against a GMM ***g***.
- Compute the ratio of above two.

```
rng(1)              %% Data
K = 4;
x=zeros(50,2);
x(:,1) = x(:,1)+2.2;  x(:,2) = x(:,2)+0.5;
y=zeros(50,2);
y(:,2) = y(:,2)+2.1;   y(:,1) = y(:,1)+0.5;
c3=rand(50,2)+1;       c1 = rand(50,2)+2;
c4 = rand(50,2)+x;    c2 = rand(50,2)+y;
t=[c1; c2; c3; c4];
```





Data Point = 0.5 2 **Regularization = 0**

P = 0.0000 0.9868 0.0132 0.0000 0.0032

Data Point = 1.0 2.5

P = 0.0000 0.9999 0.0001 0.0000 0.4794

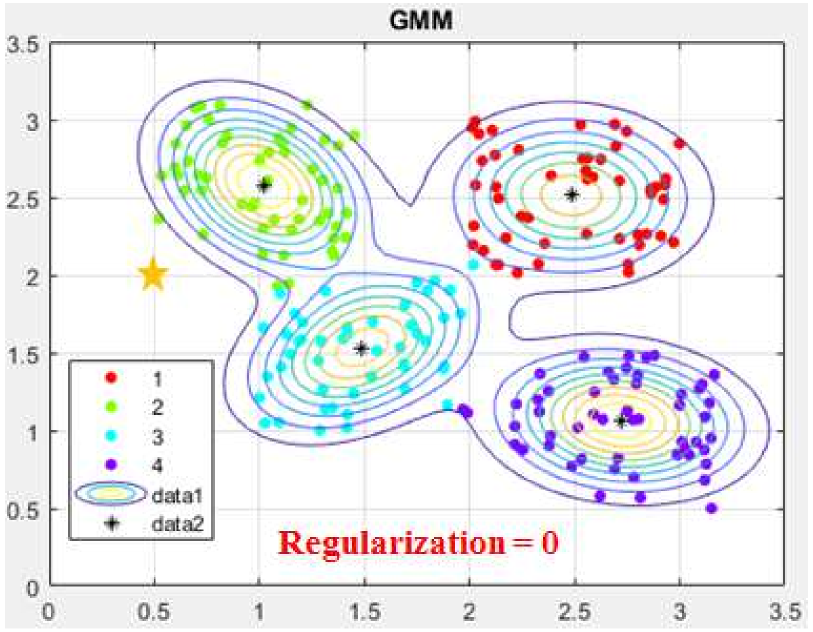
Data Point = 1.5 1.5

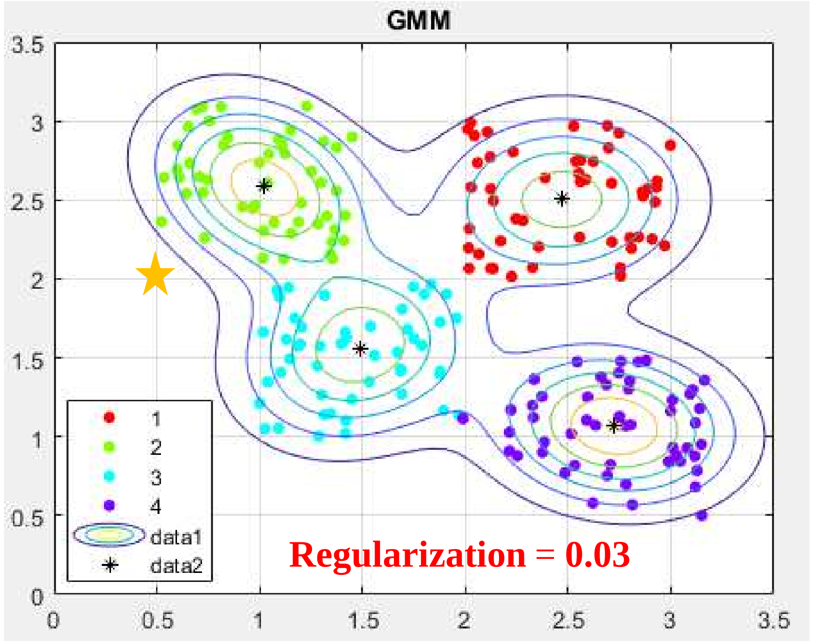
P = 0.0000 0.0029 0.9965 0.0005 0.4312

Data Point = 5 5

P = 1.0000 0.0000 0.0000 0.0000 0.0000

Anomaly Detection in GMM **w/ Regularization = 0.03**





Data Point = 0.5 2 **Regularization = 0**

P = 0.0000 0.9868 0.0132 0.0000 0.0032

Data Point = 1.0 2.5

P = 0.0000 0.9999 0.0001 0.0000 0.4794

Data Point = 1.5 1.5

P = 0.0000 0.0029 0.9965 0.0005 0.4312

Data Point = 5 5

P = 1.0000 0.0000 0.0000 0.0000 0.0000

Data Point = 0.5 2 **Regularization = 0.03**

P = 0.0000 0.8420 0.1580 0.0000 0.0149

Data Point = 1.0 2.5

P = 0.0006 0.9907 0.0087 0.0000 0.3273

Data Point = 1.5 1.5

P = 0.0007 0.0104 0.9861 0.0028 0.2897

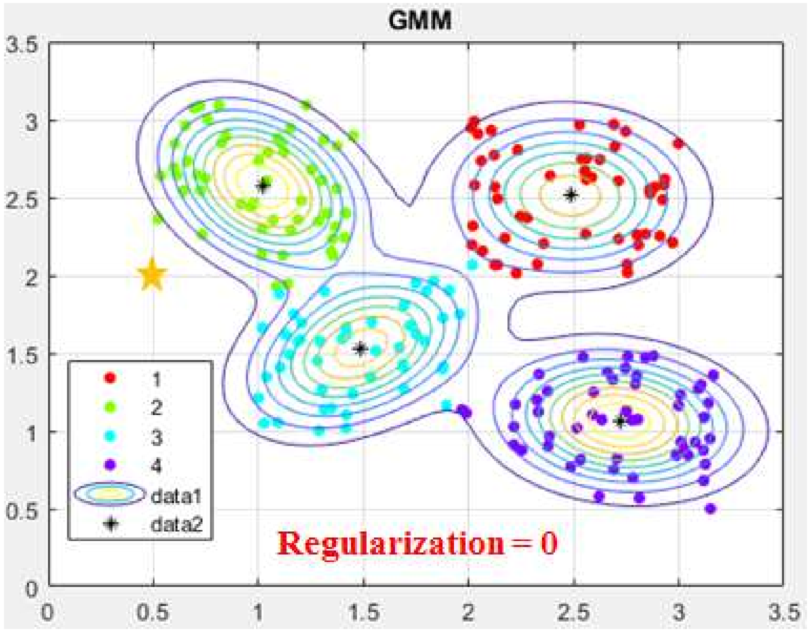
Data Point = 5 5

P = 1.0000 0.0000 0.0000 0.0000 0.0000

Identify Outliers w/ 4 One Class SVM RBF Models

- Build 4 one-class SVM-RBF models.
  - RBF, outlier ϵ C3, but *P*↓↓↓
  - GMM, outlier ϵ C2, but *P*↑↑↑ w/ PDF↓
  - RBF, outlier ϵ C3, but *P*↓↓↓
  - GMM, outlier ϵ C1, but *P*↑↑↑ w/ PDF↓

Data Point = 0.5	**2**	**RBF1**	
P = 2.572e-06	5.3751e-05	**0.00013274**	1.1234e-06
Data Point = 1.0	**2.5**	**RBF2**	
P = 0.022418	**0.98502**	0.00017523	2.1481e-07
Data Point = 1.5	**1.5**	**RBF3**	
P = 0.0013203	0.0011747	**0.99333**	7.409e-07
Data Point = 5 5		**RBF4**	
P = 1.5639e-06	1.8047e-06	**0.0003285**	1.2992e-06



Data Point = 0.5	**2**	**GMM Regularization = 0**		
P = 0.0000	**0.9868**	0.0132	0.0000	0.0032
Data Point = 1.0	**2.5**			
P = 0.0000	**0.9999**	0.0001	0.0000	0.4794
Data Point = 1.5	**1.5**			
P = 0.0000	0.0029	**0.9965**	0.0005	0.4312
Data Point = 5 5				
P = **1.0000**	0.0000	0.0000	0.0000	0.0000

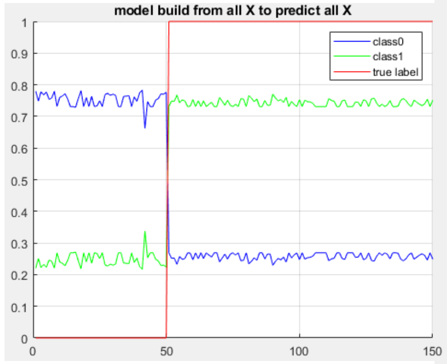
Code for The Outlier Detection Example

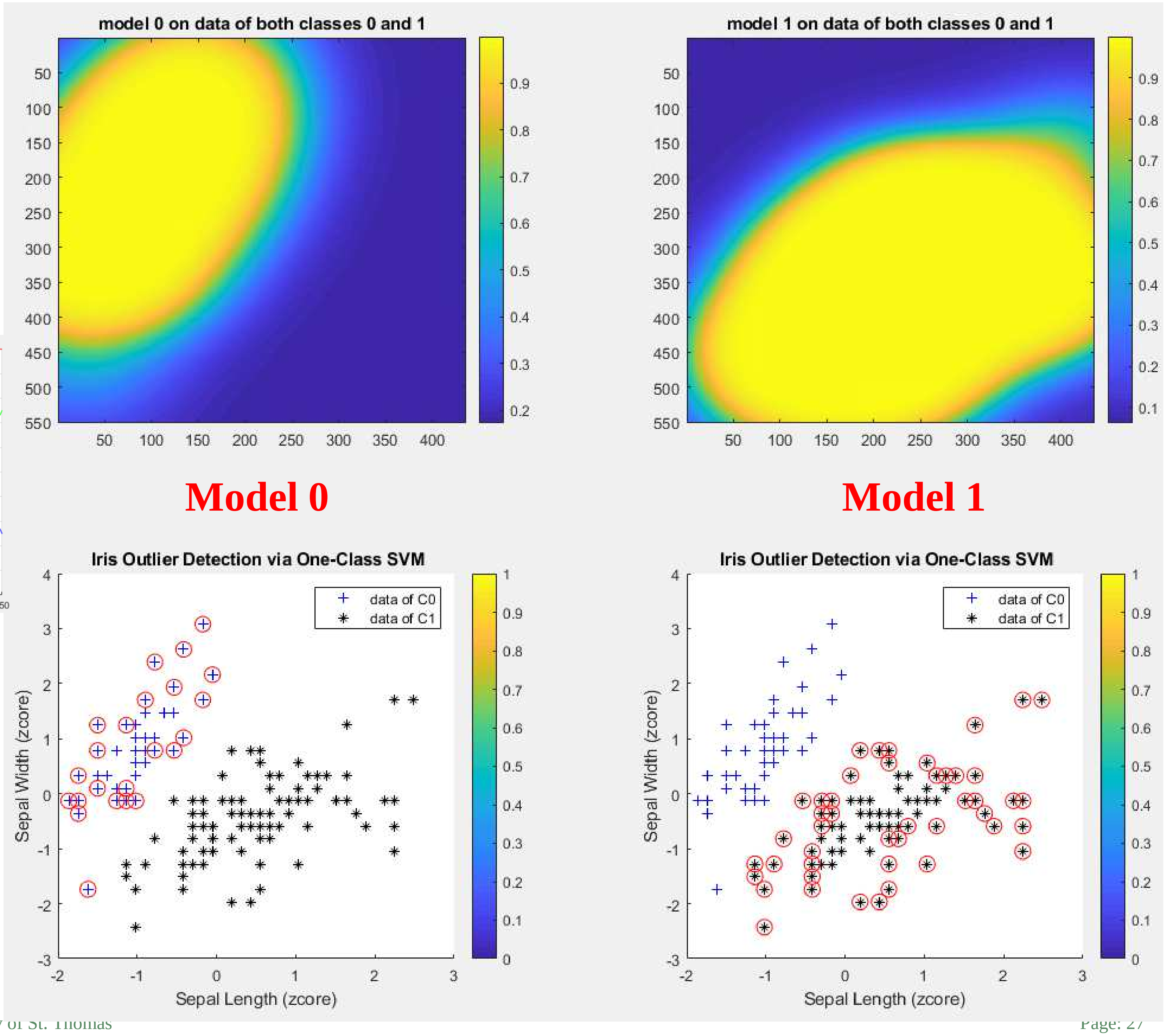
```
%% Build 4 one-class SVM-RBF models.
mdl_c1 = fitcsvm(c1, zeros(length(c1), 1), 'KernelFunction', 'rbf', 'ScoreTransform', 'logit');
mdl_c2 = fitcsvm(c2, zeros(length(c2), 1), 'KernelFunction', 'rbf', 'ScoreTransform', 'logit');
mdl_c3 = fitcsvm(c3, zeros(length(c3), 1), 'KernelFunction', 'rbf', 'ScoreTransform', 'logit');
mdl_c4 = fitcsvm(c4, zeros(length(c4), 1), 'KernelFunction', 'rbf', 'ScoreTransform', 'logit');

new_data = [0.5 2; 1.0 2.5; 1.5 1.5; 5 5];

%% predict probability of new data against each SVM RBF
[~, scores_c1] = predict(mdl_c1, new_data);
[~, scores_c2] = predict(mdl_c2, new_data);
[~, scores_c3] = predict(mdl_c3, new_data);
[~, scores_c4] = predict(mdl_c4, new_data);

%% build ONE GMM with 4 components
GMM = fitgmdist(t, K, 'RegularizationValue', 0.03);
[idx, nlogl, P, logpdf] = cluster(GMM, new_data);
```

```
%% Data
rng(1)
K = 4;
x=zeros(50,2);
x(:,1) = x(:,1)+2.2;  x(:,2) = x(:,2)+0.5;
y=zeros(50,2);
y(:,2) = y(:,2)+2.1;   y(:,1) = y(:,1)+0.5;
c3=rand(50,2)+1;      c1 = rand(50,2)+2;
c4 = rand(50,2)+x;   c2 = rand(50,2)+y;
t=[c1; c2; c3; c4];
```

```
gmm = mixture.GaussianMixture(n_components=5, covariance_type='full').fit(X)
gmm.predict(X)
```

SVM / RBF One Class, Iris (50 vs. 100) Dataset



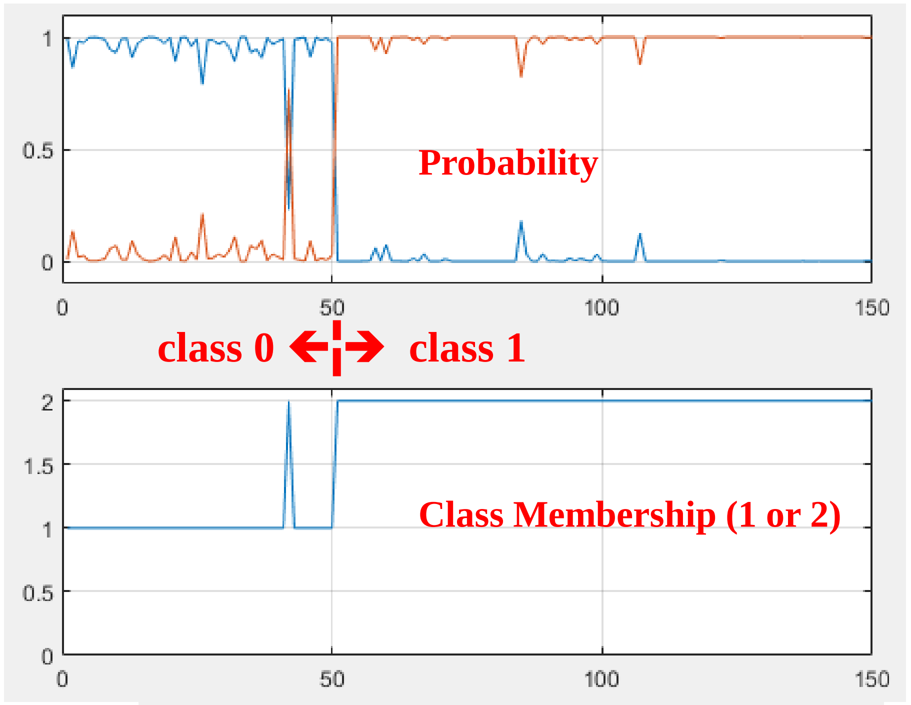


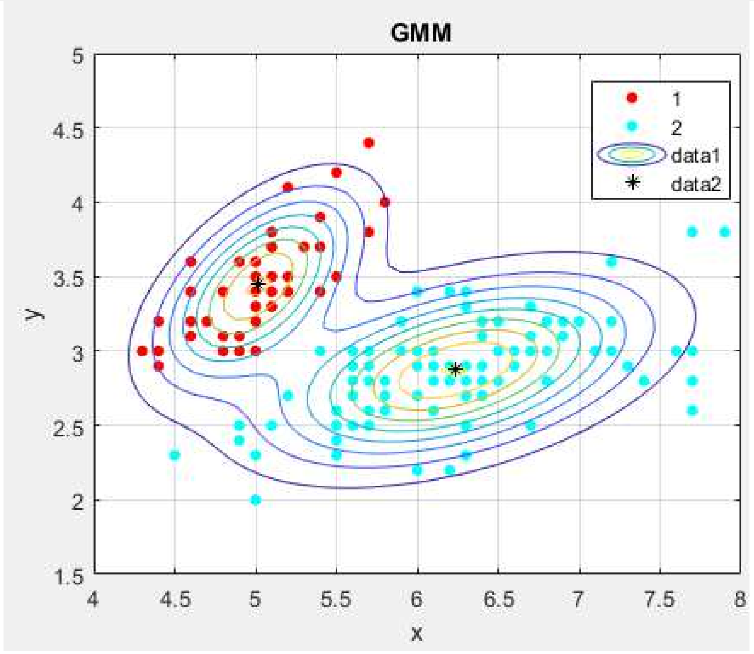
GMM, Iris (50 vs. 100)

- Assume no class info.➔so clustering 1st w/ GMM.
- Then predict records' GMM (class) probability.

```
load fisheriris;
X = meas(:, 1:2);
%%  NO need for Y   % Y = ~strcmp(species,'setosa');

%% build 1 GMM with 2 components
K = 2;
GMM = fitgmdist(X, K, 'RegularizationValue', 0.03);
[idx, nlogl, P, logpdf] = cluster(GMM, X);

figure, subplot(2,1,1), plot(P), ylim([-0.1 1.1]), grid on
subplot(2,1,2), plot(idx), ylim([0 2.1]), grid on;
```

```
h = 0.01; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)),...
   min(X(:,2)):h:max(X(:,2)));
[idx2, nlogl2, P2, logpdf2] = cluster(GMM, [X1(:),X2(:)]);
scoreGrid = reshape(P2(:,2),size(X1,1),size(X2,2));

figure, subplot(2,1,1), hold on, imagesc(scoreGrid), colorbar;
xlabel('Sepal Length'), ylabel('Sepal Width'), hold off

subplot(2,1,2), gscatter(X(:,1), X(:,2), idx); hold on
h = ezcontour(@(x,y)pdf(GMM,[x y]),[4 8],[1.5 5]);
plot(GMM.mu(:,1), GMM.mu(:,2), 'k*'), grid on, title('\bf GMM'), hold off
```

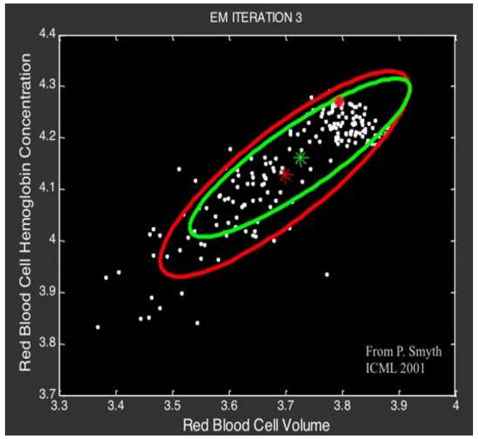




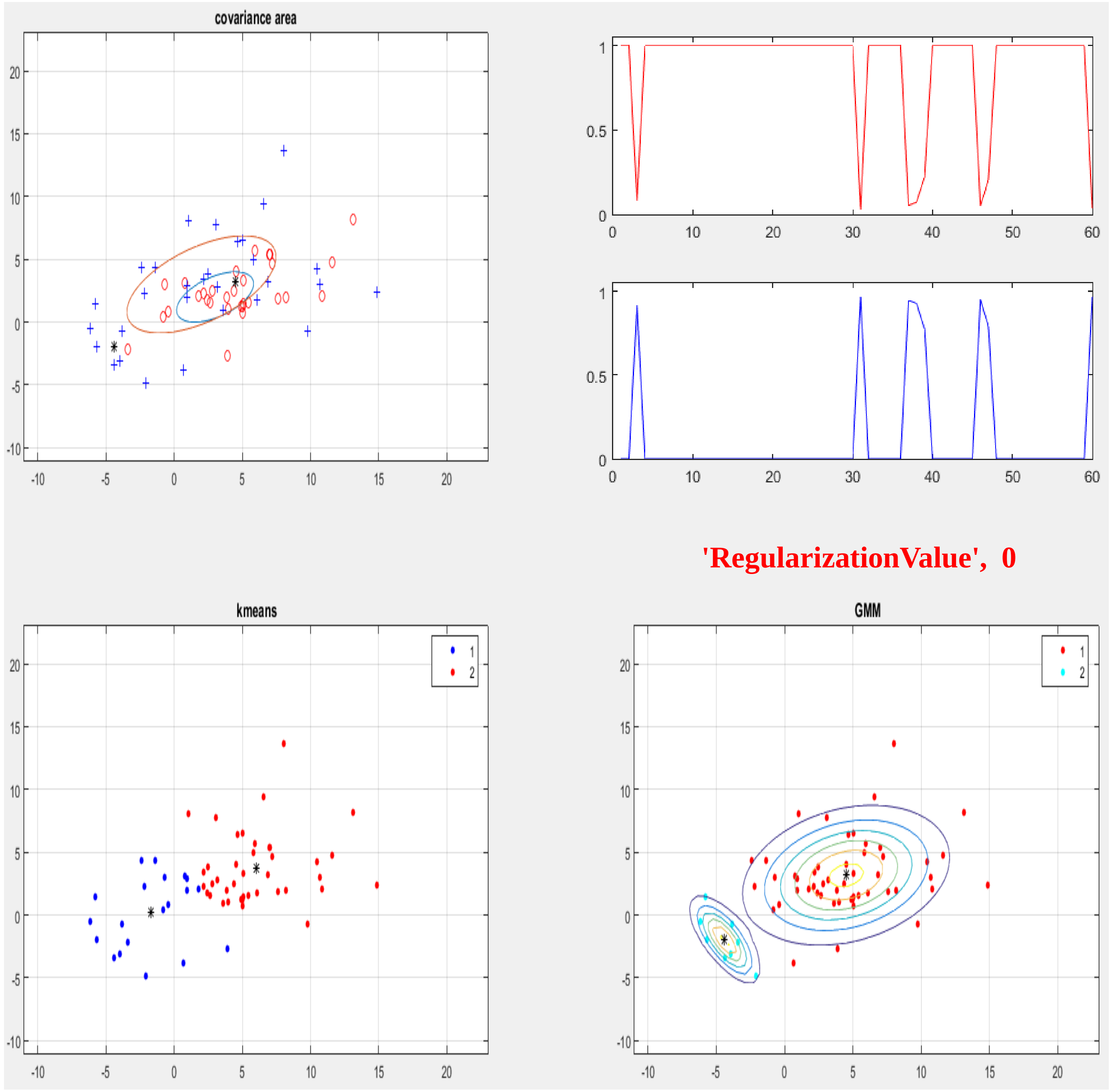
GMM on Overlapping Groups

No

Regularization



- Raw data
  - Center1 [3, 2]
  - Center2 [2, 3]

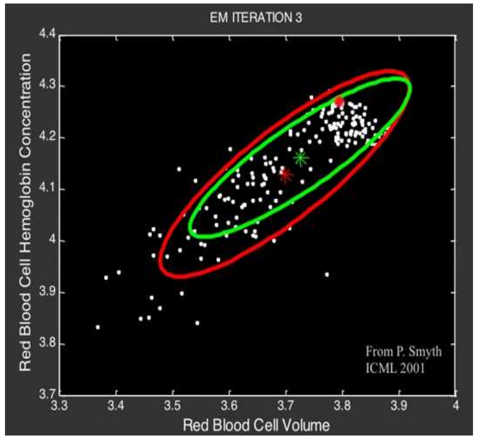


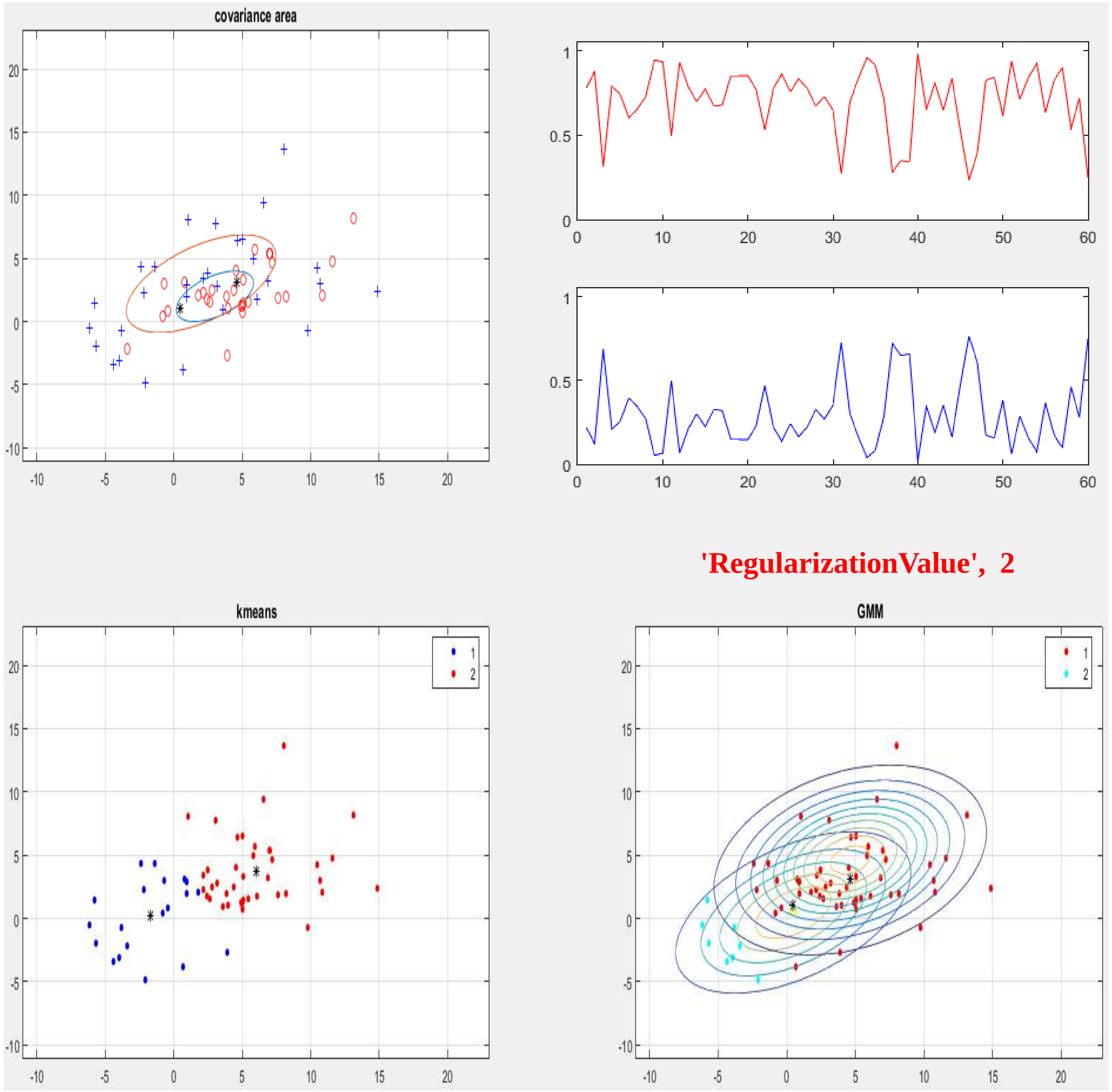
'RegularizationValue', 0

GMM on Overlapping Groups

with

Regularization



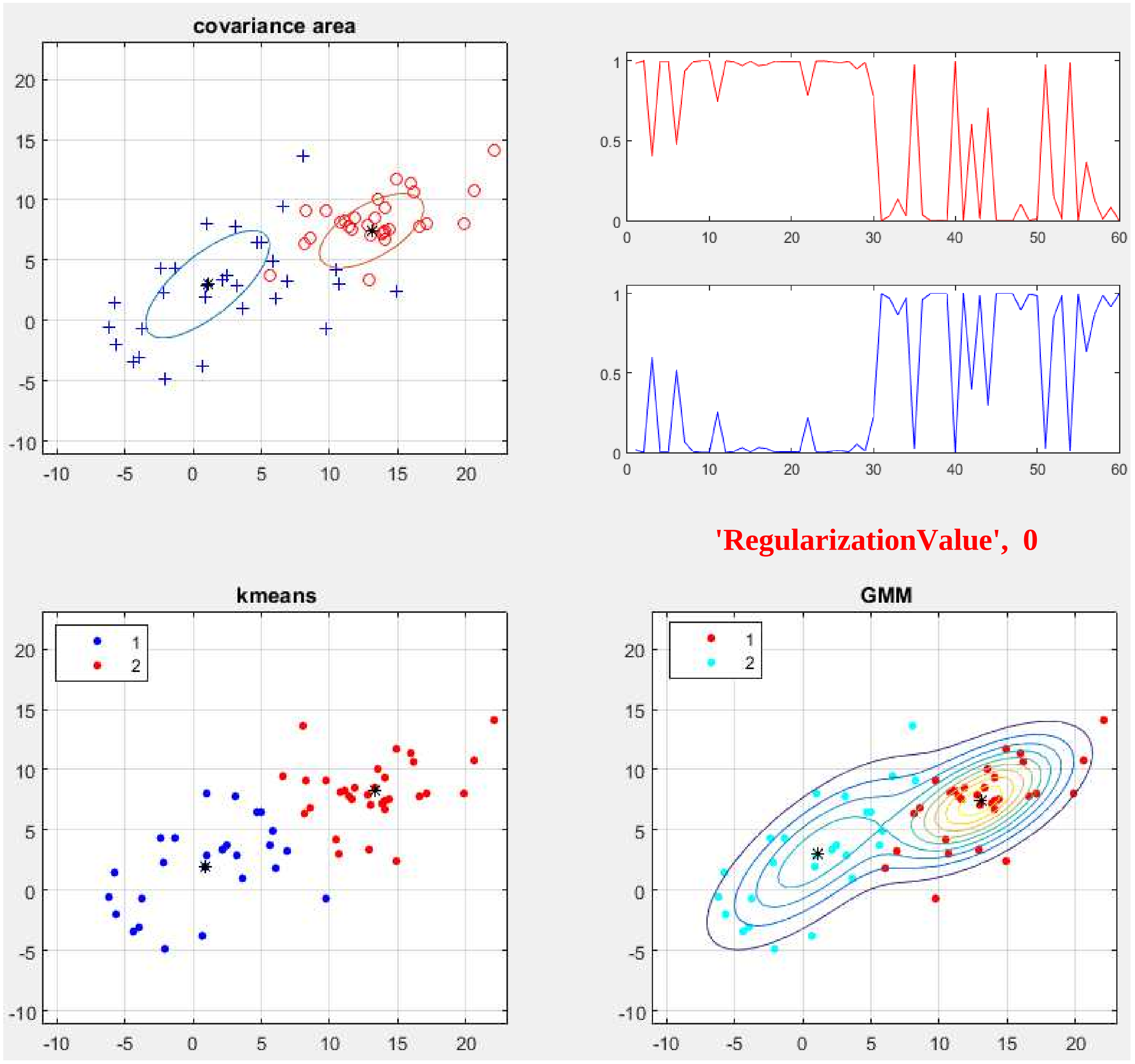


'RegularizationValue', 2

- Raw data
  - Center1 [3, 2]
  - Center2 [2, 3]

GMM on Overlapping Groups

No Regularization

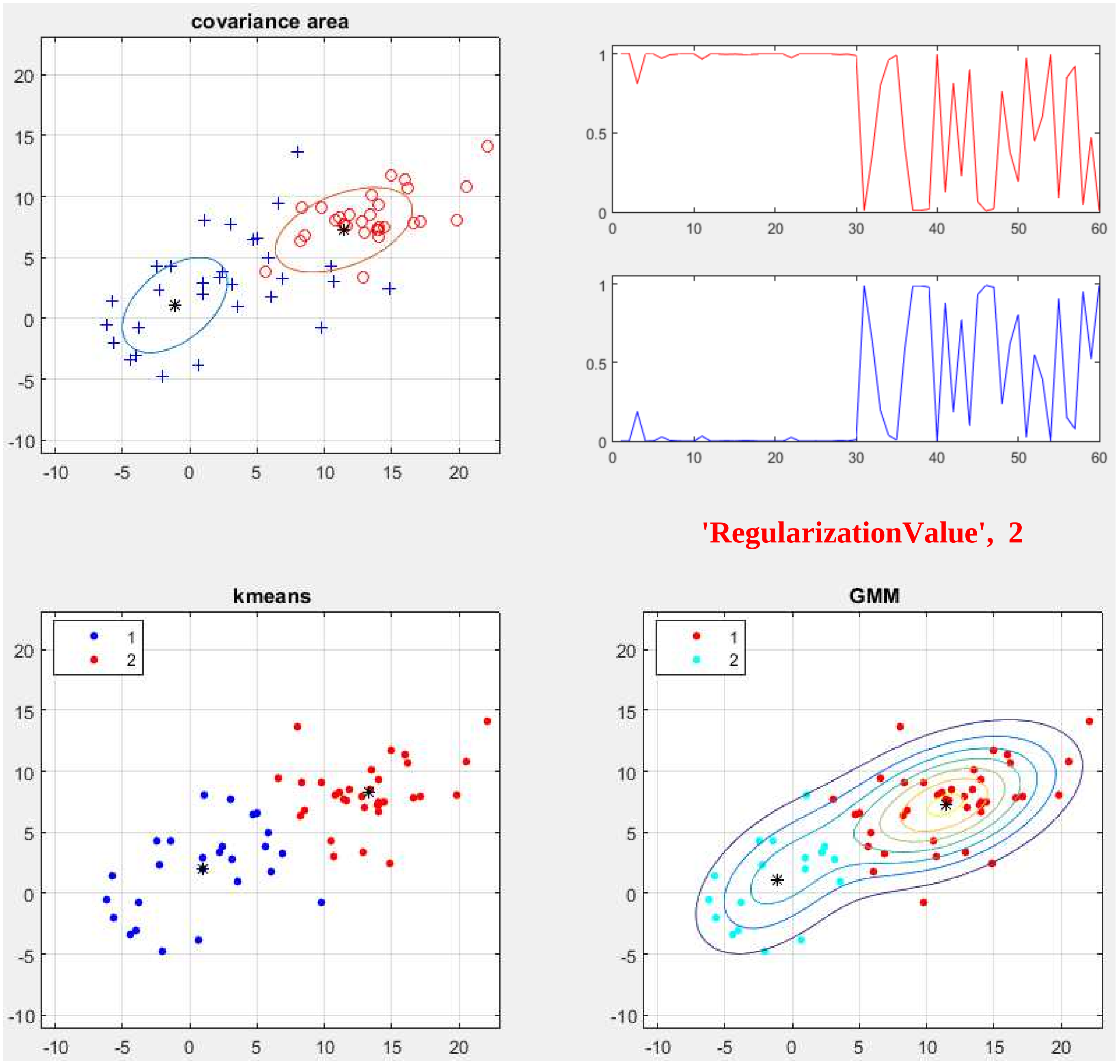


- Raw data
  - Center1 [12, 8]
  - Center2 [2, 3]

GMM on Overlapping Groups

with

Regularization



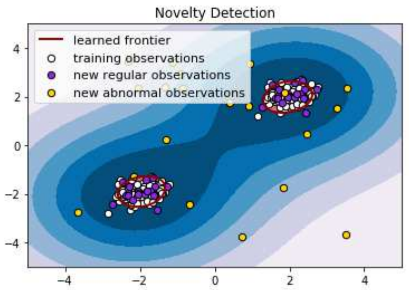
'RegularizationValue', 2

- Raw data
  - Center1 [12, 8]
  - Center2 [2, 3]

Code for GMM on Overlapping Groups **(most code for plotting)**

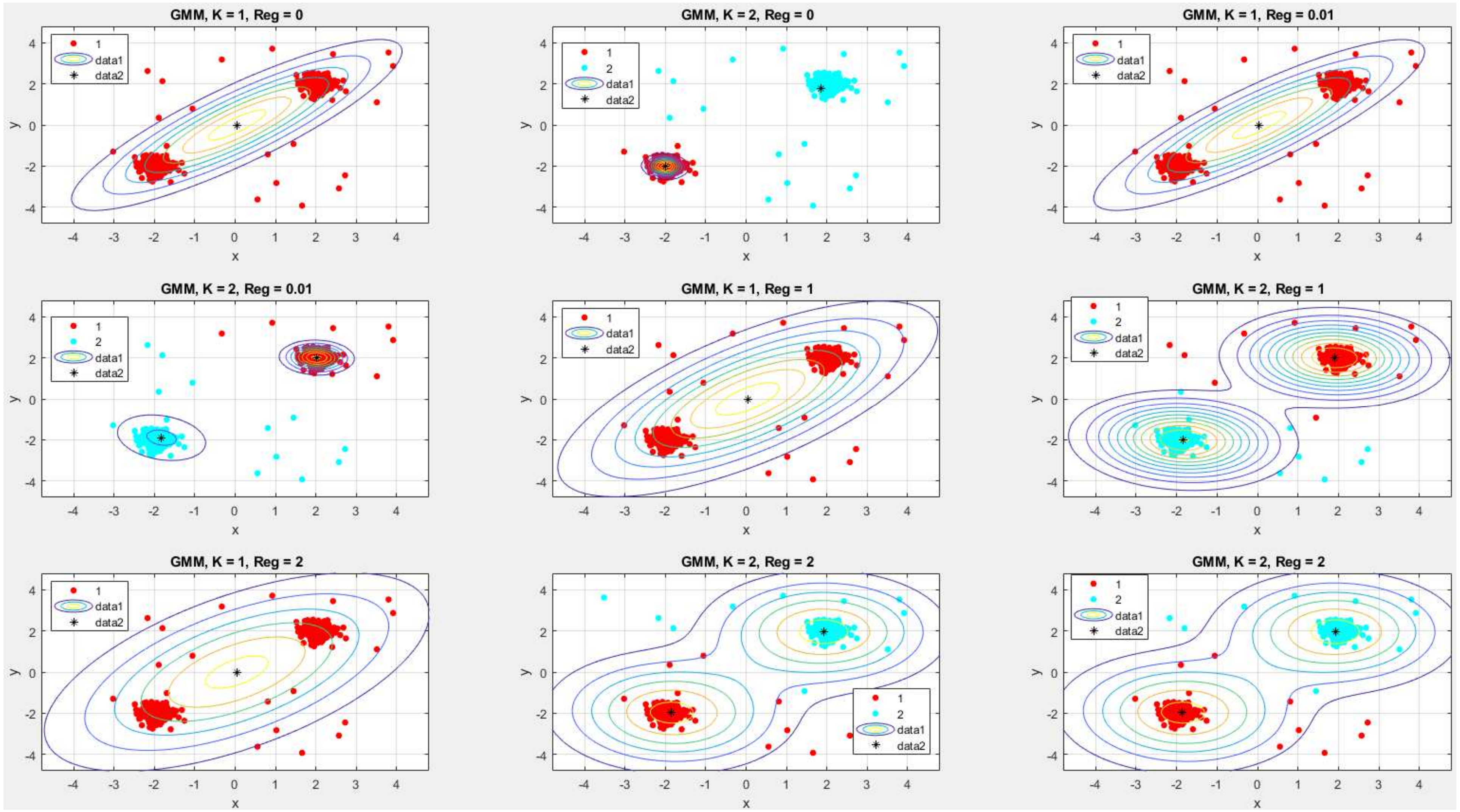
```
%% generate overlapping data
rng default                  % for reproducibility
mu = [12, 8];
sigma = [8, 3; 3, 4];        % covariance matrix
r1 = mvnrnd(mu, sigma, 30);   % random data 1
mu = [2,3];
sigma = [30, 12; 12, 15];  % covariance matrix
r2 = mvnrnd(mu, sigma, 30);   % random data 2
t = [r1; r2];                % merge to 1 dataset

%% k-means
[cidx, cen] = kmeans(t, 2);

%% plot k-means
figure,  subplot(2,2,3),
gscatter(t(:,1), t(:,2), cidx, 'br')
hold on, plot(cen(:,1), cen(:,2), 'k*'), grid on
xlim([-11 23]), ylim([-11 23]), title('\bf kmeans')
```

```
%% create GMM
GMM = fitgmdist(t, 2);
%% create clusters based on GMM
[idx, nlogl, P, logpdf] = cluster(GMM, t, 'RegularizationValue', 0);
%% plot GMM
subplot(2,2,4), gscatter(t(:,1), t(:,2), idx); hold on
ezcontour(@(x,y)pdf(GMM, [x y]), [-11 23], [-11 23]);
plot(GMM.mu(:,1), GMM.mu(:,2), 'k*'), title('\bf GMM')
xlabel(''), ylabel(''), grid on, hold off
subplot(2,2,1), plot(r1(:,1), r1(:,2), 'or'), hold on
plot(r2(:,1), r2(:,2), '+b'),
plot_gaussian_ellipsoid(GMM.mu(2,:), GMM.Sigma(:,:,2));
xlim([-11 23]), ylim([-11 23])
plot_gaussian_ellipsoid(GMM.mu(1,:), GMM.Sigma(:,:,1));
xlim([-11 23]), ylim([-11 23]), grid on
plot(GMM.mu(:,1), GMM.mu(:,2), 'k*'),  hold off
title('covariance area')
%% plot cluster membership (probability)
figure, subplot(2,1,1),plot(P(:,1), 'r'), ylim([0 1.05])
subplot(2,1,2), plot(P(:, 2), 'b'), ylim([0 1.05])
```



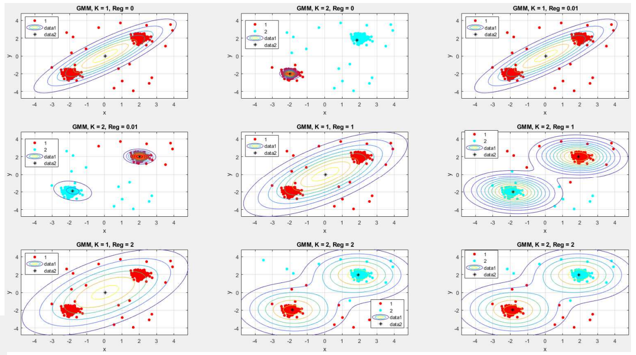
GMM Needs Known *k* Components

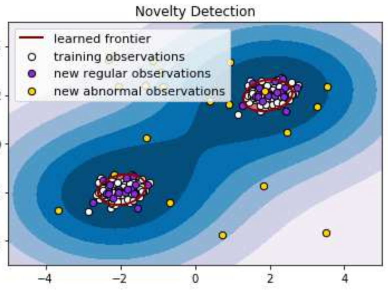
- Fail if # GMM = 1➔ must create 2 GMMs.
  - But, you know the 2GMMs represent records of **ONE** class.

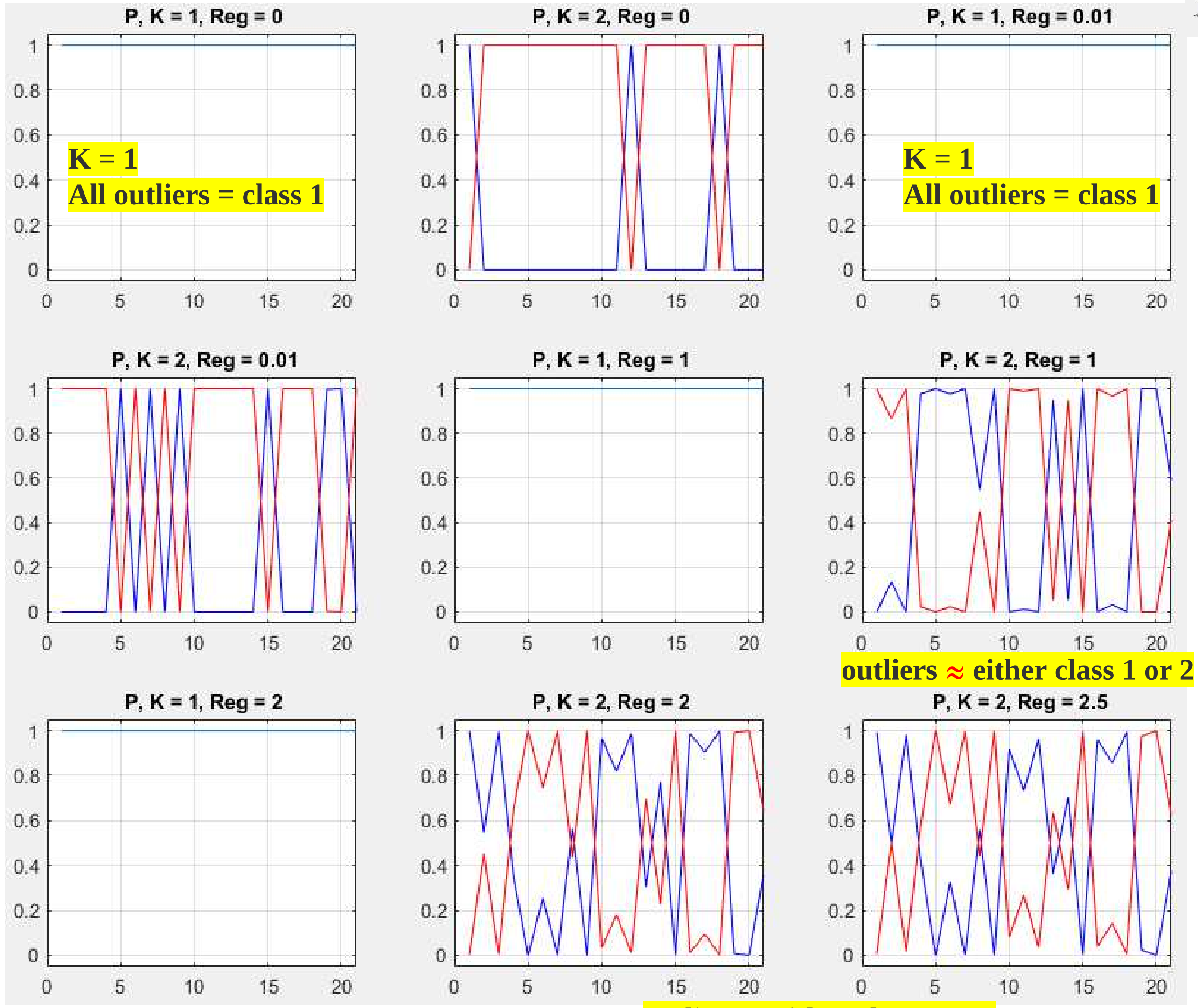


Identifying Outliers (cont'd from last slide)

- Use probability to identify outliers
  - Effects of ***regularization***.







K = 1
All outliers = class 1

K = 1
All outliers = class 1

outliers ≈ either class 1 or 2

outliers ≈ either class 1 or 2

EM Summary

- Clusters are assigned by selecting GMMs to maximize the posterior probability.
  - or minimize ***NLL***.

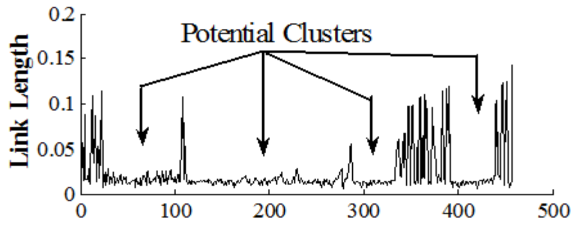
- An iterative algorithm that converges to a **local** optimum.

- GMM clustering is referred to as a soft clustering method. (i.e. probability)

- $\boldsymbol{\mu}$ and Σ of each GMM represent clusters.

- Final thoughts / reviews…

- If you don't have labels for records, clustering first.
  - But, what is the ***K***?
  - Minimum spanning tree (MST).
  - Then, you can do classification using clustering probability.



- If you have labels for VERY FEW records?
  - Un-supervised learning (clustering), semi-supervised learning (transductive…)

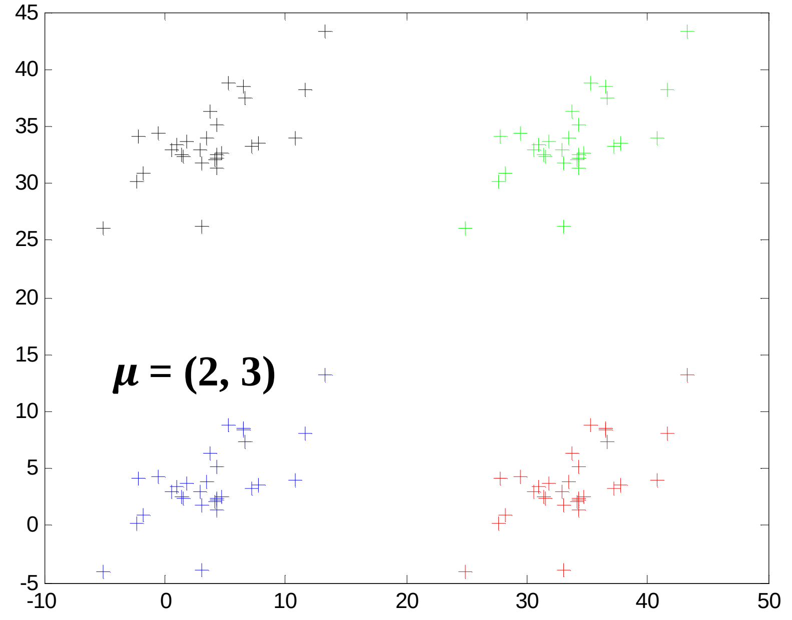
Appendix

Multiple Multivariate Clusters

- Matlab function **mvnrnd(μ, Σ, #pts)**
  - Use covariance Σ (not standard deviation σ).

```
mu = [2,3];
sigma = [10, 6; 6, 10];       % symmetric covariance matrix
rng default                   % For reproducibility
r = mvnrnd(mu, sigma, 30);
r2 = r;         r3 = r;         r4 = r;
MoveGap = 30;
figure, plot(r(:,1), r(:,2), '+'),
hold on
r2(:, 1) = r2(:, 1) + MoveGap ; plot(r2(:,1), r2(:,2), '+r'),
r3(:, 2) = r3(:, 2) + MoveGap ; plot(r3(:,1), r3(:,2), '+k'),
r4 = r4 + MoveGap ;           % move both axes
plot(r4(:,1), r4(:,2), '+g'),
hold off
```

$$\Sigma = \begin{bmatrix} \mathbf{10} & \mathbf{6} \\ \mathbf{6} & \mathbf{10} \end{bmatrix}$$

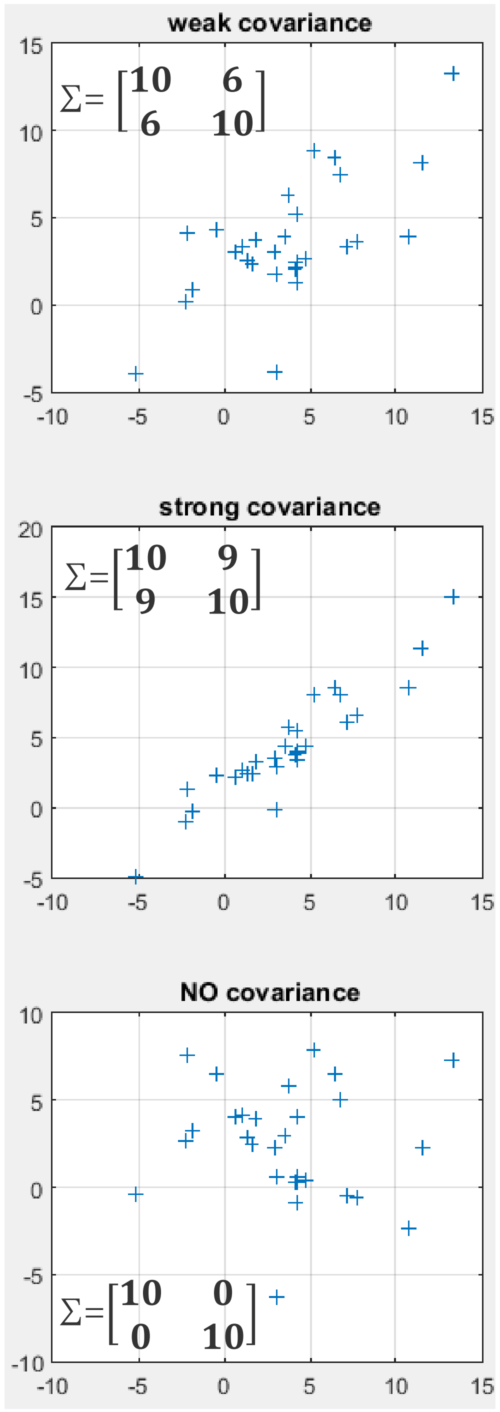


Multivariate Normal Random Numbers

- Matlab function **mvnrnd(μ, Σ, #pts)**

μ = (2, 3)

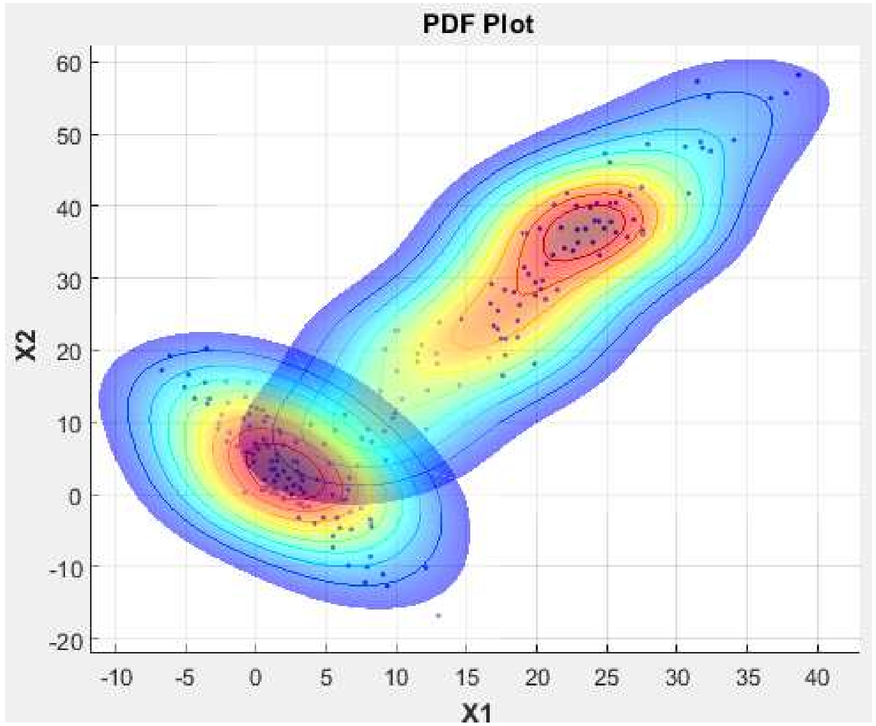
```
rng default                    % for reproducibility
mu = [2,3];
sigma = [10, 6; 6, 10];        % symmetric WEAK covariance matrix
r = mvnrnd(mu, sigma, 30);              % μ, Σ (not σ), #pts
figure,
subplot(3,1,1), plot(r(:,1), r(:,2), '+'), title('\bf weak covariance'), grid on

%%%%%%%%%%%%%%%%%%
sigma = [10, 9; 9,10];         % symmetric STRONG covariance matrix
rng default                    % for reproducibility
r = mvnrnd(mu, sigma, 30);              % μ, Σ (not σ), #pts
subplot(3,1,2), plot(r(:,1), r(:,2), '+'), title('\bf strong covariance'), grid on

%%%%%%%%%%%%%%%%%%
sigma = [10 10];               % x1, x2 no covariance = sigma [10, 0; 0, 10];
rng default                    % for reproducibility
r = mvnrnd(mu, sigma, 30);              % μ, Σ (not σ), #pts
subplot(3,1,3), plot(r(:,1), r(:,2), '+'), title('\bf NO covariance'), grid on
```

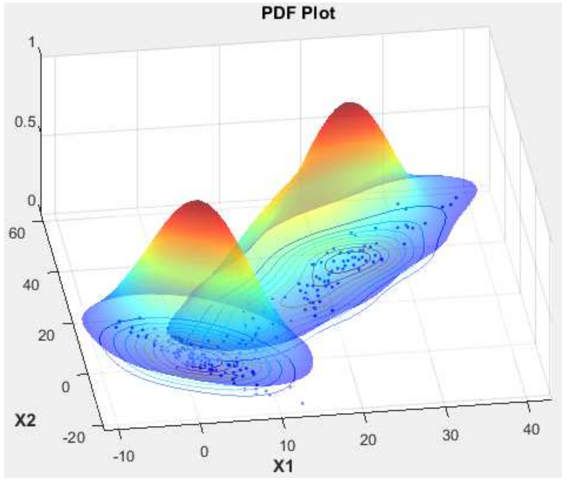


Plot Data with 2 Classes

http://stackoverflow.com/questions/9134014/contour-plot-coloured-by-clustering-of-points-matlab

```
rng(5)

% dataset 1
mu = [20  30];     sigma = [5  8;   8   15]*10;
r = mvnrnd(mu,sigma,100);

% dataset 2
mu2 = [2  3];       sigma2 = [3  -5;   -5   11] * 5;
r2 = mvnrnd(mu2,sigma2,100);

% plot
data = [r; r2];
hx = figure;
PDF_Plot(r, hx),
hold on
PDF_Plot(r2, hx),
hold off
title('\bf PDF Plot'), xlabel('\bf X1');   ylabel('\bf X2');
```

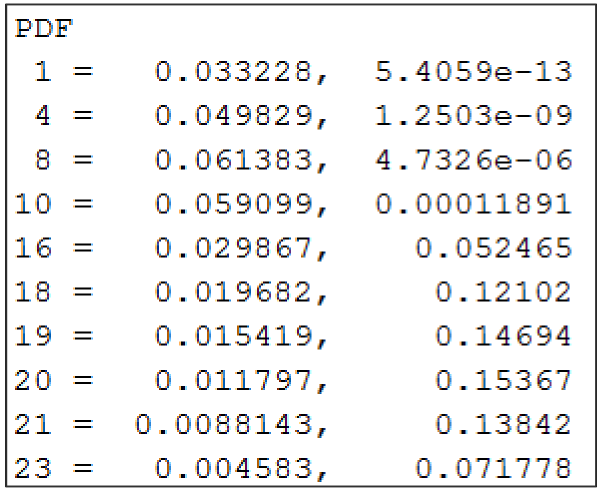




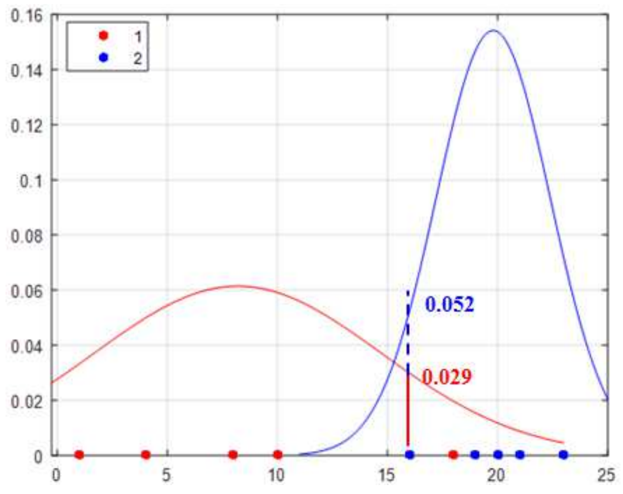
Chicken and Eggs (example with $\boldsymbol{k}$ = 2)

- Chicken and Eggs
  - Need good $\boldsymbol{\mu}$ and Σ to guess each point belongs to which Gaussian component.
    - $\boldsymbol{\mu_R}$ = 8.2, Σ_R = 42.2, $\boldsymbol{\mu_B}$ = 19.8, Σ_R = 6.7
  - Need to know which component each point came from to compute $\boldsymbol{\mu}$ and Σ.

- Assume assigning points to wrong components, we can re-estimate new $\boldsymbol{\mu}$ and Σ of each component to minimize Negative Log-Likelihood.
  - Expectation (E-step), followed by Maximization (M-step).



```
PDF
 1 =    0.033228,  5.4059e-13
 4 =    0.049829,  1.2503e-09
 8 =    0.061383,  4.7326e-06
10 =    0.059099,  0.00011891
16 =    0.029867,    0.052465
18 =    0.019682,     0.12102
19 =    0.015419,     0.14694
20 =    0.011797,     0.15367
21 =   0.0088143,     0.13842
23 =    0.004583,    0.071778
```



$$P(x \mid c) = f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Matlab function =

normpdf(data, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$)

mvnpdf(data, $\boldsymbol{\mu}$, Σ)

$$\frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Posterior Probability

- ***P***(***c*** | ***x***) = posterior probability of class ***c*** provided instance ***x***. (i.e. Prob. of ***x*** belong to class ***c***)
  - http://www.mathworks.com/help/stats/compactclassificationdiscriminant.predict.html

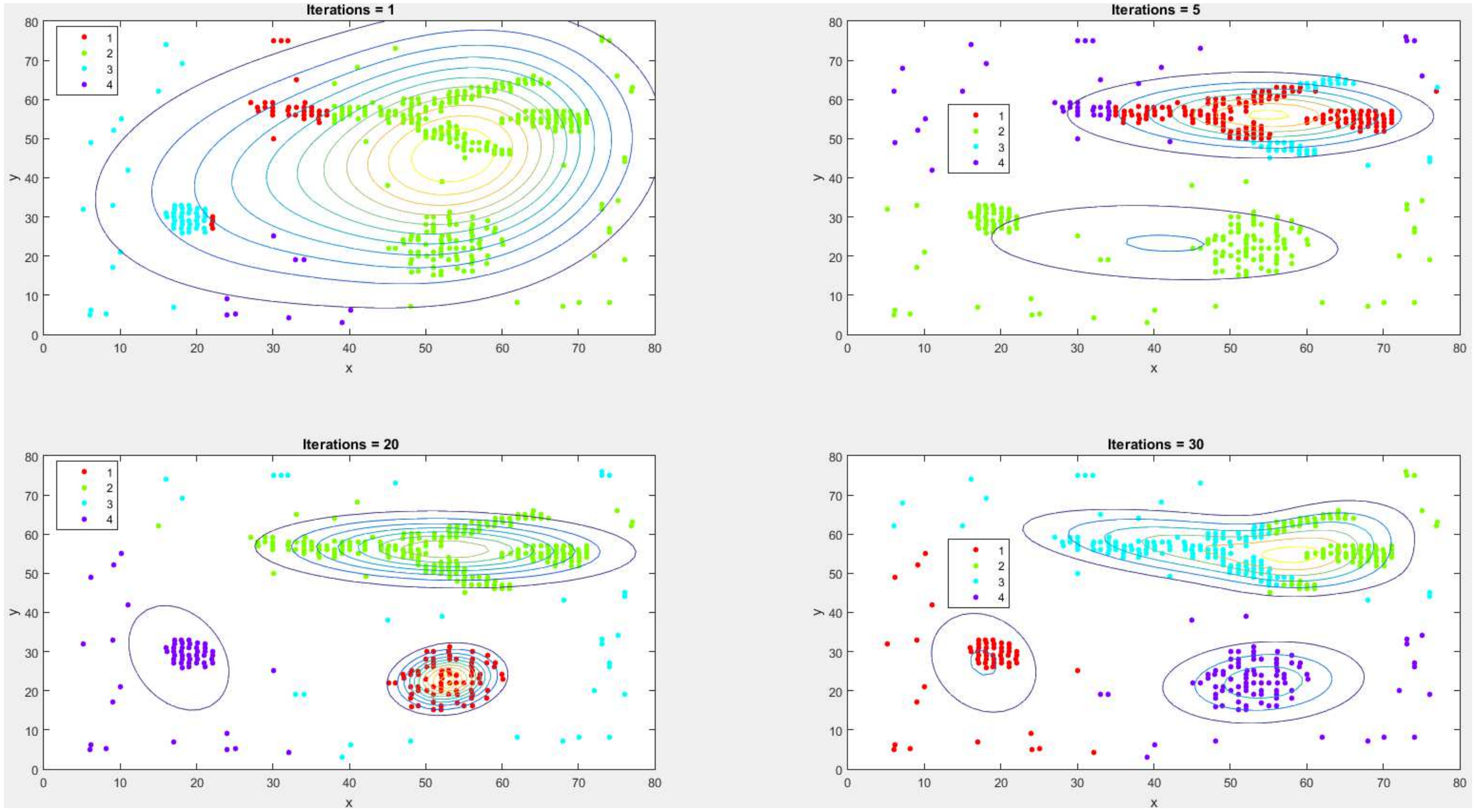
- ***P***(***c*** | ***x***) $= \frac{P(x \mid c)P(c)}{P(x)}$ (very similar to the ***Naïve Bayes formula***)
  - Very similar to the NB formula, except how $P(x \mid c)$ is calculated.
  - ***P***(***x***) is a normalization constant. (and it does not need to be actually computed!)
  - ***P***(***c*** | ***x***) = the product of
    - $P(c)$ the prior probability of class ***c***, and
    - $P(x \mid c)$ the probability of an instance ***x*** provided (in) class ***c***.
      - A **normal density function** in class ***c*** w/ mean $\boldsymbol{\mu}$ & covariance $\boldsymbol{\sigma}^2$ at point ***x***.

$$P(x \mid c) = f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- ***M***aximization: Adjust GMMs (i.e. $\boldsymbol{\mu_j}$ & $\boldsymbol{\Sigma_j}$) based on weights.

$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

Y-Shape Data



Iterations = 1
Iterations = 5
Iterations = 20
Iterations = 30
1
2
3
4
x
y