ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Introduction to Machine Learning

Dealing with missing data
(Unsupervised learning via prob. modeling)

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

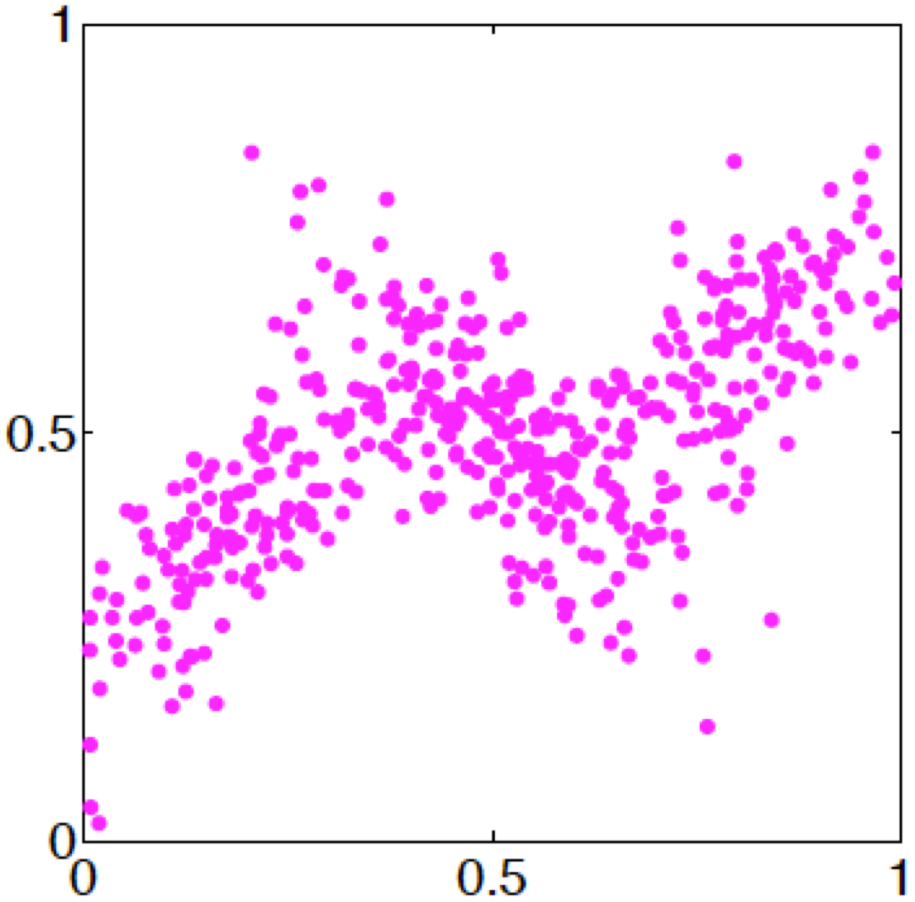
Missing data

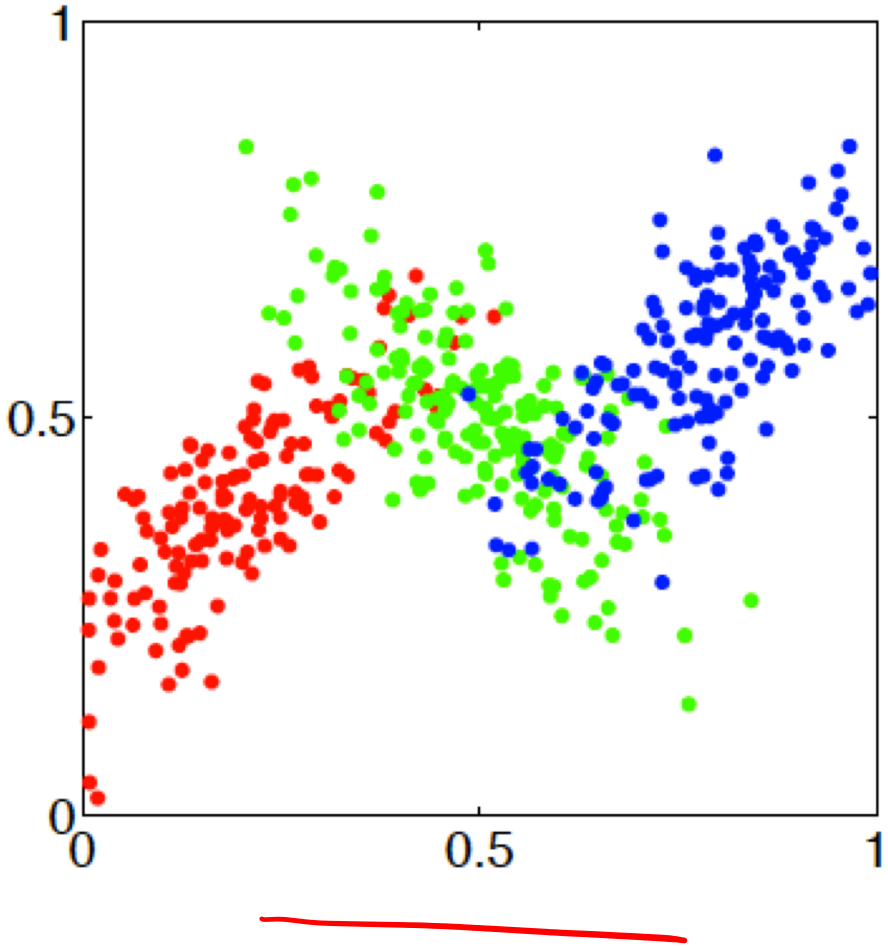
X_1	X_2	…	X_d	Y
1.1	N/A	…	.5	1
N/A	.4	…	.7	1
…	…	…	…	…
-.4	-.5	…	N/A	7

Missing labels

X_1	X_2	…	X_d	Y
1.1	.3	…	.5	N/A
.5	.4	…	.7	N/A
…	…	…	…	…
-.4	-.5	…	.3	N/A

What if we assume P(X | Y) is Gaussian?



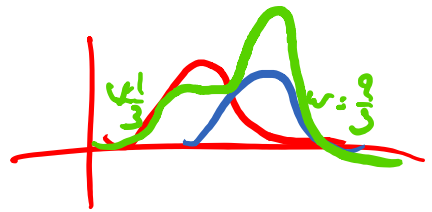


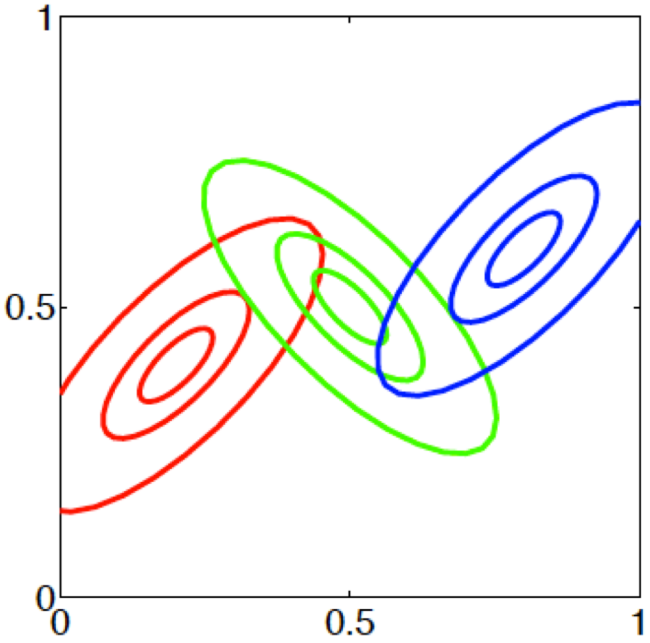
Gaussian mixtures

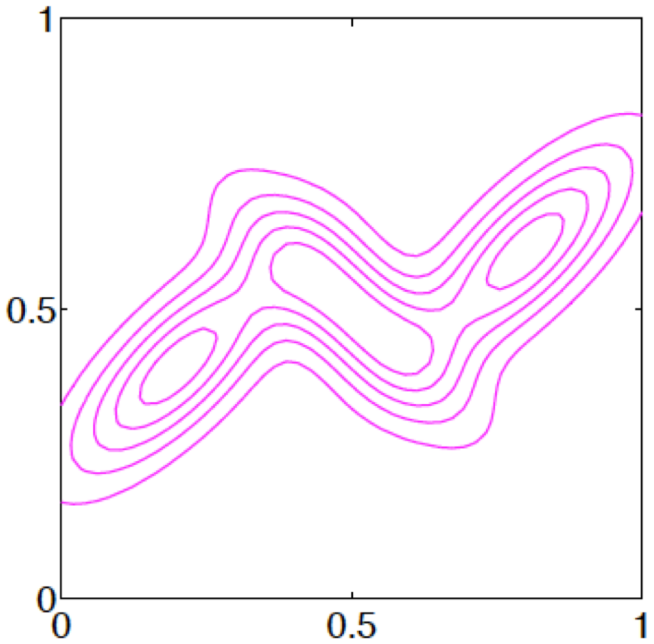
- Convex-combination of Gaussian distributions

$$P(\mathbf{x} \mid \theta) = P(\mathbf{x} \mid \mu, \Sigma, \mathbf{w}) = \sum_{i} w_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$$

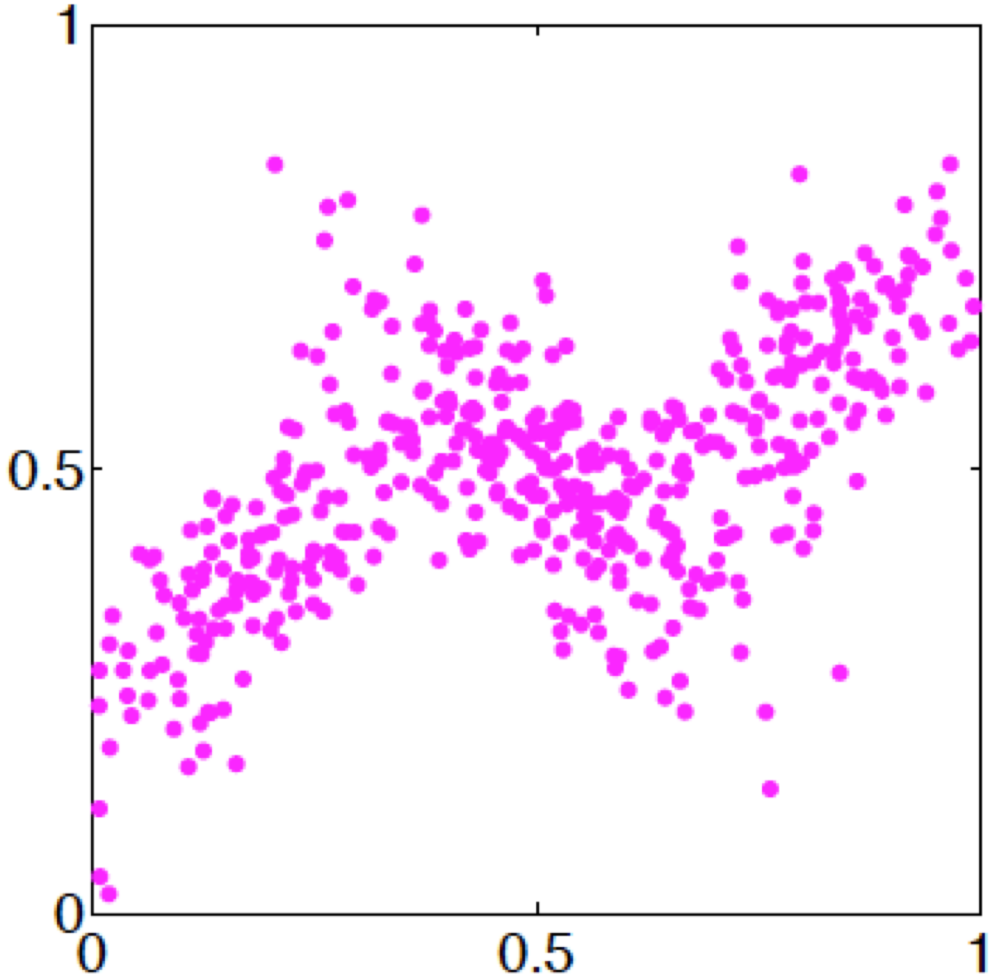
where $w_i \geq 0$ and $\sum_i w_i = 1$







Fitting a mixture model



$$(\mu^*, \Sigma^*, w^*) = \arg\min - \sum_i \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x_i} \mid \mu_j, \Sigma_j)$$

MLE for Gaussian mixtures

- Would like to minimize

$$L(w_{1:k}, \mu_{1:k}, \Sigma_{1:k}) = -\sum_{i=1}^{n} \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x}_i \mid \mu_j, \Sigma_j)$$

- This is a nonconvex objective
- Could still try to apply (stochastic) gradient descent
- Challenges
  - Covariance matrices must remain symmetric positive definite
  - These constraints might be difficult to maintain

- In the following, will discuss an alternative approach

GMMs vs. Gaussian Bayes Classifiers

- The joint distribution $P(z, \mathbf{x}) = w_z \mathcal{N}(\mathbf{x} \mid \mu_z, \Sigma_z)$ of *(cluster_index,features)* is identical to the generative model used by the Gaussian Bayes Classifier

- **Main difference**: In contrast to GBCs, in GMMs the (label) variable z is *unobserved*!

➔ Fitting a GMM = Training a GBC without labels

➔ Clustering = latent variable modeling

➔ If we could just get the labels, could compute MLE in closed form!

The Hard-EM algorithm

- Initialize the parameters $\theta^{(0)}$
- For t=1,2,...
  - E-step: Predict most likely class for each data point.

$$z_i^{(t)} = \arg\max_z P(z \mid \mathbf{x}_i, \theta^{(t-1)})$$

$$= \arg\max_z P(z \mid \theta^{(t-1)}) P(\mathbf{x}_i \mid z, \theta^{(t-1)})$$

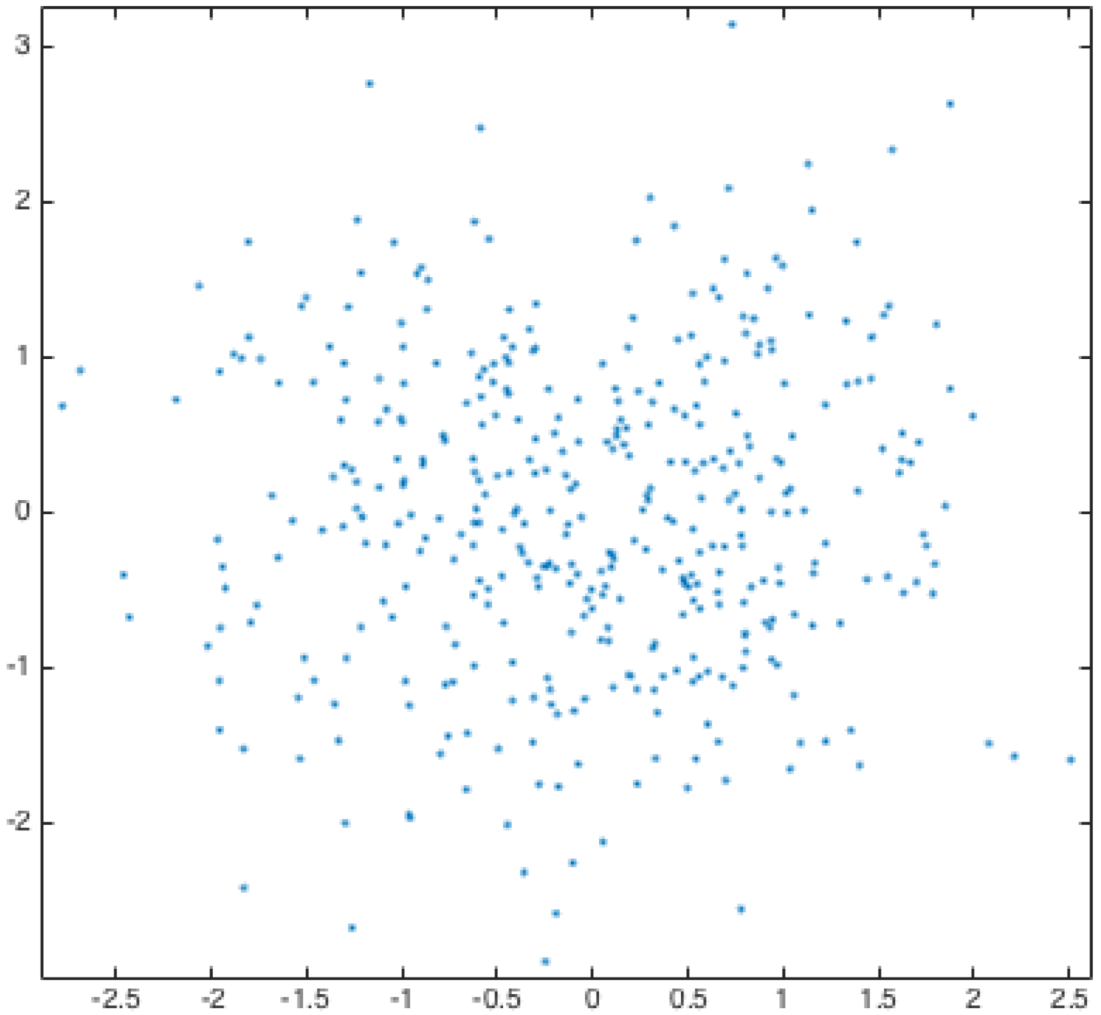
  - Now we got complete data!

$$D^{(t)} = \{(\mathbf{x}_1, z_1^{(t)}), \ldots, (\mathbf{x}_n, z_n^{(t)})\}$$

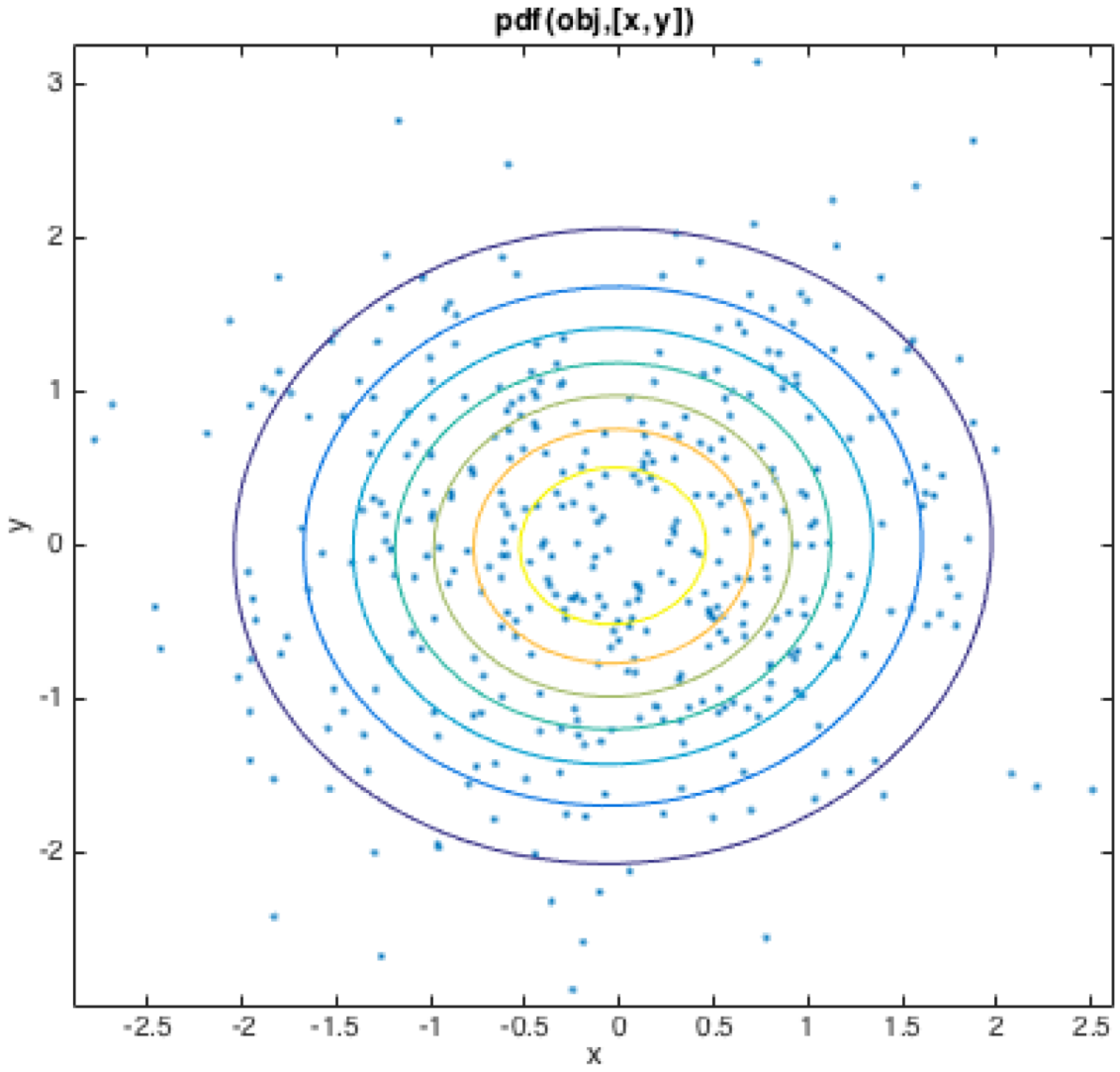
  - M-step: Compute MLE as for the Gaussian Bayes classifier

$$\theta^{(t)} = \arg\max_\theta P(D^{(t)} \mid \theta)$$

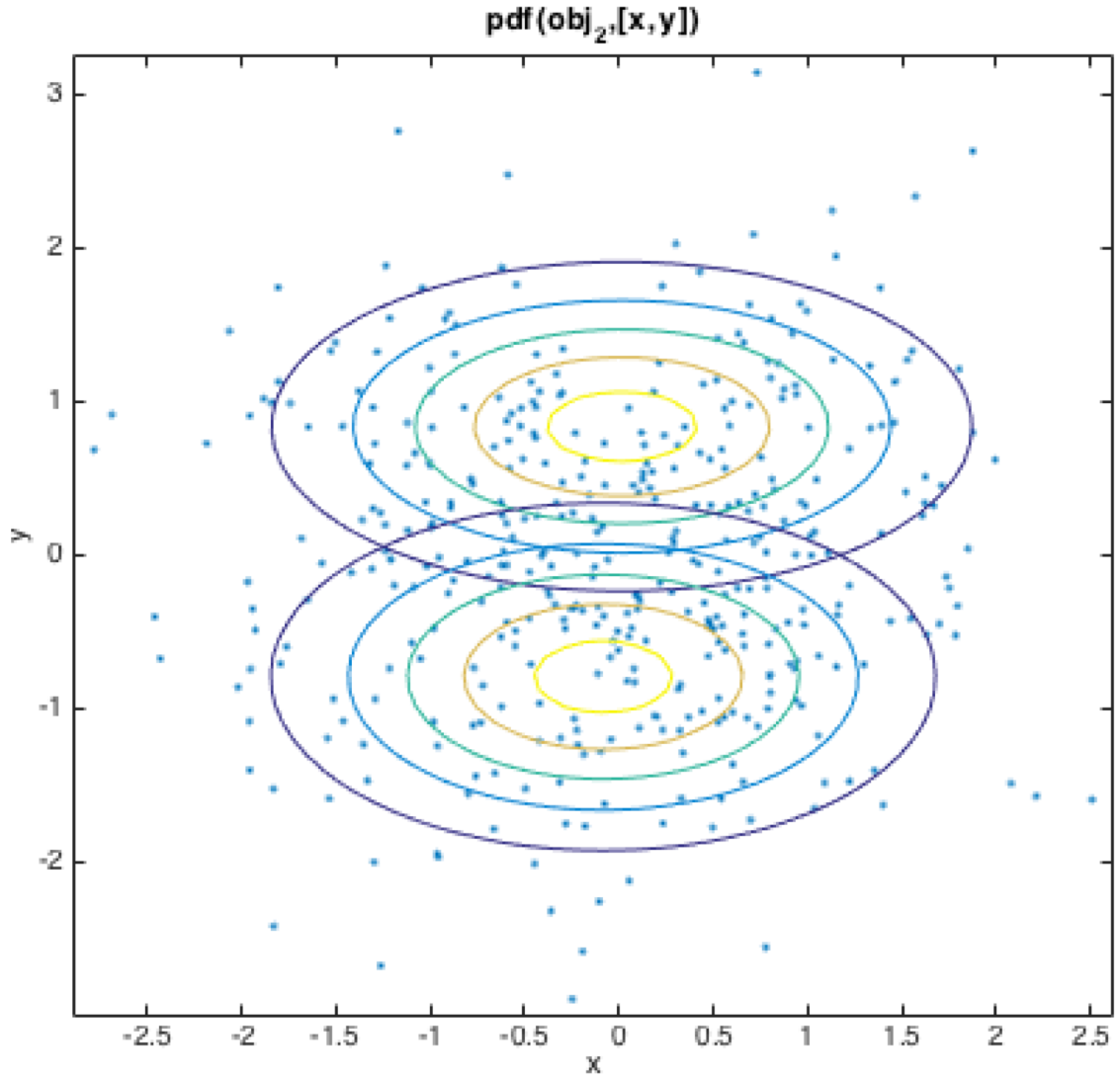
Practical issues with Hard-EM



Model generating the data: single Gaussian



Fit with Hard-EM



Problems with this approach

- Points are assigned a fixed label, even though the model is uncertain
- Intuitively, this tries to extract «too much information» from a single point
- In practice, this may work poorly if clusters are overlapping (more later)

Posterior probabilities

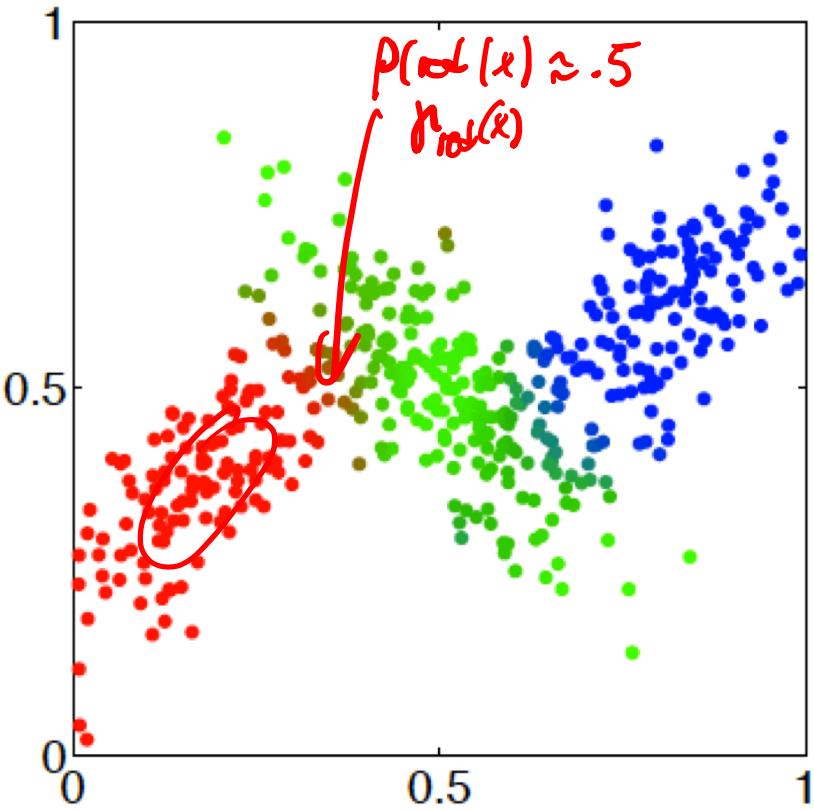
$\theta = (w, \mu, \Sigma)$

- Suppose we're given a model $P(z \mid \theta), P(\mathbf{x} \mid z, \theta)$
- Then, for each data point, we can compute a posterior distribution over cluster membership
- This means inferring distributions over latent (hidden) variables z

$$\gamma_j(\mathbf{x}) = P(Z = j \mid \mathbf{x}, \Sigma, \mu, \mathbf{w})$$

Bayes' rule

$$= \frac{w_j P(\mathbf{x} \mid \Sigma_j, \mu_j)}{\sum_\ell w_\ell P(\mathbf{x} \mid \Sigma_\ell, \mu_\ell)}$$



Maximum likelihood estimation

- Can show: At MLE

$$(\mu^*, \Sigma^*, w^*) = \arg\min - \sum_i \log \sum_{j=1}^{k} w_j \mathcal{N}(\mathbf{x_i} \mid \mu_j, \Sigma_j)$$

it must hold that

$$\mu_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$\Sigma_j^* = \frac{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)(\mathbf{x}_i - \mu_j^*)(\mathbf{x}_i - \mu_j^*)^T}{\sum_{i=1}^n \gamma_j(\mathbf{x}_i)}$$

$$w_j^* = \frac{1}{n} \sum_{i=1}^{n} \gamma_j(\mathbf{x}_i)$$

These equations are *coupled* → difficult to solve jointly

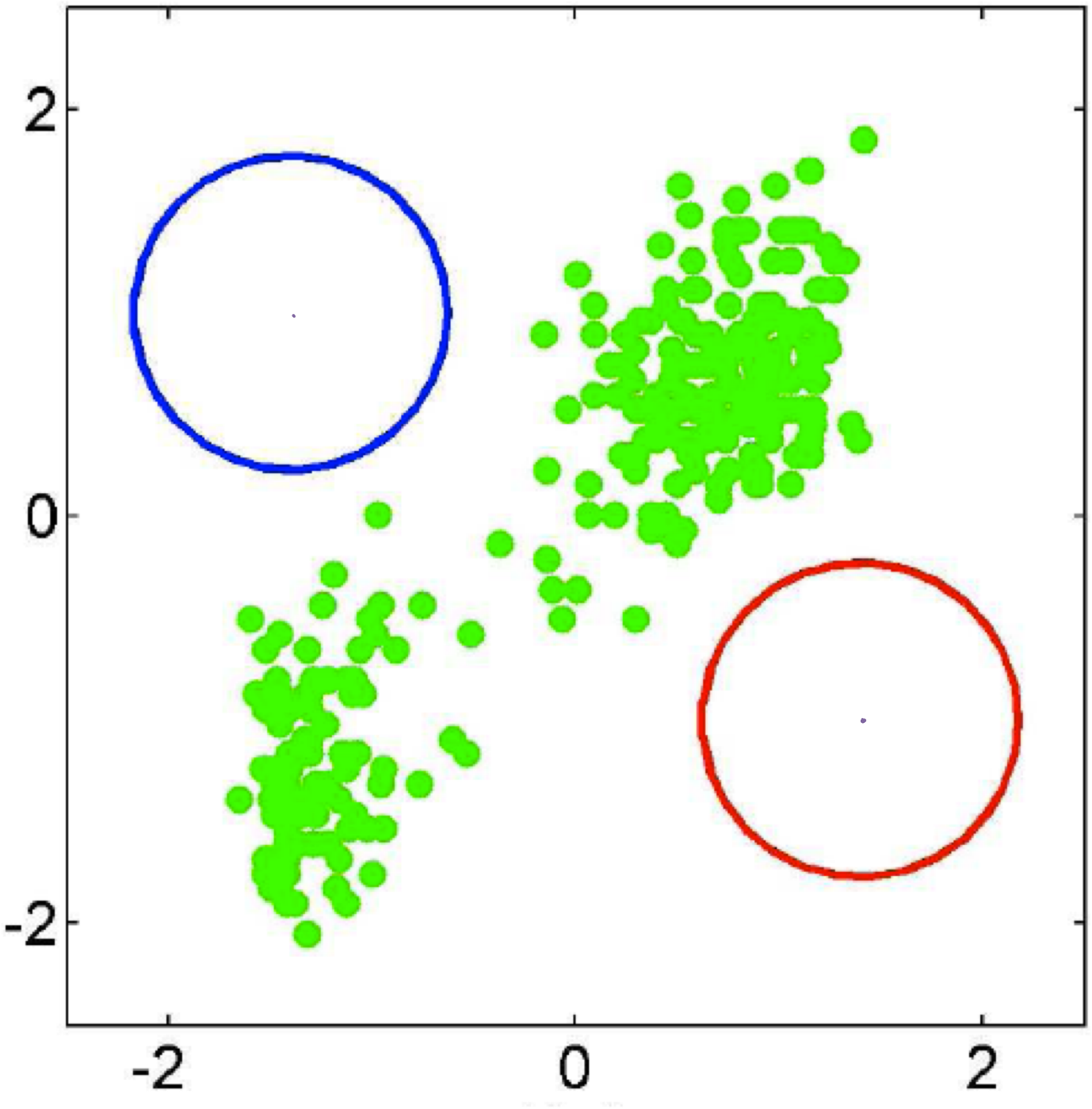
Expectation-Maximization (Soft-EM)

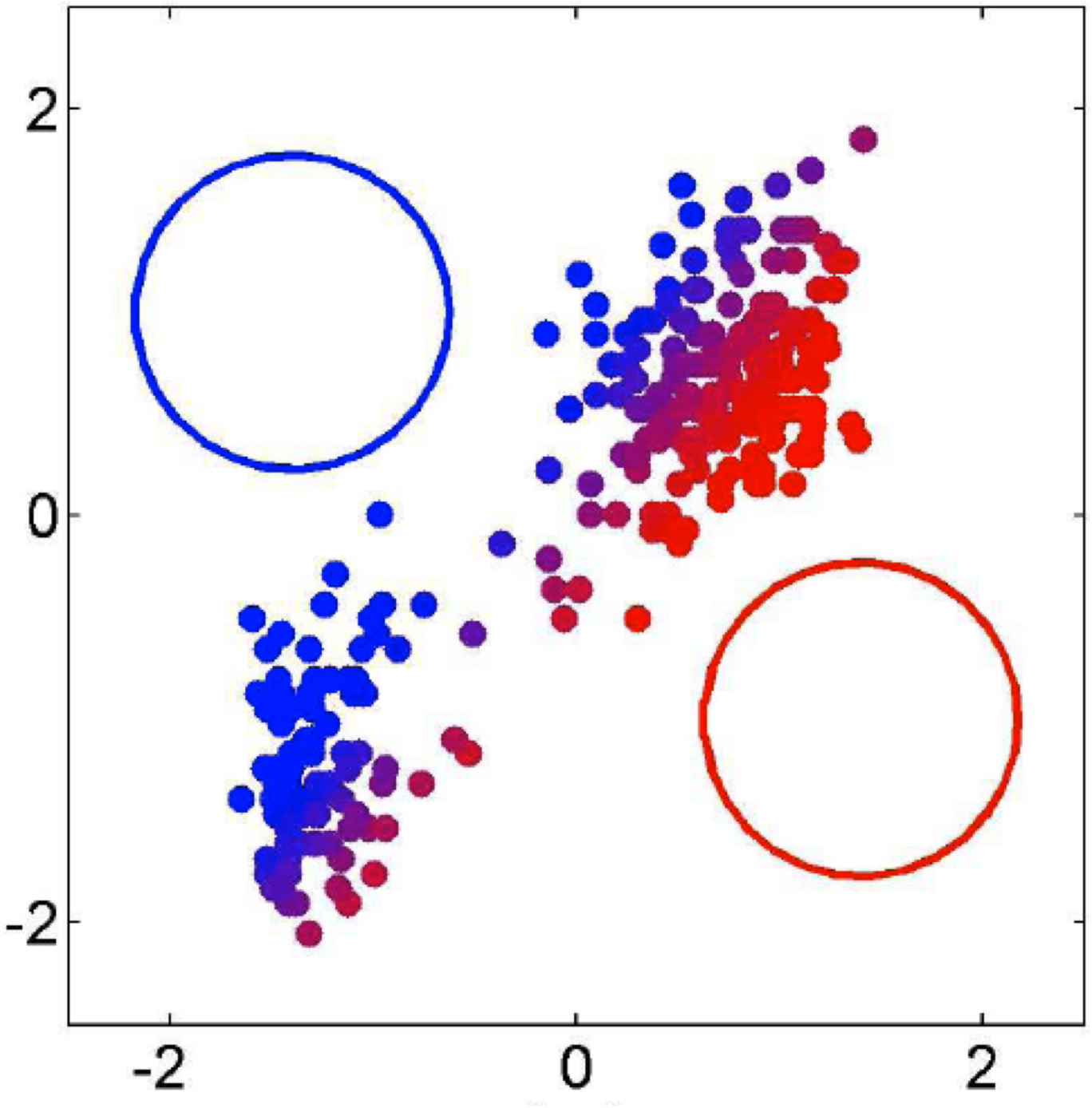
- While not converged
  - E-step: calculate cluster membership weights ("Expected sufficient statistics") for each point (aka "responsibilities"):

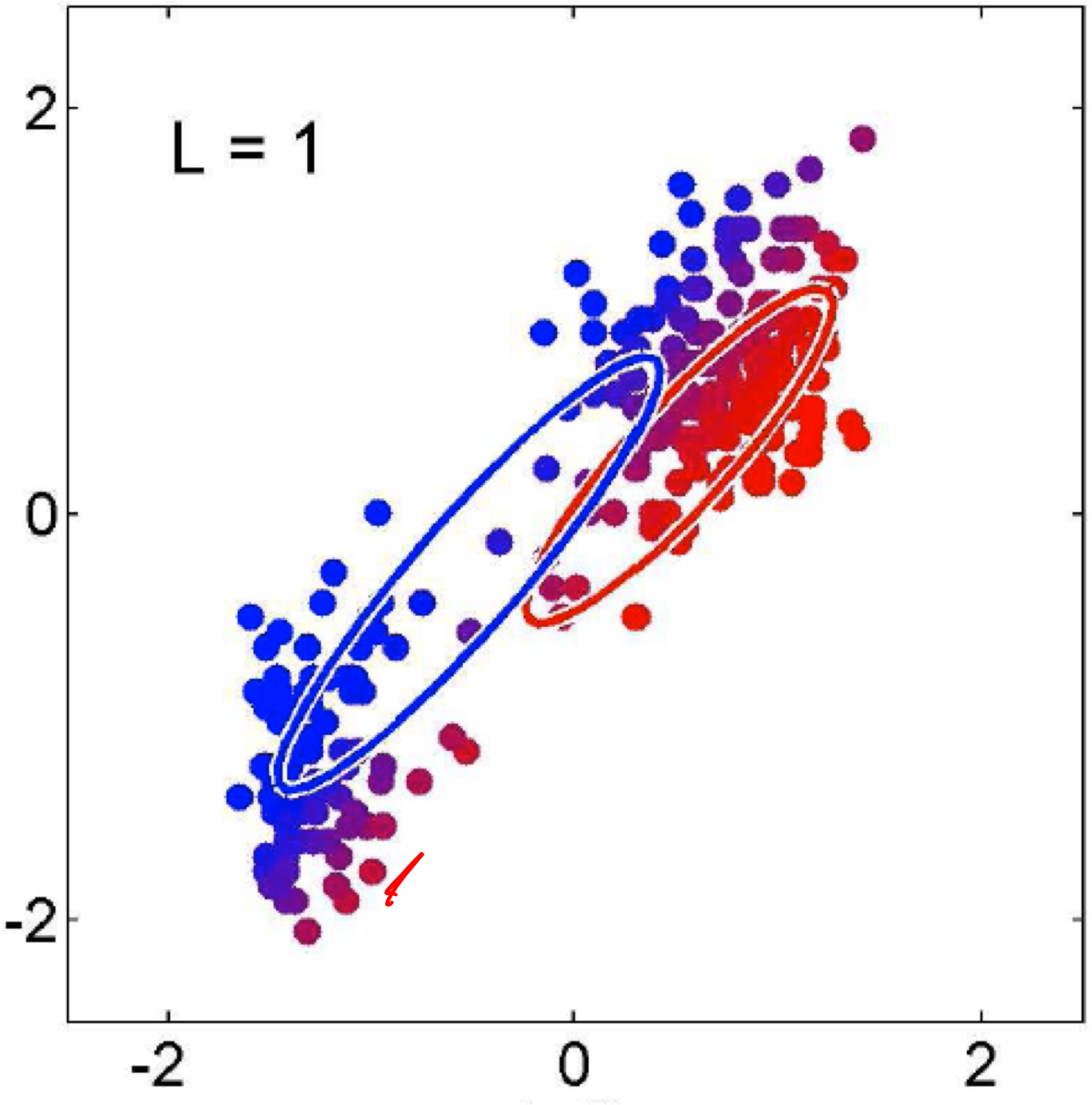
    Calculate $\gamma_j^{(t)}(\mathbf{x}_i)$ for each *i* and *j* given estimates of $\mu^{(t-1)}, \Sigma^{(t-1)}, \mathbf{w}^{(t-1)}$ from previous iteration
  - M-step: Fit clusters to weighted data points (closed form Maximum likelihood solution!)

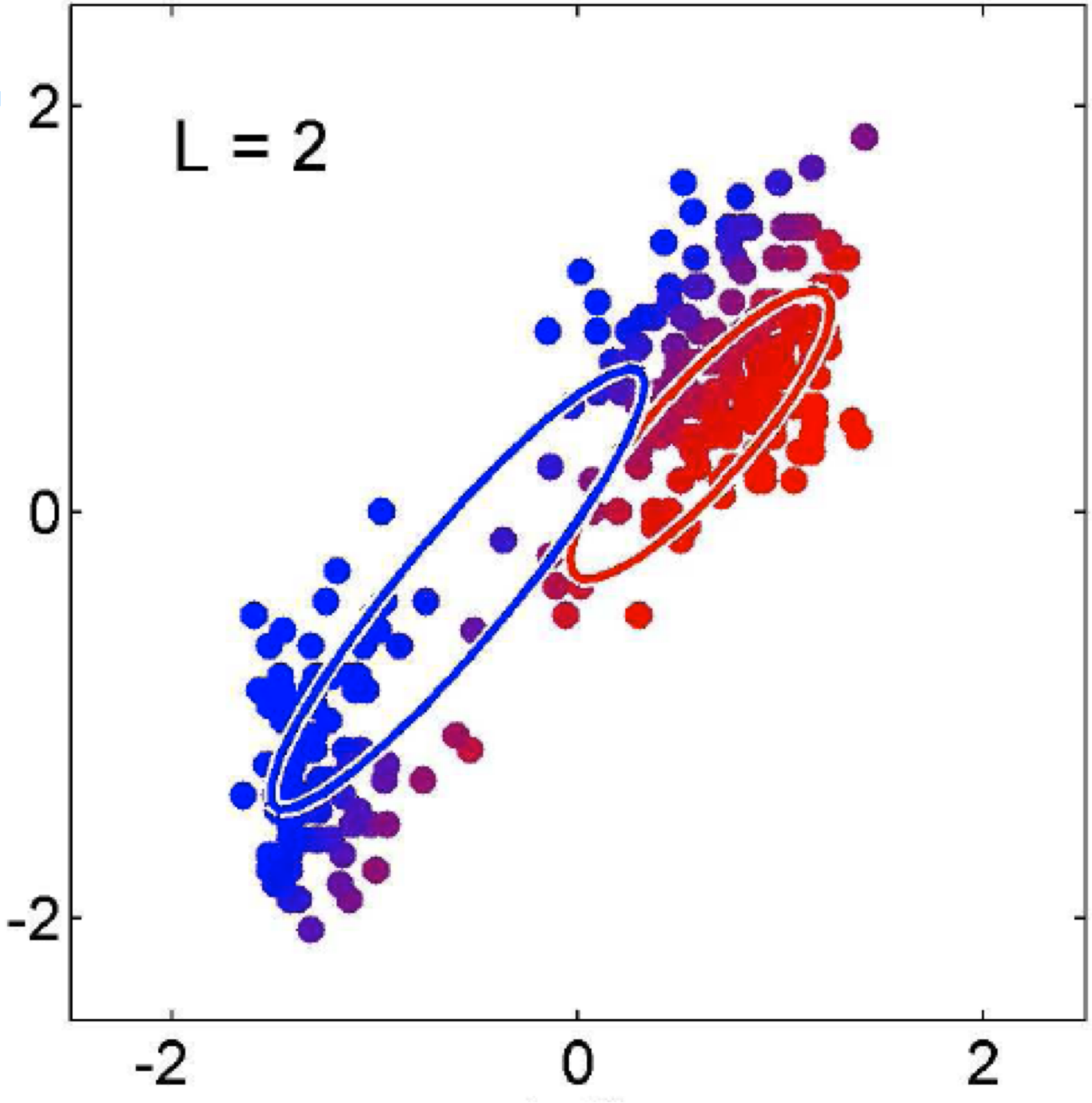
$$w_j^{(t)} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i) \quad \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)}$$

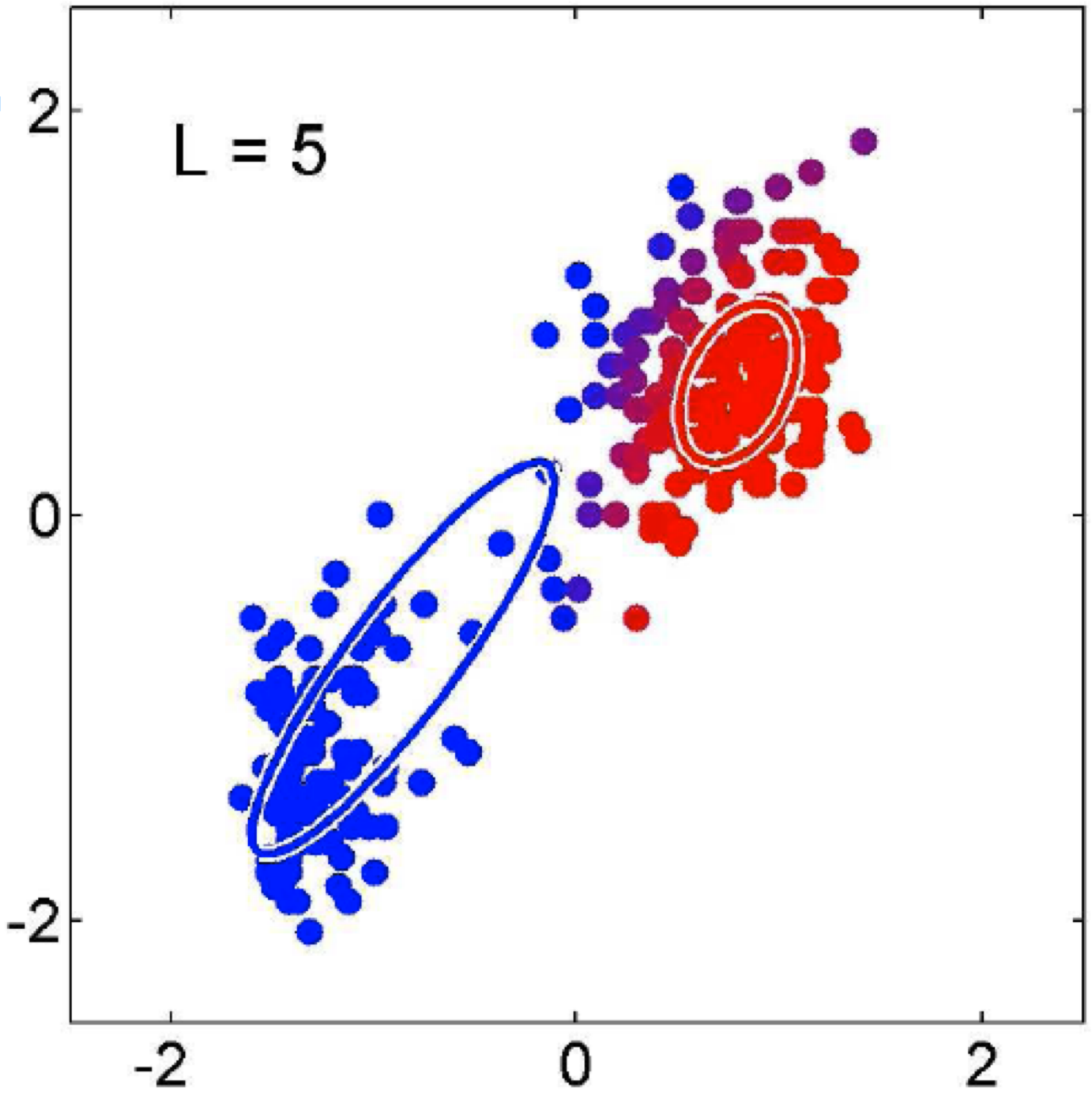
$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)(\mathbf{x}_i-\mu_j^{(t)})(\mathbf{x}_i-\mu_j^{(t)})^T}{\sum_{i=1}^{n}\gamma_j^{(t)}(\mathbf{x}_i)}$$

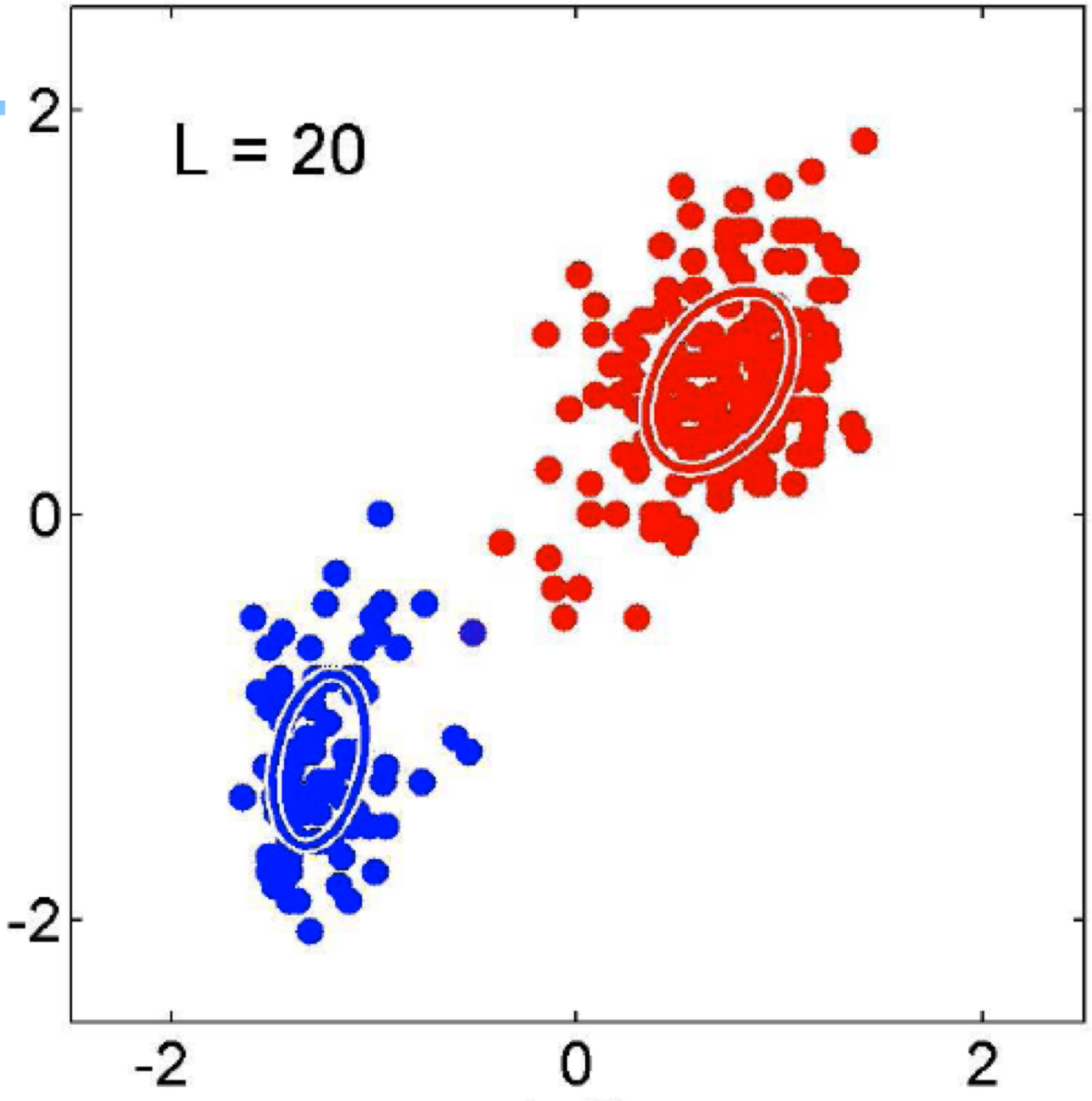

2
0
-2
-2
0
2


2
0
-2
-2
0
2

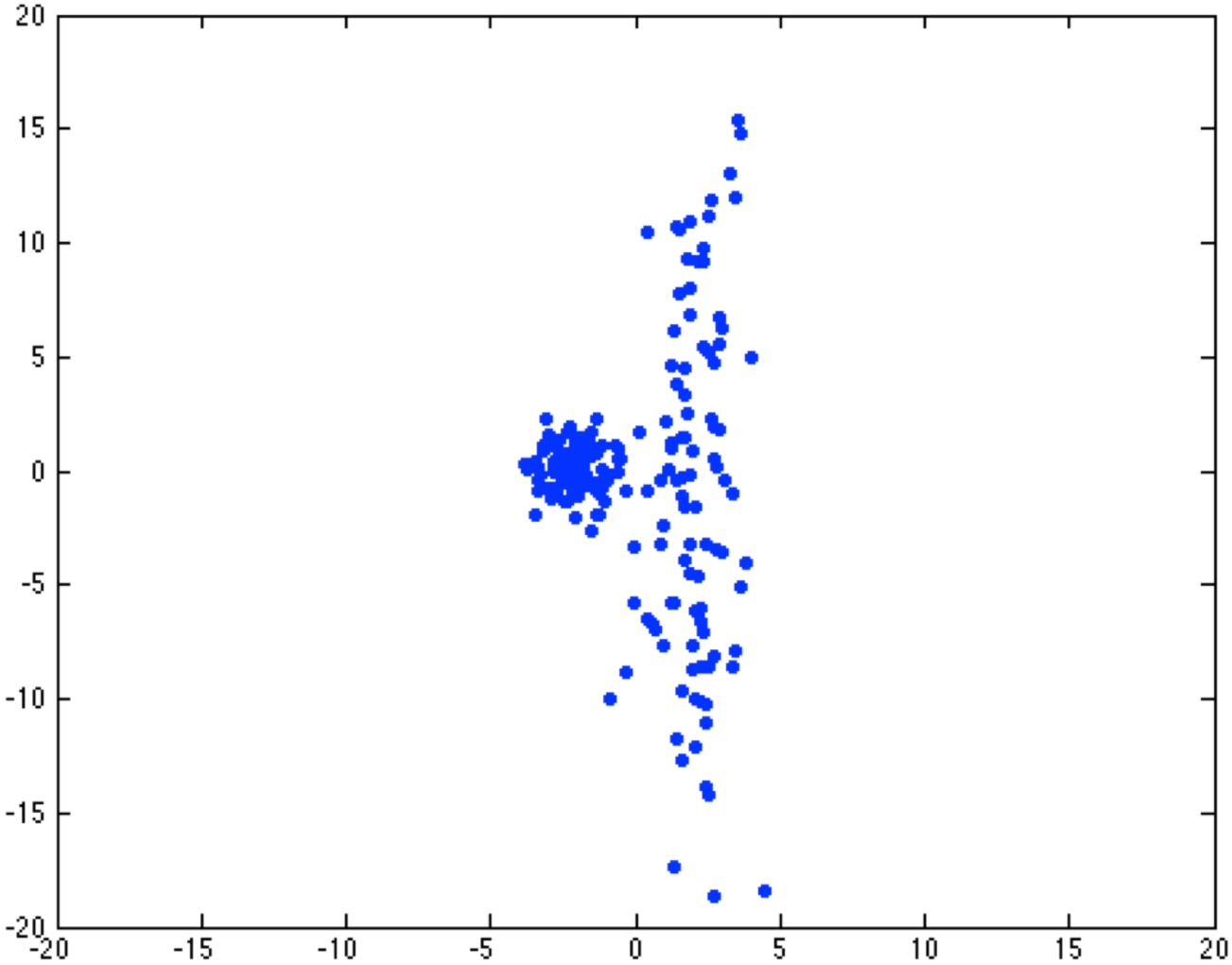

L = 1
2
0
-2
-2
0
2


L = 2
2
0
-2
-2
0
2

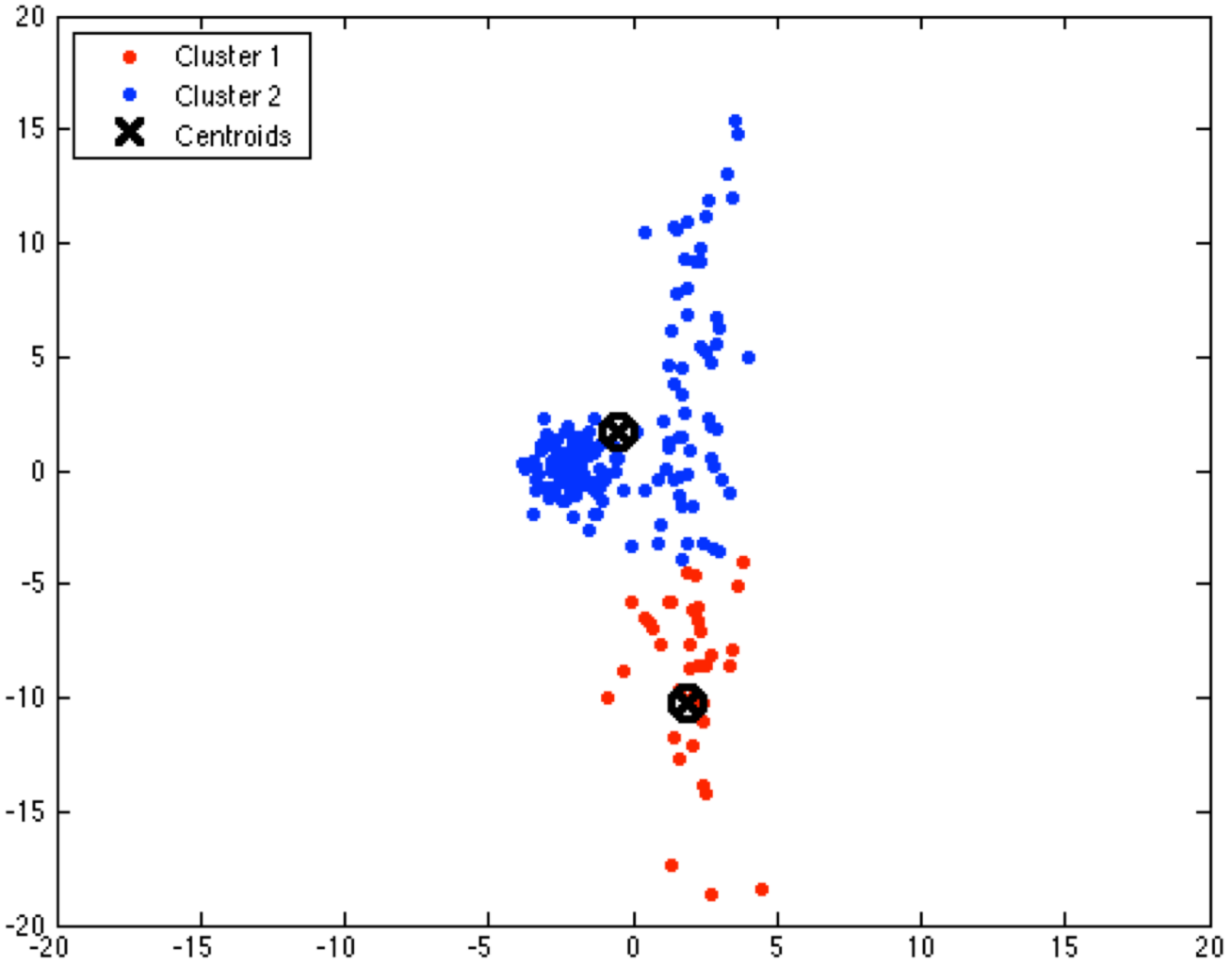

L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

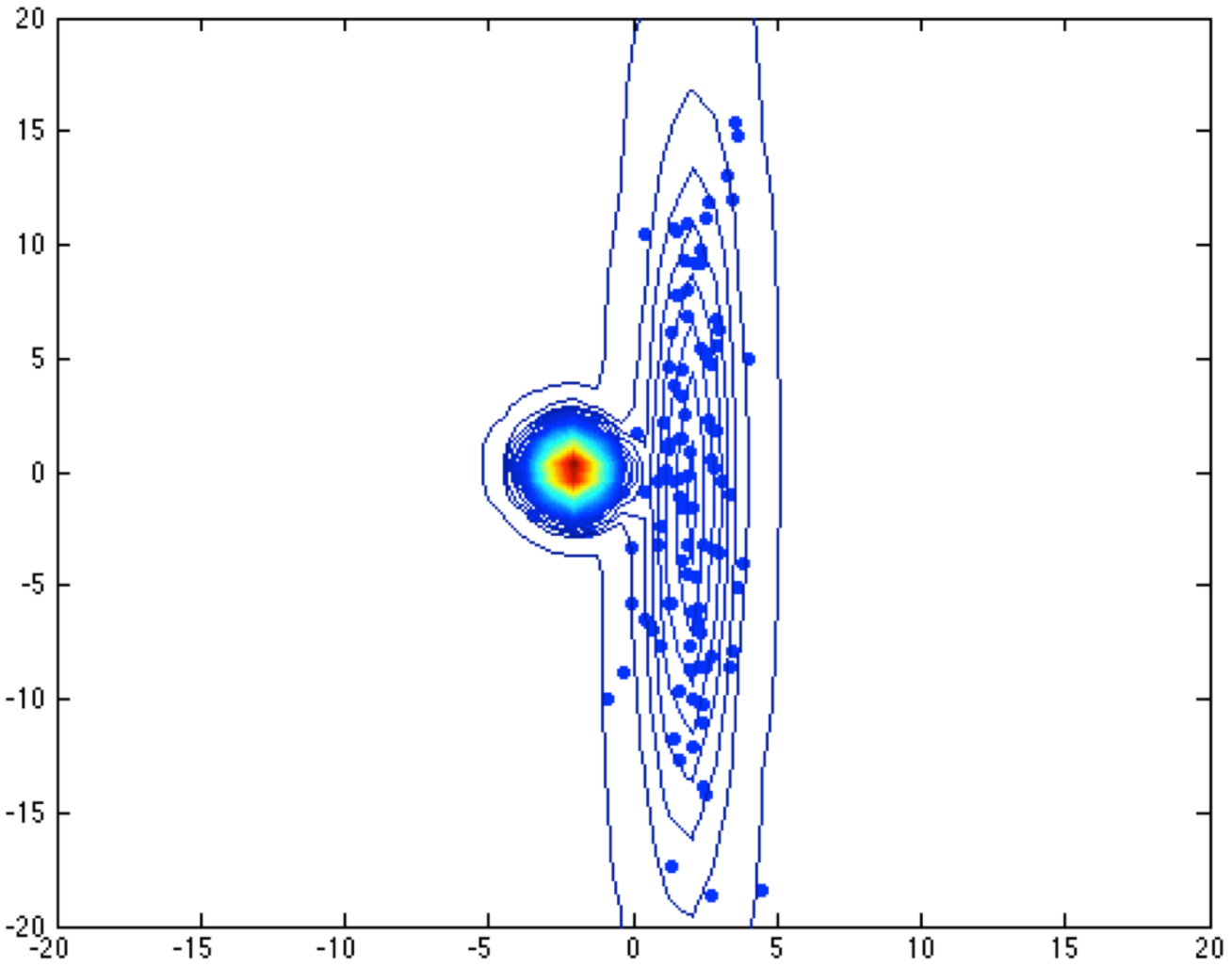
What will k-means do on this data?



Best solution found by k-Means



Example fit with GMMs



Note: Constrained GMMs

- Can consider special cases of Gaussian mixtures
  - Spherical $\Sigma_j = \sigma_j^2 \cdot I_d$
  - Diagonal $\Sigma_j = \begin{pmatrix} \sigma_{j,1}^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_{j,d}^2 \end{pmatrix}$ a.k.a Gaussian Naïve Bayes
  - Tied $\Sigma_1 = \dots \Sigma_C$
  - Full
- Done by tying together parameters in MLE

EM GMM Demo

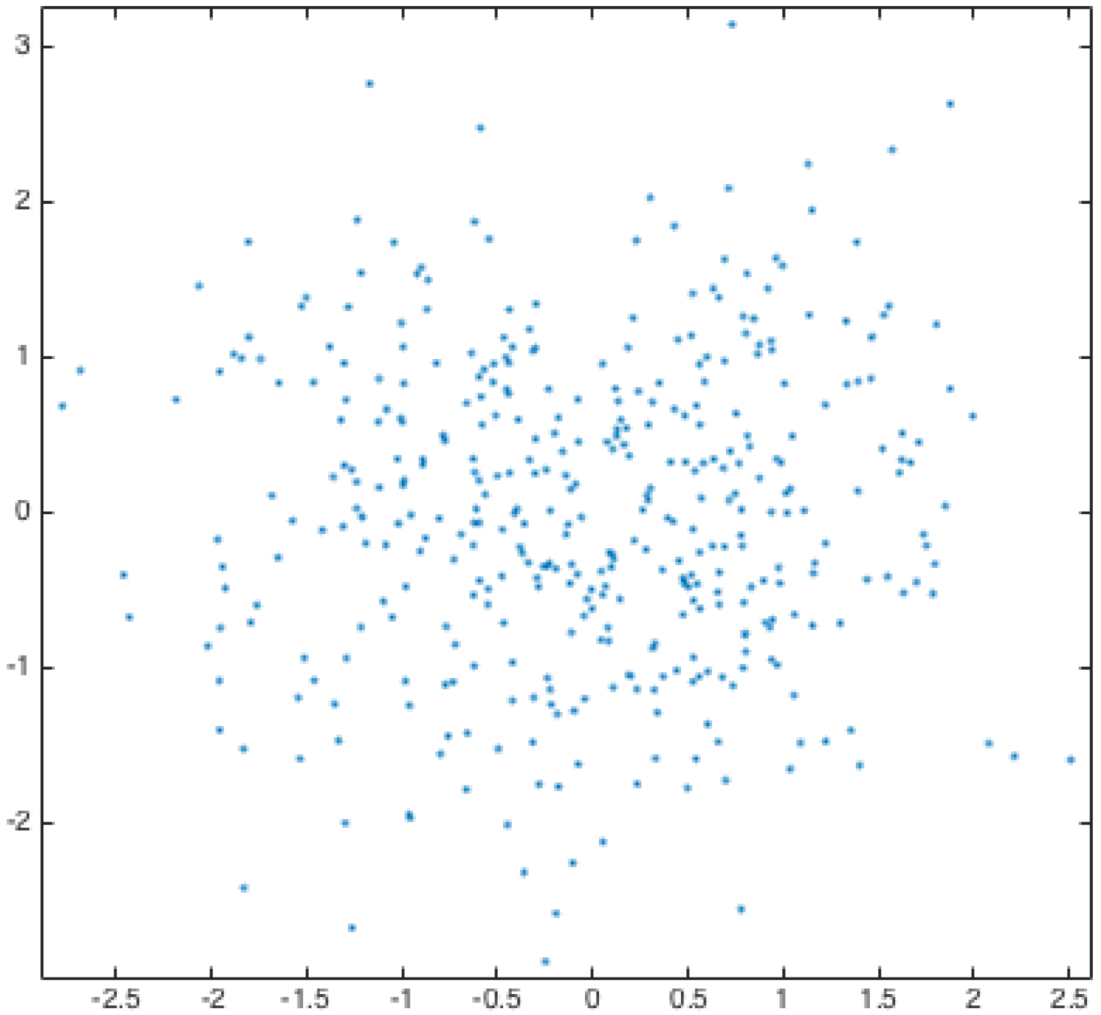
Soft EM vs Hard EM

- The EM algorithm we discussed uses soft assignments to clusters, and is hence called «Soft EM»
- The intuitive first attempt we discussed first uses hard assignments, and is called «Hard EM»

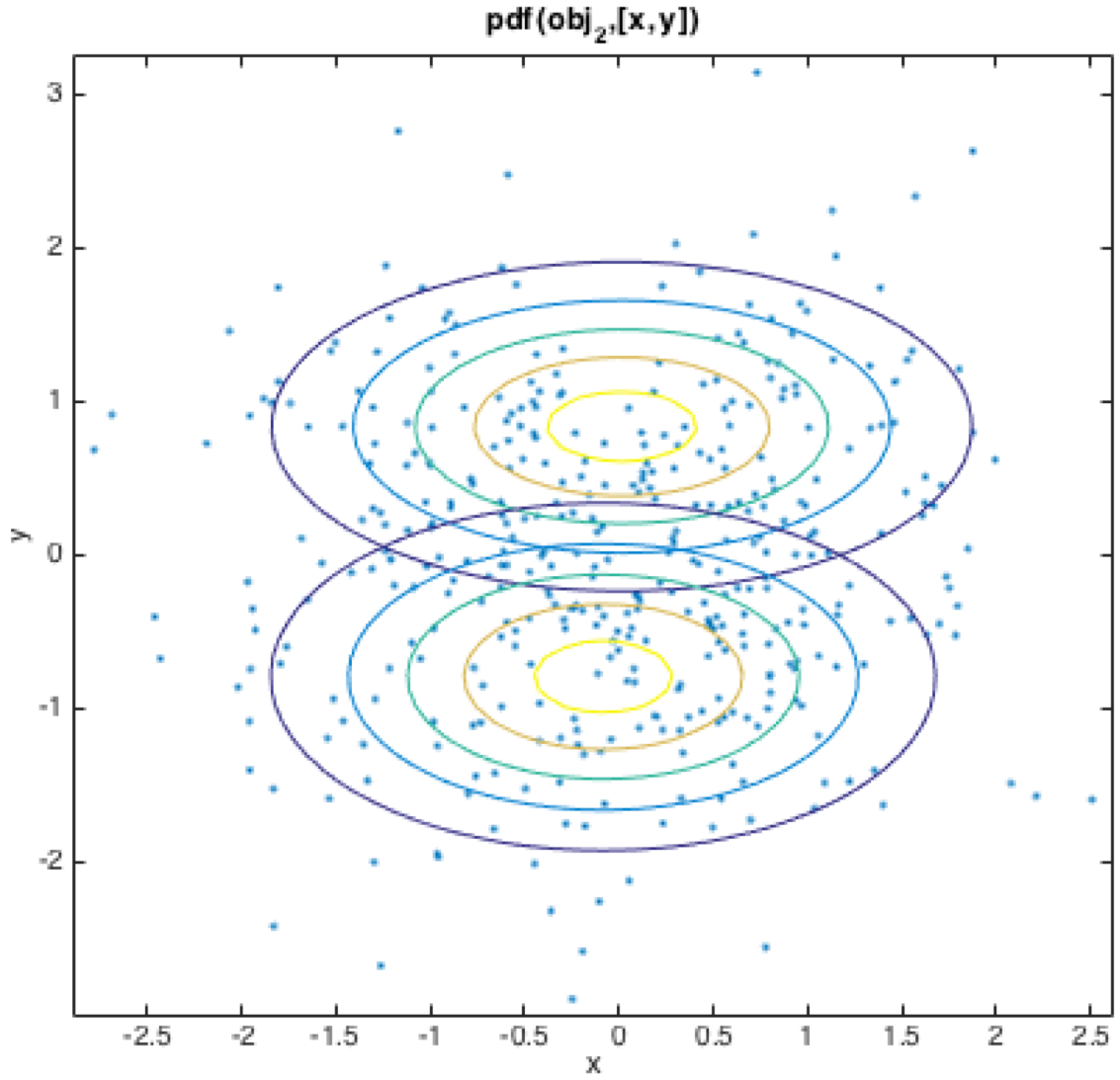
- In general, Soft EM will typically result in higher likelihood values
- Reason: It can deal better with «overlapping clusters»

- The term EM alone typically refers to Soft EM

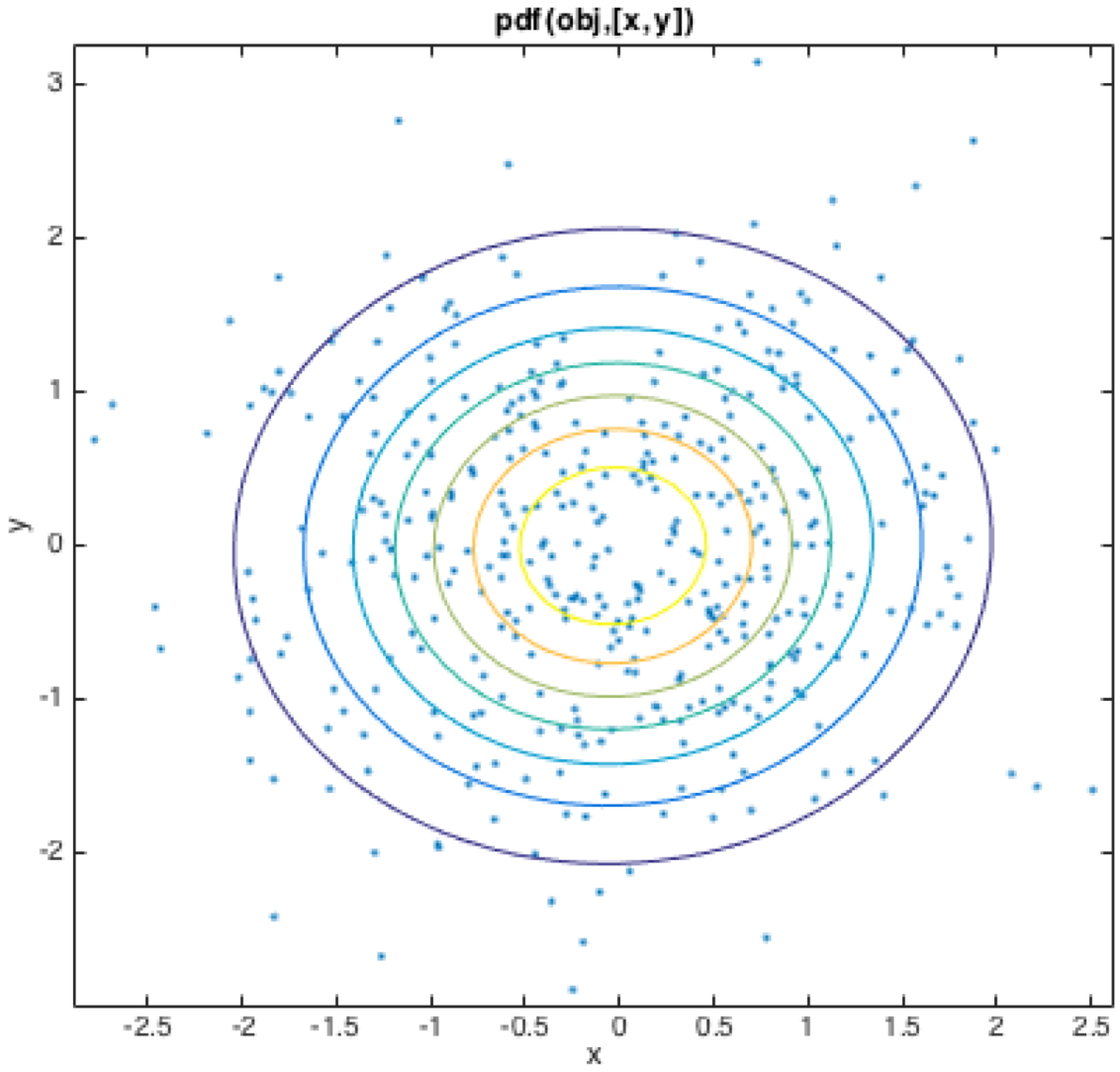
Practical issues with Hard-EM



Fit with Hard-EM



Fit with Soft-EM



Hard EM with uniform weights and spherical covariances

Suppose: $w_1 \dots w_k = \frac{1}{k}$; $\Sigma_{1:k} = I \cdot \sigma^2$

Consider Hard EM-algorithm

E-step In iteration t for point x_i

$$z_i^{(t)} = \underset{z \in \{1..k\}}{\operatorname{argmax}} P(z \mid x_i, \theta^{(t-1)}) = \underset{z}{\operatorname{argmax}} \; w_z \, \mathcal{N}(x_i; \mu_z^{(t-1)}, \Sigma_z^{(t-1)})$$

($\Sigma_z^{(t-1)} = \sigma^2 \cdot I$)

$$= \underset{z}{\operatorname{argmax}} \; \frac{1}{k} \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x_i - \mu_z^{(t-1)}\|_2^2}{2\sigma^2}\right)$$

$$= \underset{z}{\operatorname{argmin}} \; \frac{1}{2\sigma^2} \|x_i - \mu_z^{(t-1)}\|_2^2 = \underset{z}{\operatorname{argmin}} \; \|x_i - \mu_z^{(t-1)}\|_2$$

Same decisions as k-means!

M-step For MLE $\mu_j^{(t)} = \frac{1}{n_j} \sum_{i: z_i^{(t)} = j} x_i$

k-Means Algorithm vs. EM for GMM

- Can understand k-Means Algorithm (Lloyd's heuristic) as special case of Hard-EM for GMMs:
  - Uniform weights over mixture components
  - Assuming identical, spherical covariance matrices
- Can also understand k-Means Algorithm as limiting case of Soft-EM for GMM:
  - Assumptions same as above, with additionally variances tending to 0

$\Sigma_z = \sigma^2 \cdot I$

Why? As $\sigma \to 0$, it holds that

$$\gamma_i(k) \to \begin{cases} 1 & \text{if } \mu_i \text{ is (unique) closest to } x \\ 0 & \text{if not (and ties)} \end{cases}$$

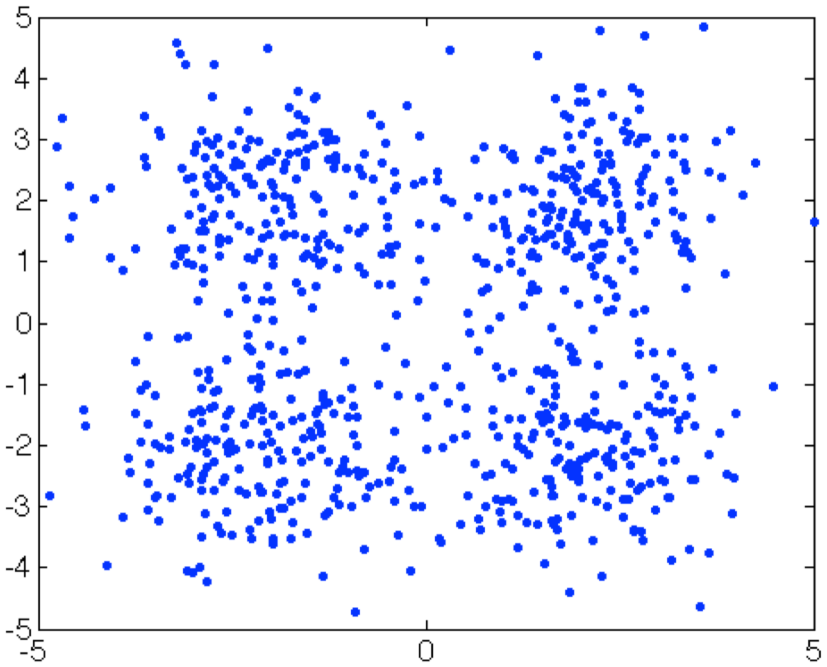
Initialization

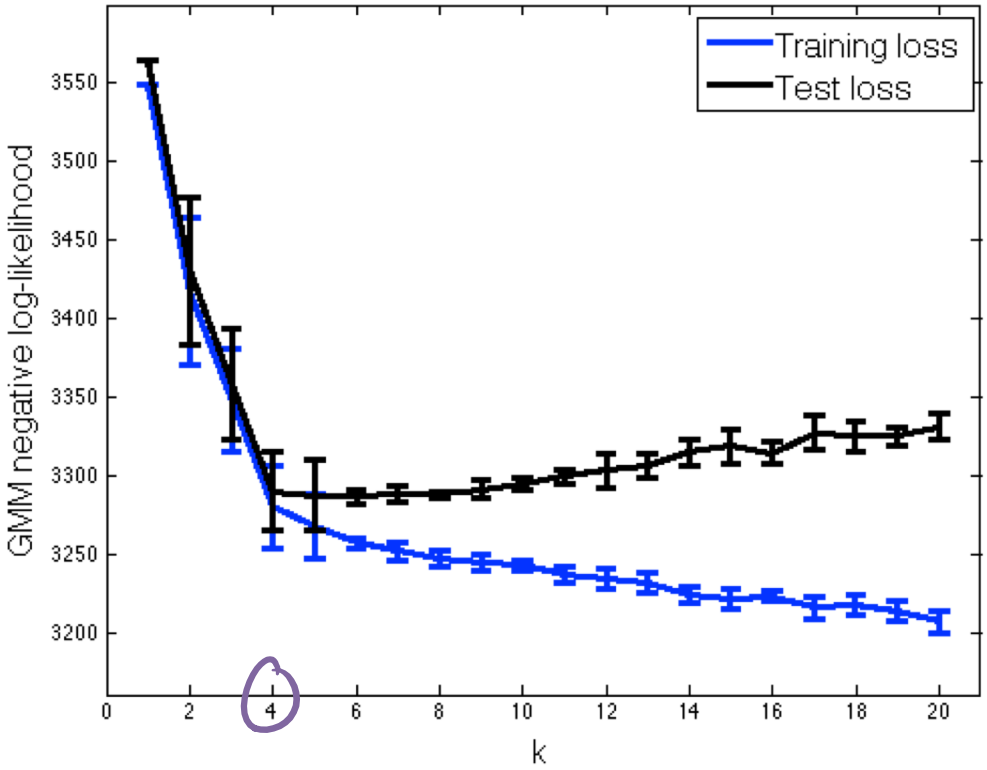
- How should we initialize the parameters?
- For weights:
  - Typically use uniform distribution
- For means:
  - Randomly initialize
  - Use k-means++
- For variances
  - Initialize as spherical, e.g., according to empirical variance in the data

Selecting *k*

- In GMMs, we face a similar challenge of selecting the number of clusters
- Can generally use the same techniques
- However, in contrast to k-means, for GMMs typically cross-validation works fairly well
  - Aim to maximize log-likelihood on validation set

Selecting *k*





- If data is truly generated from a GMM, cross-validation can be used to select *k*

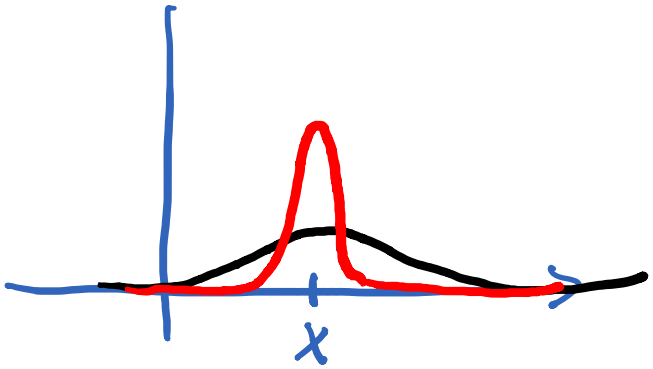
Degeneracy of GMMs

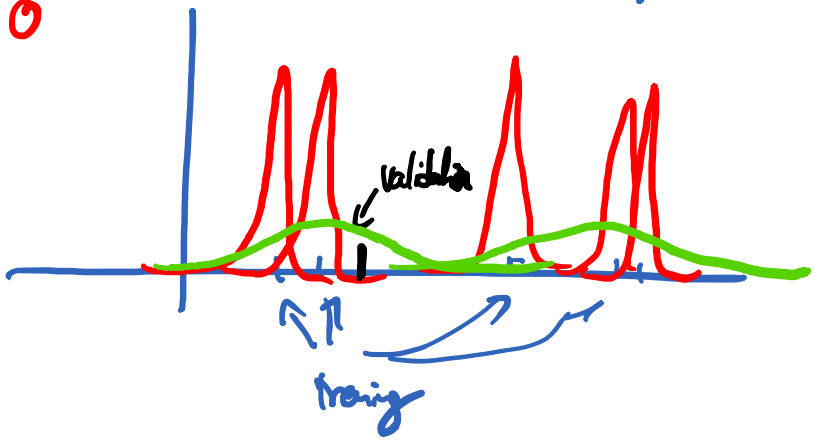
- Suppose we are given a single data point. What is the optimal log-likelihood that can be achieved? (1D)

$$-\log P(x \mid \mu, \sigma) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(x-\mu)^2$$

$\to -\infty$ as $\sigma \to 0$

0 if $\mu = x$





Degeneracy of GMMs

- Suppose we are given a single data point. What is the optimal log-likelihood that can be achieved? (1D)

$$-\log P(x \mid \mu, \sigma) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(x-\mu)^2$$

- Loss converges to $-\infty$ as $\mu = x, \sigma \to 0$

- Thus, optimal GMM chooses *k*=*n*, and puts one Gaussian around each data point with variance tending to 0!

➔ Overfitting! This explains poor test set log-likelihood!

Avoiding degeneracy in GMMs

- Can avoid variances tending to 0 by simply adding a small term to the diagonal of the MLE:

$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)(\mathbf{x}_i - \mu_j^{(t)})(\mathbf{x}_i - \mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)} + \nu^2 \mathbf{I}$$

- Seems like a hack!
- Can be motivated from a Bayesian standpoint: This is equivalent to placing a (conjugate) Wishart prior on the covariance matrix, and computing the MAP instead of MLE to regularize
- Can choose ν by cross-validation