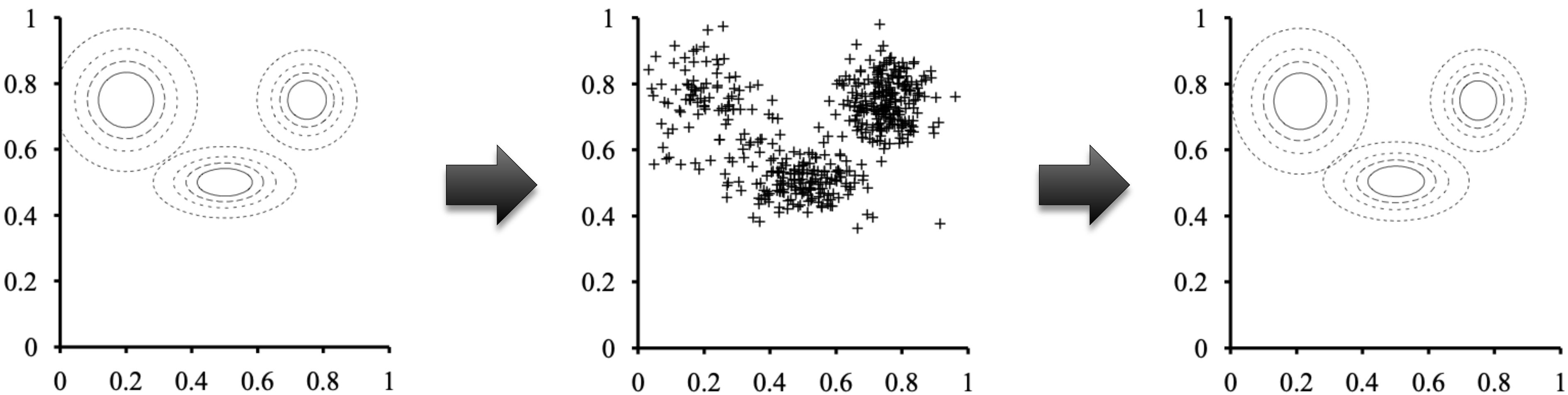
Data Mining & Machine Learning

CS37300
Purdue University

Nov 6, 2023

GMM Parameter Estimation



Learning the model from data

- We want to invert the generative process
- Given the dataset, find the parameters
  - Mixing coefficients $p(k)$
  - Component means and covariance matrix $N_k(\mu_k, \Sigma_k)$
- If we knew which component generated each point then the MLE solution would involve fitting each component distribution to the appropriate cluster points
- Problem: the cluster memberships are **hidden**

Expectation-maximization (EM) algorithm

- Popular algorithm for parameter estimation in data with hidden/unobserved values
    - Hidden variables=cluster memberships
- Basic idea
    - Initialize hidden variables and parameters
    - “Expectation” step: Estimate distributions of hidden variables given current estimates of the parameters

      E Step
    - “Maximization” step: Update parameters by maximizing the expected log-likelihood (expectation under the estimated distributions of the hidden variables)

      M-Step
    - Repeat

Details: How to learn GMMs?

Score function for GMM

- **Log likelihood** takes the following form (for model M={w,μ,Σ}):

$$
\begin{aligned}
log\, p(D|w,\mu,\Sigma) &= \sum_{i=1}^{N} log\, p(x_n|M) \\
&= \sum_{i=1}^{N} log\left[\sum_{k=1}^{K} p(x_n|k,M)P(k|M)\right] \\
&= \sum_{i=1}^{N} log\left[\sum_{k=1}^{K} w_k N(x_n|\mu_k,\Sigma_k)\right]
\end{aligned}
$$

- Note the sum over components is inside the log
- There is no closed form solution for the MLE

Hidden cluster membership variables

- Consider k cluster indicator variables for example x_n: $\mathbf{z_n} = [z_{n1}, ..., z_{nk}]$ which equals 1 for the cluster that x_n is a member of, and 0 otherwise
- If we knew the values of the hidden cluster membership variables (z) we could easily maximize the complete data log-likelihood, which has a closed form solution:

$$
\begin{aligned}
log\, p(D, \mathbf{z}|w, \mu, \Sigma) &= \sum_{i=1}^{N} log\Big[\sum_{k=1}^{K} z_{nk} \cdot w_k N(x_n|\mu_k, \Sigma_k)\Big] \\
&= \sum_{i=1}^{N} log\, \Big[w_{k'} N(x_n|\mu_{k'}, \Sigma_{k'})\Big] \quad \text{where } z_{nk'} \neq 0 \\
&= \sum_{i=1}^{N} log\, w_{k'} + log\, N(x_n|\mu_{k'}, \Sigma_{k'}) \quad \text{where } z_{nk'} \neq 0
\end{aligned}
$$

- Unfortunately we don't know the values for the hidden variables!
- But, for given set of parameters we can compute the ***expected values*** of the hidden variables (cluster memberships)

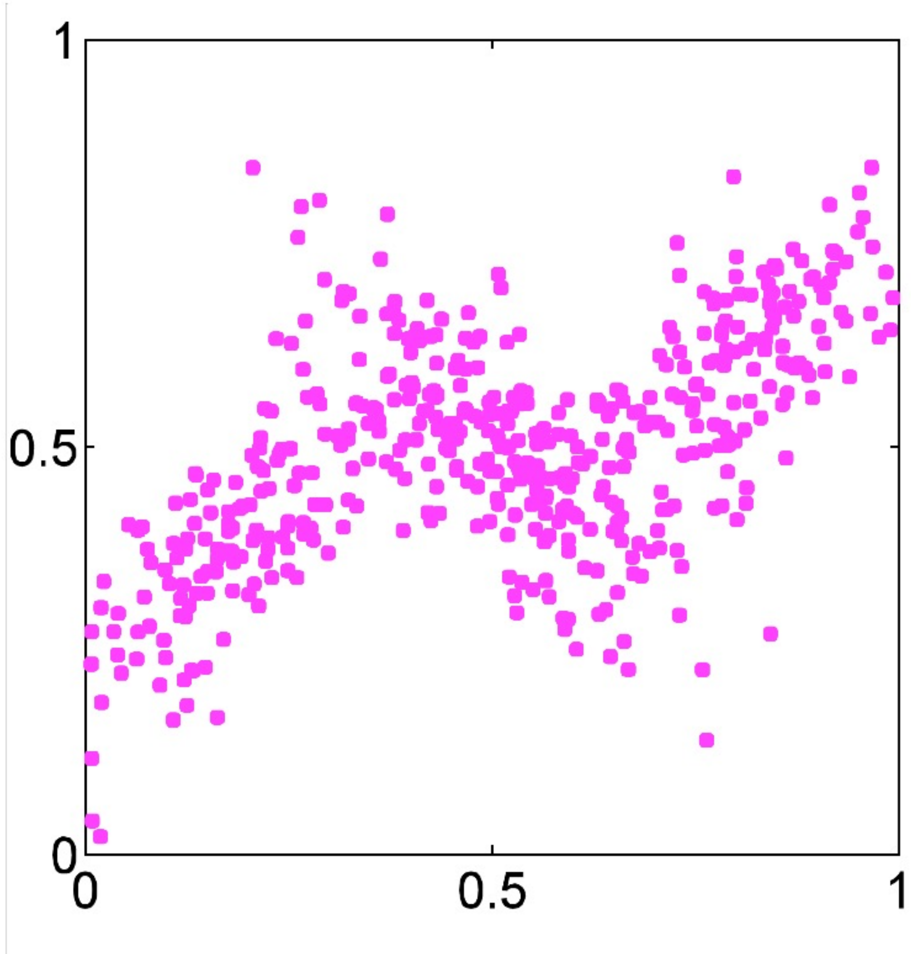
Posterior probabilities of cluster membership

- We can think of the mixing coefficients as **prior** probabilities for cluster membership
- Then for a given example x_n, we can evaluate the corresponding **posterior** probabilities of **cluster membership** with Bayes theorem:

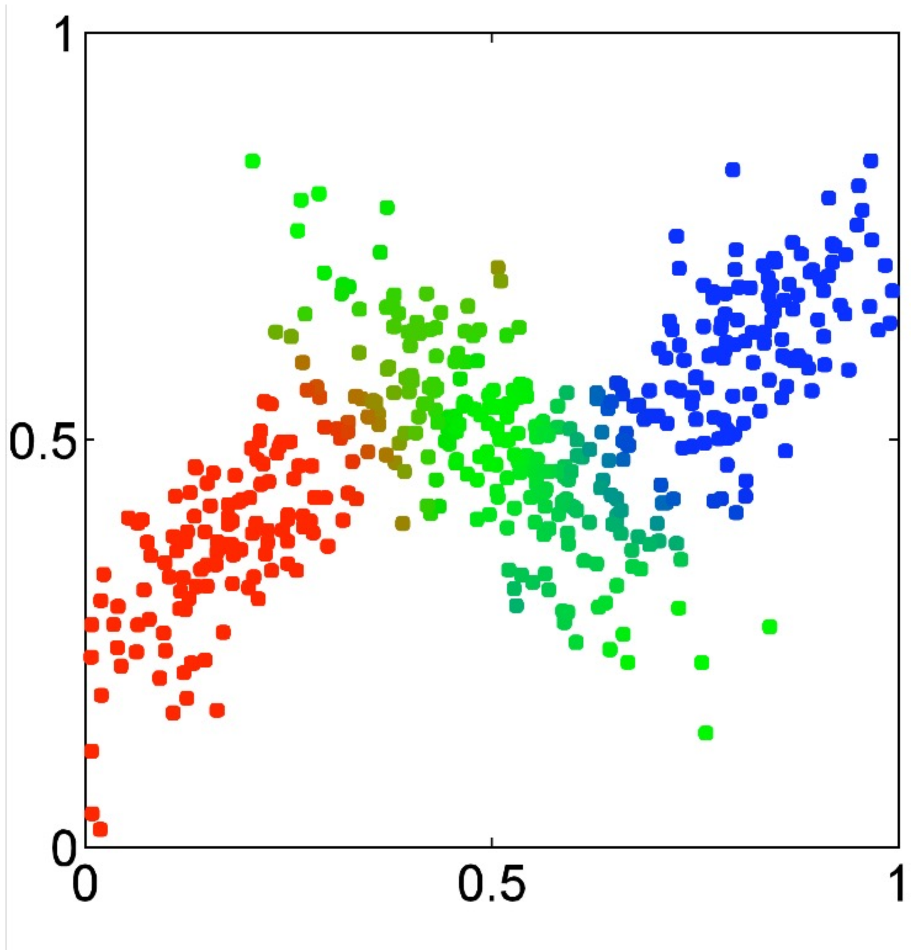
$$
\begin{aligned}
p(z_{nk} = 1 | x_n) &= \frac{p(x_n | z_{nk} = 1) p(z_{nk} = 1)}{p(x_n)} \\
&= \frac{w_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j N(x_n | \mu_j, \Sigma_j)}
\end{aligned}
$$

cluster membership for x

Unlabeled dataset



Posterior probabilities of cluster membership



EM for GMM

- Suppose we have a current estimate of all parameter values μ_k, Σ_k
- Use these to estimate probabilities of cluster memberships. Write $\gamma_i(x_n) = p(z_{ni} = 1)$

E-Step

- Now compute the **expected** log-likelihood using estimated probabilities of cluster memberships.

$$\mathbb{E}_z \, log \; p(x, z|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_i(x_n) \left[log \; w_k + log \; N(x_n|\mu_k, \Sigma_k) \right]$$

- Maximize the expected log-likelihood over all μ_k, Σ_k to update parameters

M-Step

- Repeat

M Step Details

- Now compute the **expected** log-likelihood using estimated probabilities of cluster memberships

$$\mathbb{E}_z\, log\ p(x, z|\theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_i(x_n)\left[log\ w_k + log\ N(x_n|\mu_k, \Sigma_k)\right]$$

- Maximize the expected log-likelihood over all μ_k, Σ_k to update parameters

$$w_k \leftarrow \frac{1}{n}\sum_{i=1}^{n} \gamma_k(x_i)$$

$$\mu_k \leftarrow ?$$

M Step Details

- Now compute the **expected** log-likelihood using estimated probabilities of cluster memberships

$$\mathbb{E}_z\, log\ p(x, z|\theta) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_i(x_n)\left[log\ w_k + log\ N(x_n|\mu_k, \Sigma_k)\right]$$

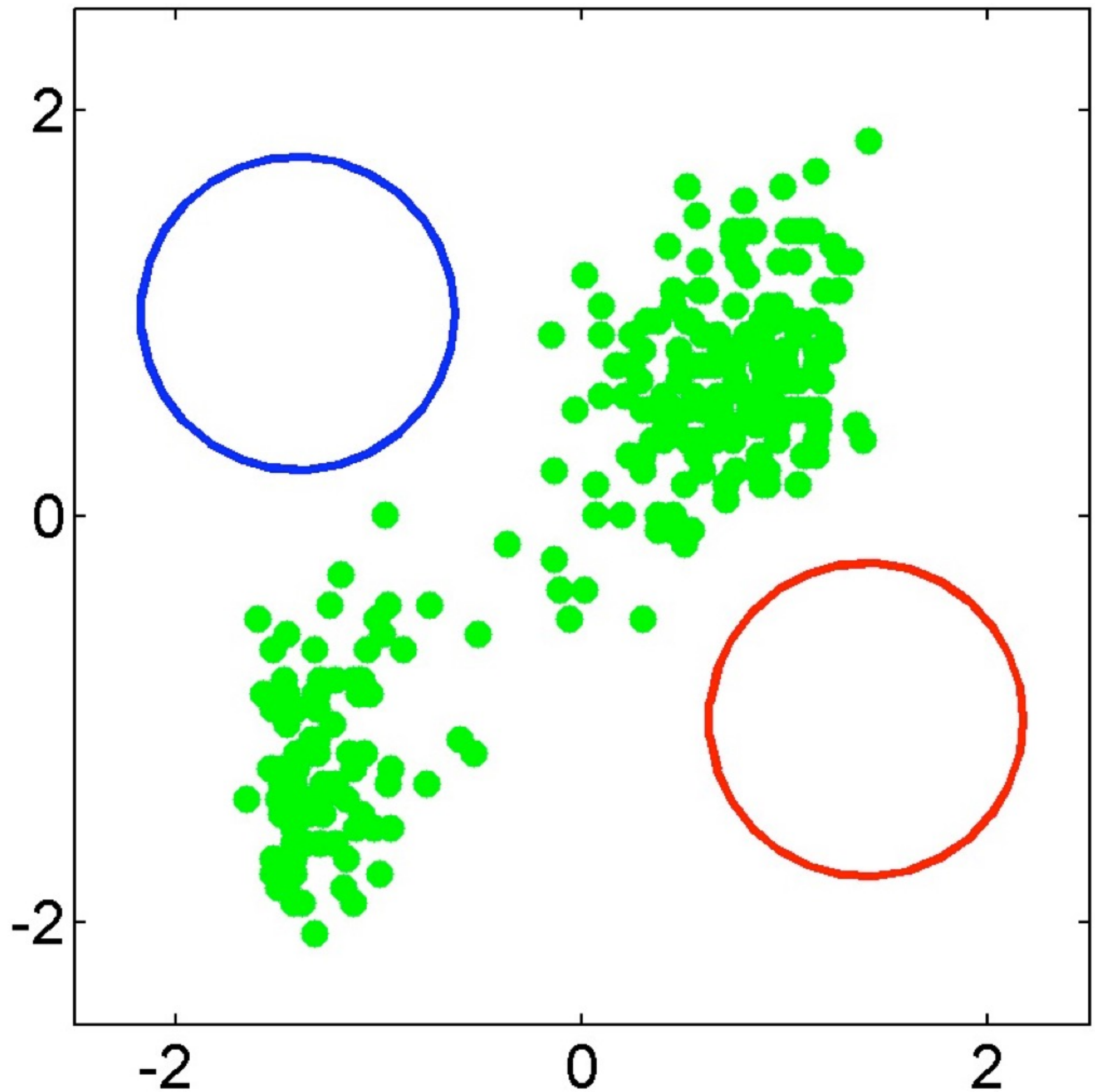
- Maximize the expected log-likelihood over all μ_k, Σ_k to update parameters

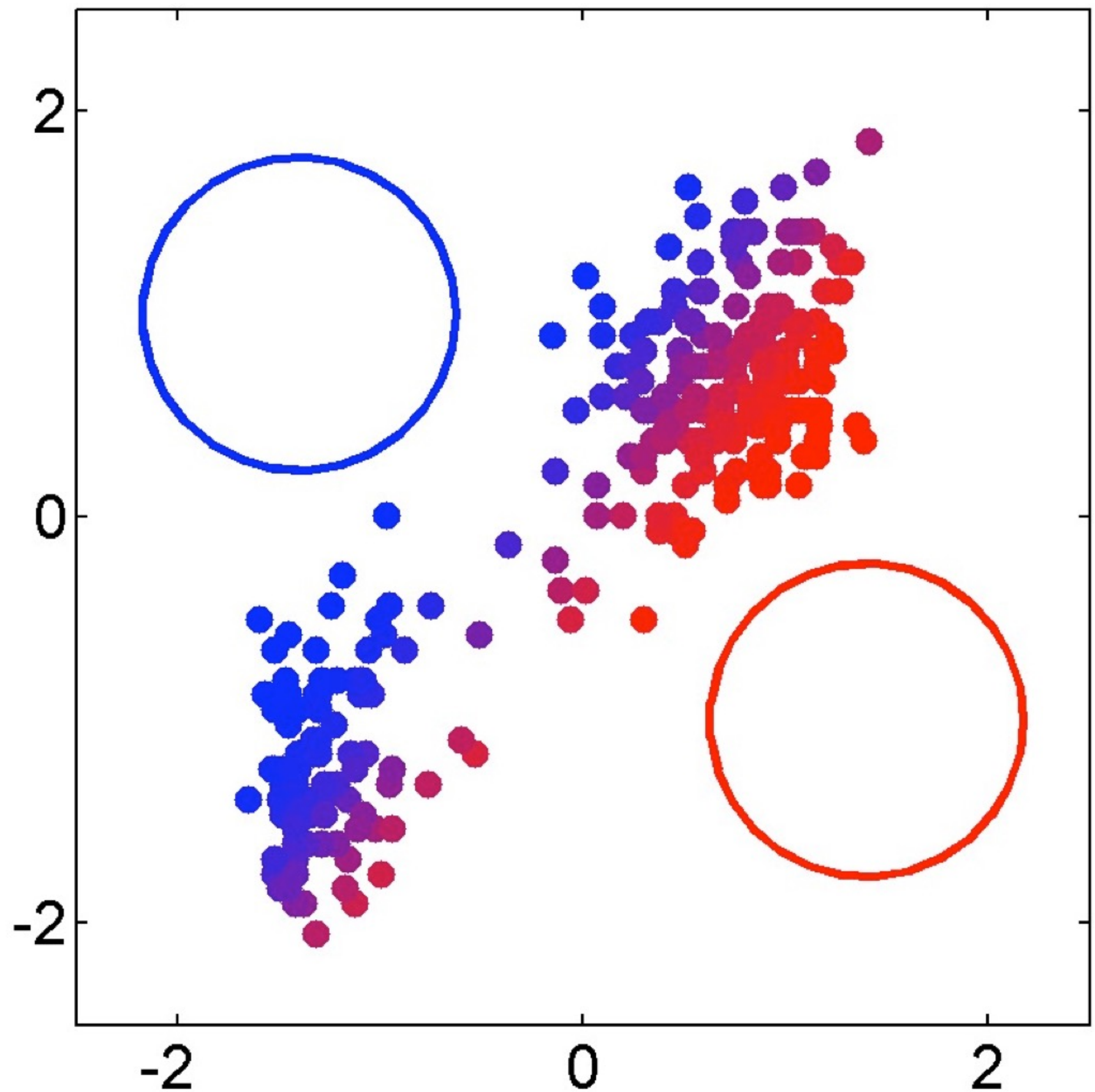
$$w_k \leftarrow \frac{1}{n}\sum_{i=1}^{n}\gamma_k(x_i)$$

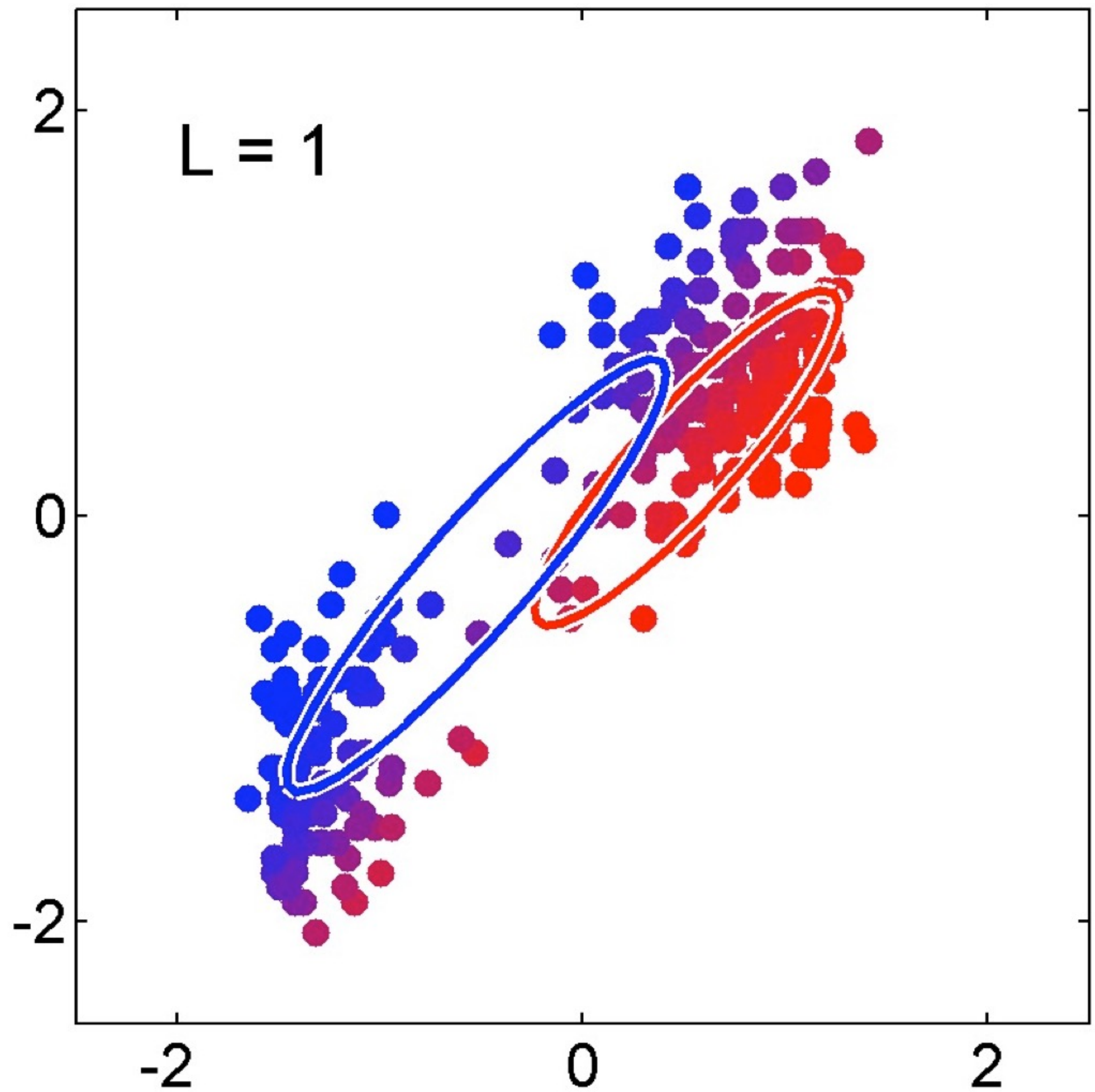
$$\mu_k \leftarrow \frac{\sum_{i=1}^{n}\gamma_k(x_i)x_i}{\sum_{i=1}^{n}\gamma_k(x_i)}$$

$$\Sigma_k \leftarrow \frac{\sum_{i=1}^{n}\gamma_k(x_i)(x_i-\mu_k)(x_i-\mu_k)^\top}{\sum_{i=1}^{n}\gamma_k(x_i)}$$

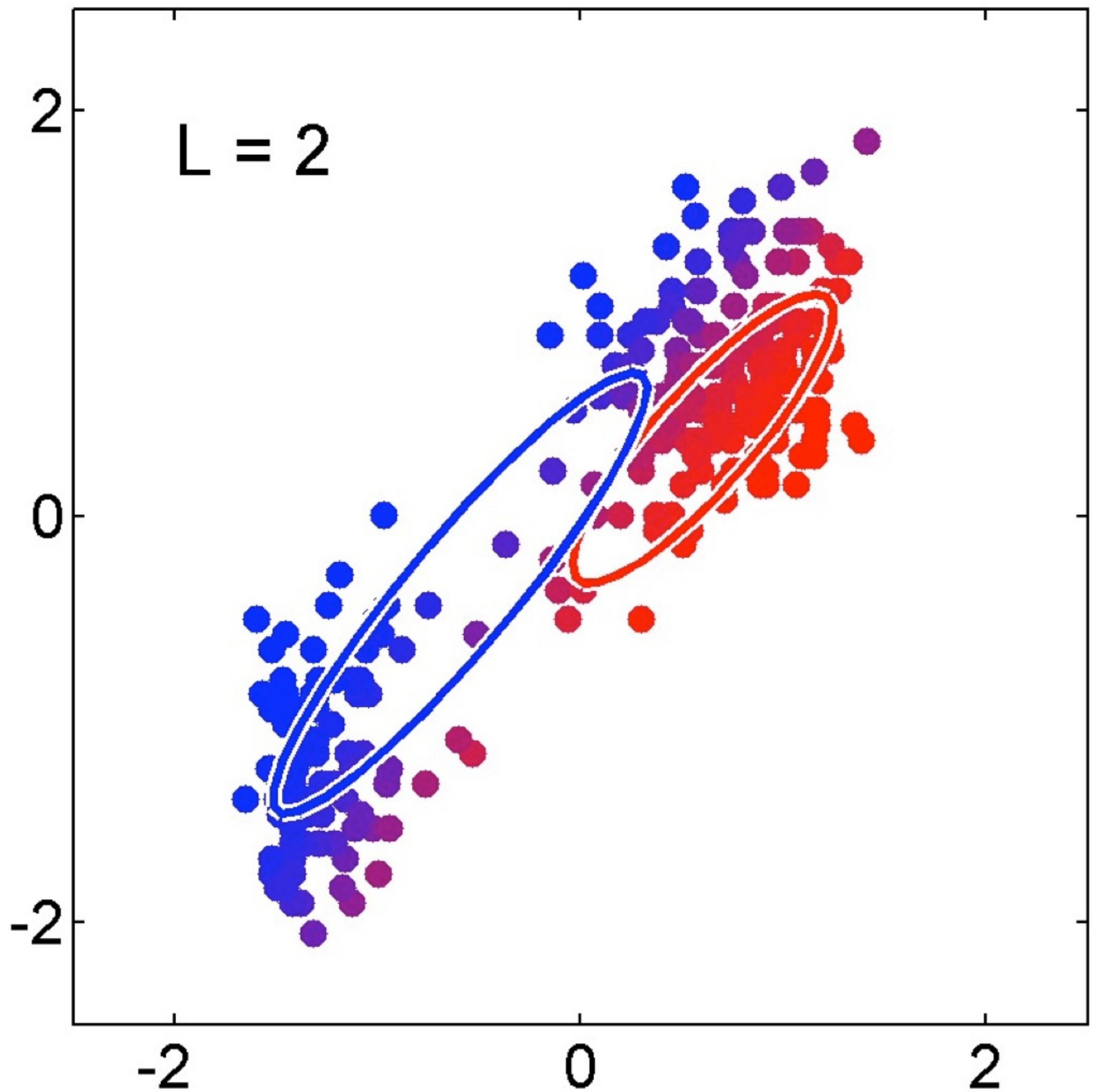
GMM example


2
0
-2
-2
0
2

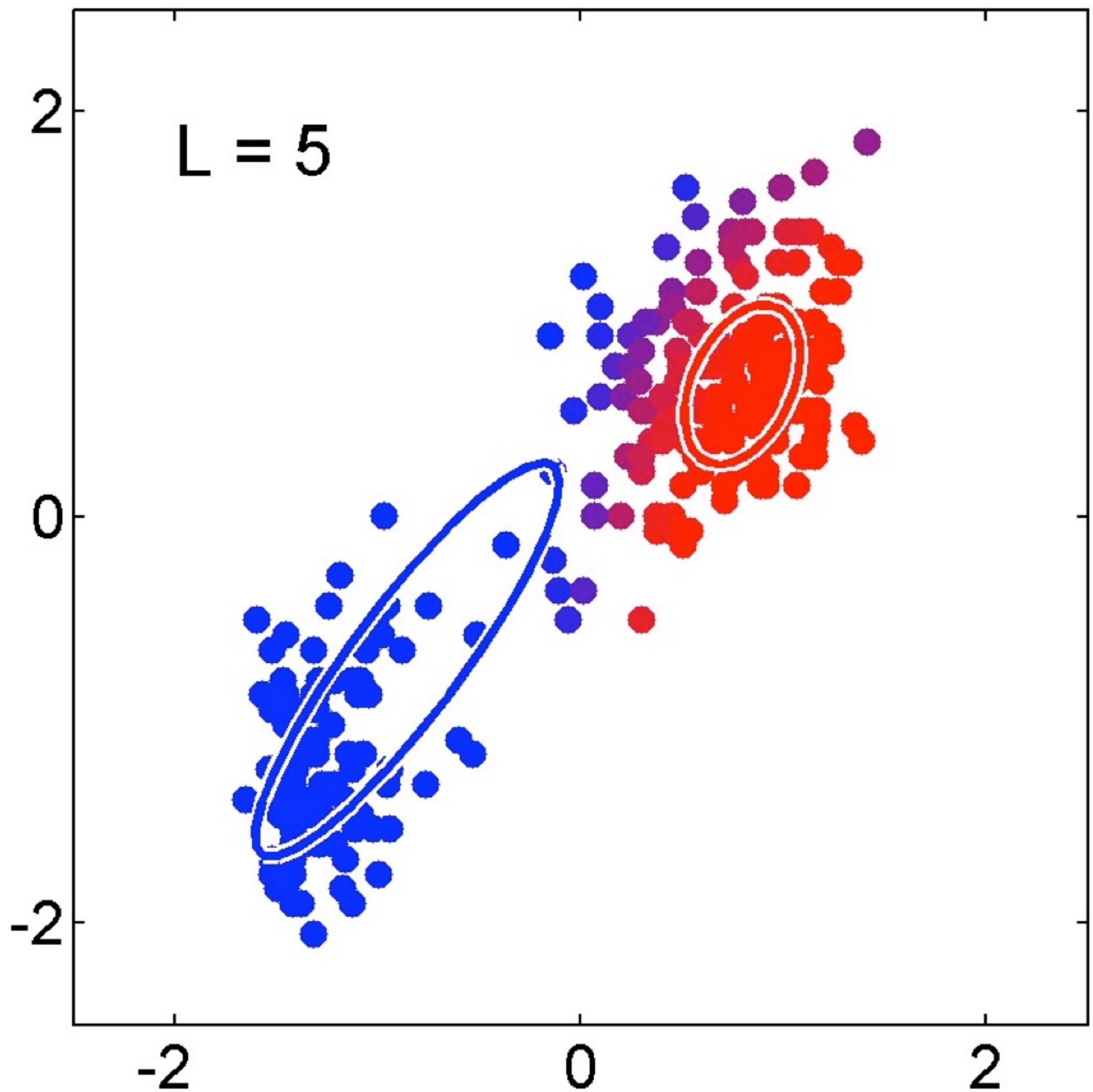

2
0
-2
-2
0
2

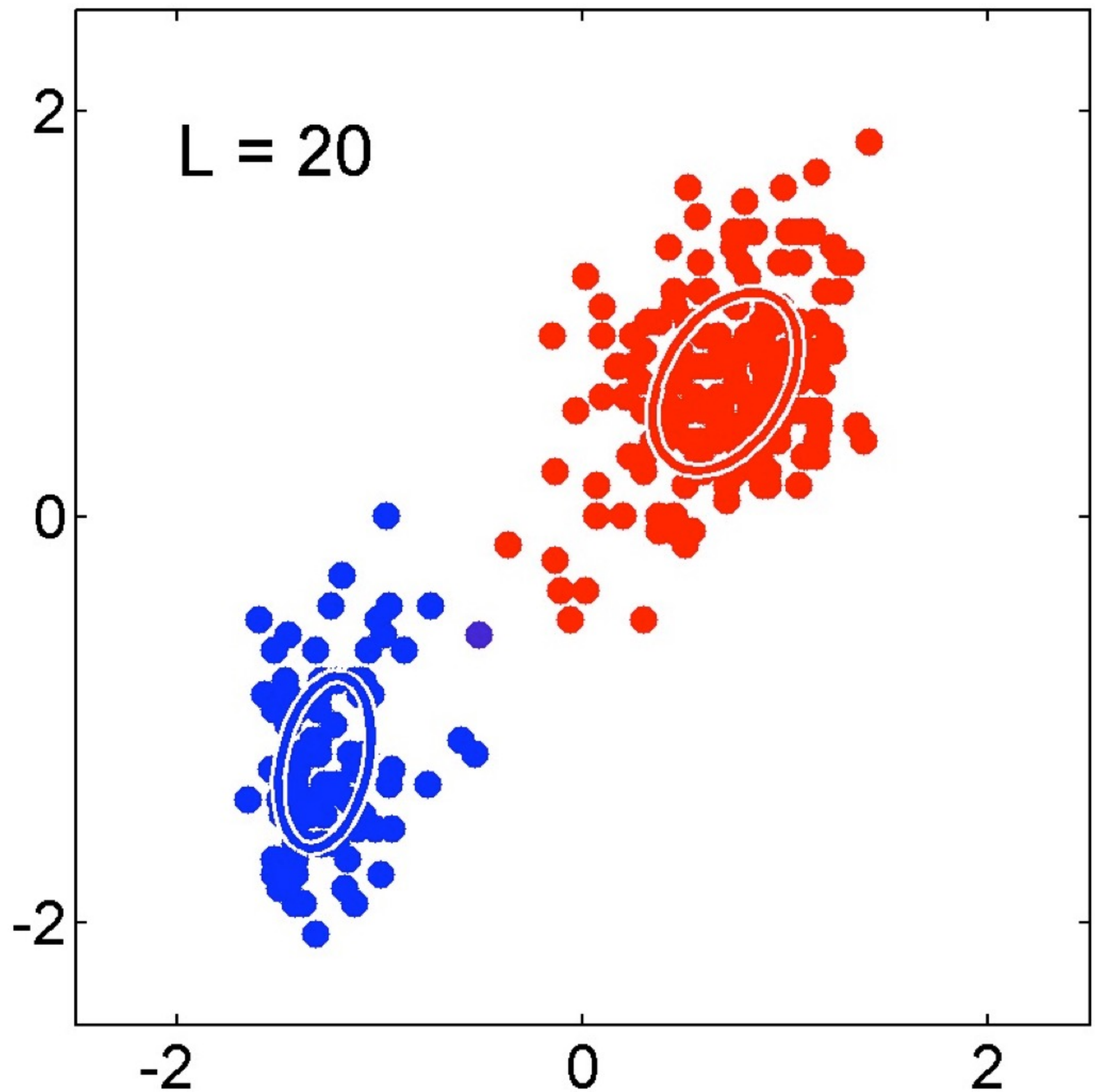


L = 1
2
0
-2
-2
0
2



L = 2
2
0
-2
-2
0
2


L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

More on EM (for general mixture models)

- Often both the E and the M step can be solved in closed form
- Neither the E step nor the M step can decrease the log-likelihood
- Algorithm is guaranteed to converge to a local maximum of the likelihood
- Must specify initialization and stopping criteria

Probabilistic clustering

- Model provides full distributional description for each component
  - May be able to interpret differences in the distributions
- Soft clustering (compared to k-mean hard clustering)
  - Given the model, each point has a k-component vector of membership probabilities
- Key cost: assumption of parametric model

Mixture models

- Knowledge representation?
  - **Parametric model**
    parameters = mixture coefficient and component parameters
- Score function?
  - **Likelihood**
- Search?
  - **Expectation maximization**
    iteratively find parameters that maximize likelihood and predicts cluster memberships
- Optimal?
  - Converges to a local max

Connection to K-Means

- If we restrict to Σ_k = Identity matrix, and w_k = 1/K
- Then only one difference between K-Means and EM for Gaussian Mixtures:
  - EM uses probabilistic cluster assignments
  - K-Means just sets $\gamma_k(x_i)$=1 for the k that EM would give highest $\gamma_k(x_i)$

- So EM is like a “soft” clustering variant of K-Means
- Plus it uses covariances to help cluster
  - E.g., K-Means can’t handle this scenario, but EM gets it right