CSC 411 Lecture 16: Expectation-Maximization for Mixture of Gaussians

Mengye Ren and Matthew MacKay

University of Toronto

A Generative View of Clustering

- Last time: hard and soft k-means algorithm
- This lecture: probabilistic formulation of clustering
- We need a sensible measure of what it means to cluster the data well
  - This makes it possible to judge different methods
  - It may help us decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we adjust the model parameters using maximum likelihood i.e. to maximize the probability that it would produce exactly the data we observed

The Generative Model

- We'll be working with the following generative model for data $\mathbf{x} \in \mathbb{R}^D$
- Assume a datapoint $\mathbf{x}$ is generated as follows:
  - Choose a cluster z from $\{1, \ldots, K\}$ such that $p(z = k) = \pi_k$
  - Given z, sample $\mathbf{x}$ from a Gaussian distribution $\mathcal{N}(\mu_z, I)$
- Can also be written:

$$p(z = k) = \pi_k$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\mu_k, I)$$

Clusters from Generative Model

- This defines a joint distribution $p(z, \mathbf{x}) = p(z)p(\mathbf{x}|z)$ with parameters $\{\pi_k, \mu_k\}_{k=1}^K$
- $p(z = k|\mathbf{x})$ can be computed using Bayes rule and tells us the probability $\mathbf{x}$ came from the k^{th} cluster
- How should we choose the parameters $\{\pi_k, \mu_k\}_{k=1}^K$?

Maximum Likelihood with Latent Variables

- Maximum likelihood principle: choose parameters to maximize likelihood of **observed data**
- We don't observe the cluster assignments z- we only see the data $\mathbf{x}$
- Given data $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, choose parameters to maximize:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$$

- We can find $p(\mathbf{x})$ by marginalizing out z:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z = k, \mathbf{x}) = \sum_{k=1}^{K} p(z = k) p(\mathbf{x} | z = k)$$

Gaussian Mixture Model (GMM)

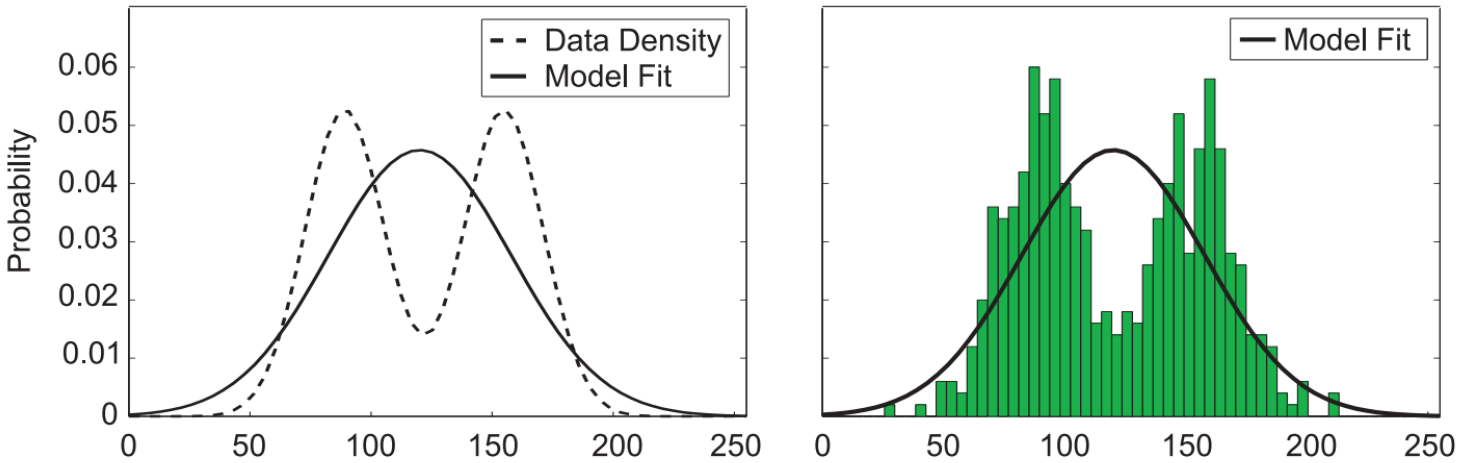
What is $p(\mathbf{x})$?

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z = k)p(\mathbf{x}|z = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, I)$$

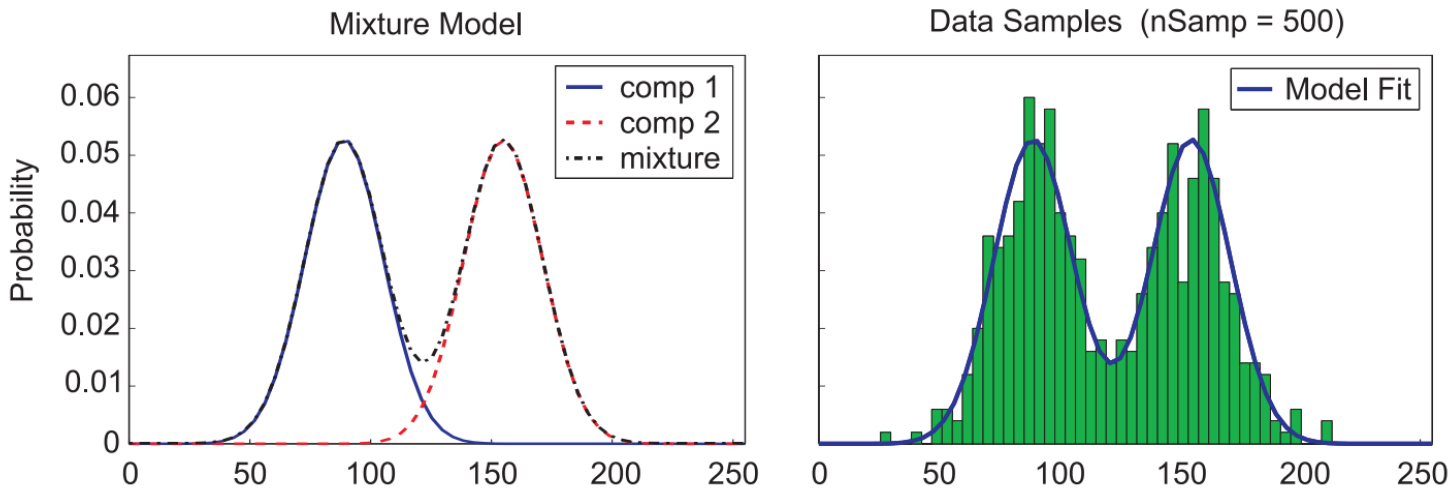
- This distribution is an example of a **Gaussian Mixture Model (GMM)**, and π_k are known as the **mixing coefficients**
- If we allow arbitrary covariance matrices, GMMs are **universal approximators of densities** (if you have enough Gaussians). Even diagonal GMMs are universal approximators.

Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

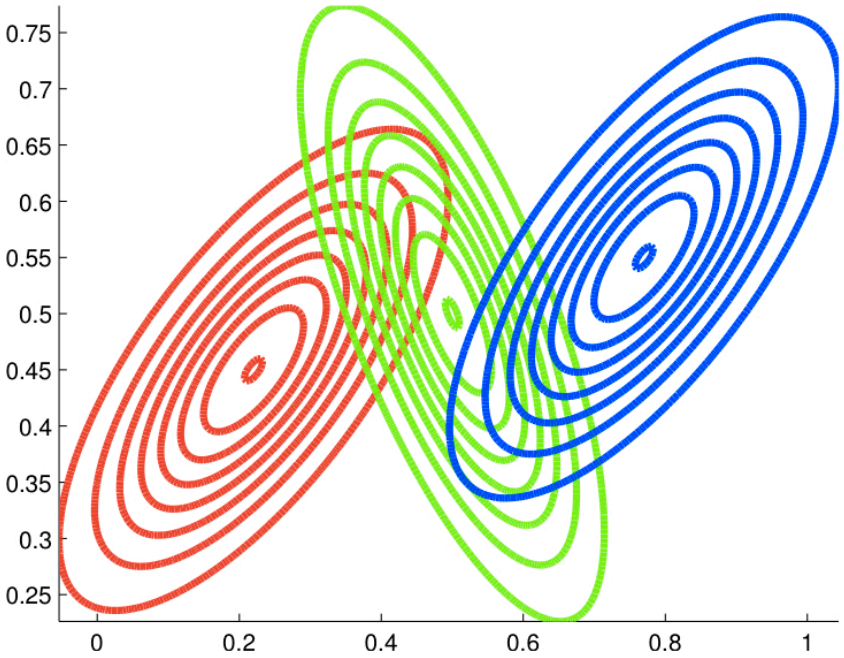


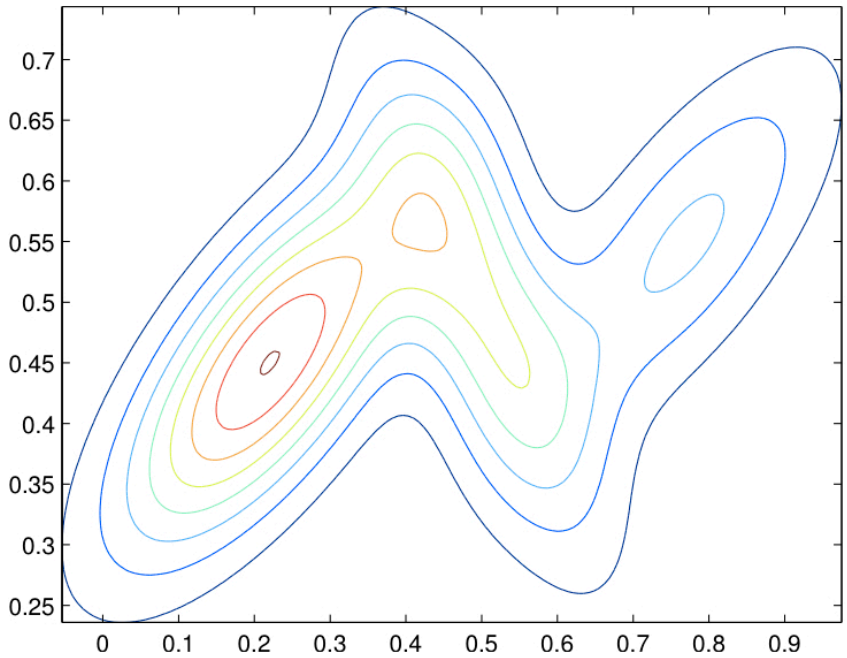
- Now, we are trying to fit a GMM (with $K = 2$ in this example):

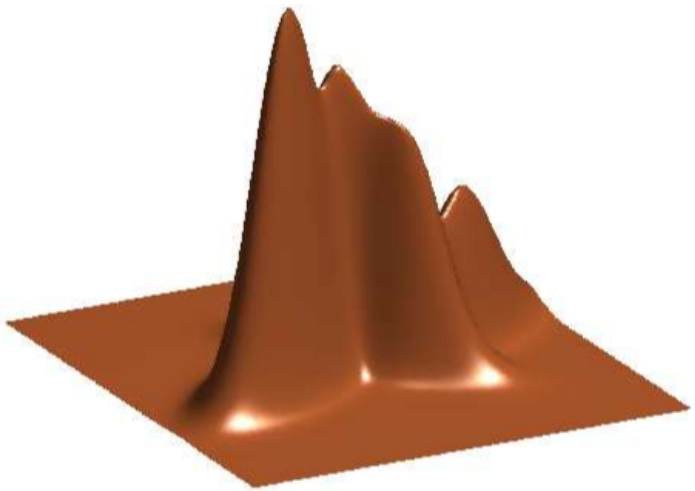


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs: Maximum Likelihood

Maximum likelihood objective:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, I) \right)$$

- How would you optimize this w.r.t. parameters $\{\pi_k, \mu_k\}$?
  - No closed form solution when we set derivatives to 0
  - Difficult because sum inside the log
- One option: gradient ascent. Can we do better?
- Can we have a closed form update?

Maximum Likelihood

- **Observation: if we knew** $z^{(n)}$ for every $\mathbf{x}^{(n)}$, (i.e. our dataset was $\mathcal{D}_{\text{complete}} = \{(z^{(n)}, \mathbf{x}^{(n)})\}_{n=1}^{N}$) the maximum likelihood problem is easy:

$$\begin{aligned}
\log p(\mathcal{D}_{\text{complete}}) &= \sum_{n=1}^{N} \log p(z^{(n)}, \mathbf{x}^{(n)}) \\
&= \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)} | z^{(n)}) + \log p(z^{(n)}) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, I) + \log \pi_k)
\end{aligned}$$

Maximum Likelihood

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(x^{(n)}|\mu_k, I) + \log \pi_k)$$

- We have been optimizing something similar for Gaussian bayes classifiers
- We would get this:

$$\begin{aligned} \mu_k &= \frac{\sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k]\, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k]} \\ \pi_k &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k] \end{aligned}$$

Maximum Likelihood

- We don't know $z^{(n)}$ for every $\mathbf{x}^{(n)}$, but we can compute $p(z^{(n)}|\mathbf{x}^{(n)})$ using Bayes rule
- Conditional probability (using Bayes rule) of $\mathbf{z}$ given $\mathbf{x}$

$$\begin{aligned} p(z=k|\mathbf{x}) &= \frac{p(z=k)p(\mathbf{x}|z=k)}{p(\mathbf{x})} \\ &= \frac{p(z=k)p(\mathbf{x}|z=k)}{\sum_{j=1}^{K} p(z=j)p(\mathbf{x}|z=j)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, I)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, I)} \end{aligned}$$

Maximum Likelihood

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(x^{(n)}|\mu_k, I) + \log \pi_k)$$

- If we plug in $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ for $\mathbb{I}[z^{(n)} = k]$, we get:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)}(\log \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) + \log \pi_k)$$

- This is still easy to optimize! Solution is similar to what we have seen:

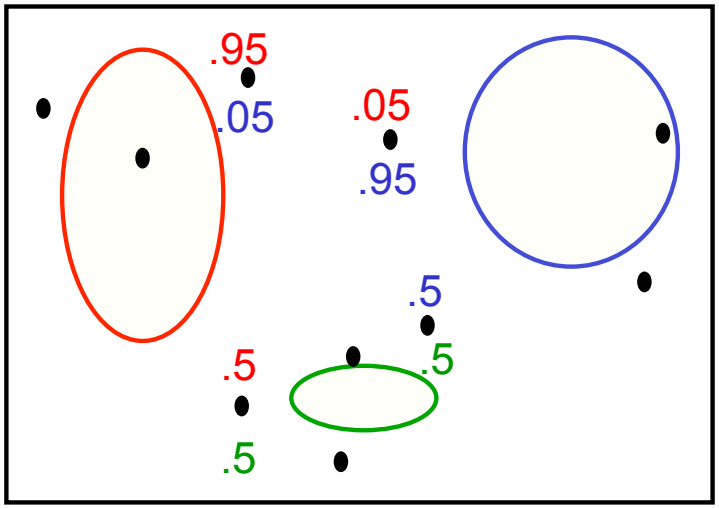
$$\mu_k = \frac{\sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} r_k^{(n)}}$$

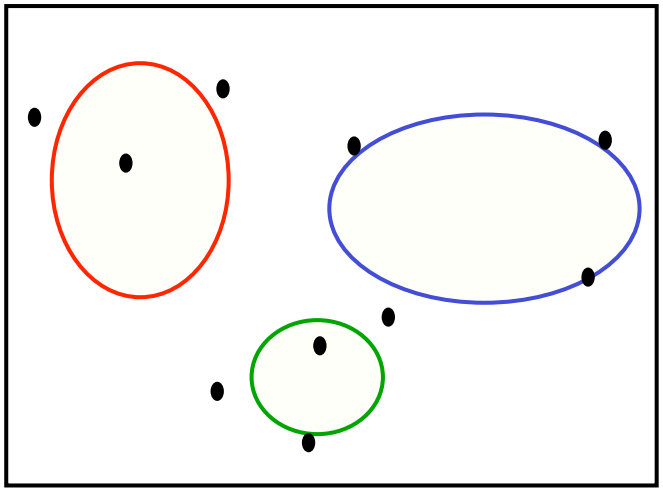
$$\pi_k = \frac{\sum_{n=1}^{N} r_k^{(n)}}{N}$$

- Note: this only works if we treat $r_k^{(n)}$ as fixed – really, it depends on the model parameters as well

How Can We Fit a Mixture of Gaussians?

- This motivates the **Expectation-Maximization algorithm**, which alternates between two steps:
  1. **E-step**: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k|x^{(n)})$ given our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.
  2. **M-step**: Use the equations on the last slide to update the parameters, assuming $r_k^{(n)}$ are held fixed- change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.





EM Algorithm for GMM

- **Initialize** the means μ_k and mixing coefficients π_k
- Iterate until convergence:
  - **E-step**: Evaluate the responsibilities $r_k^{(n)}$ given current parameters

$$r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, I)}$$

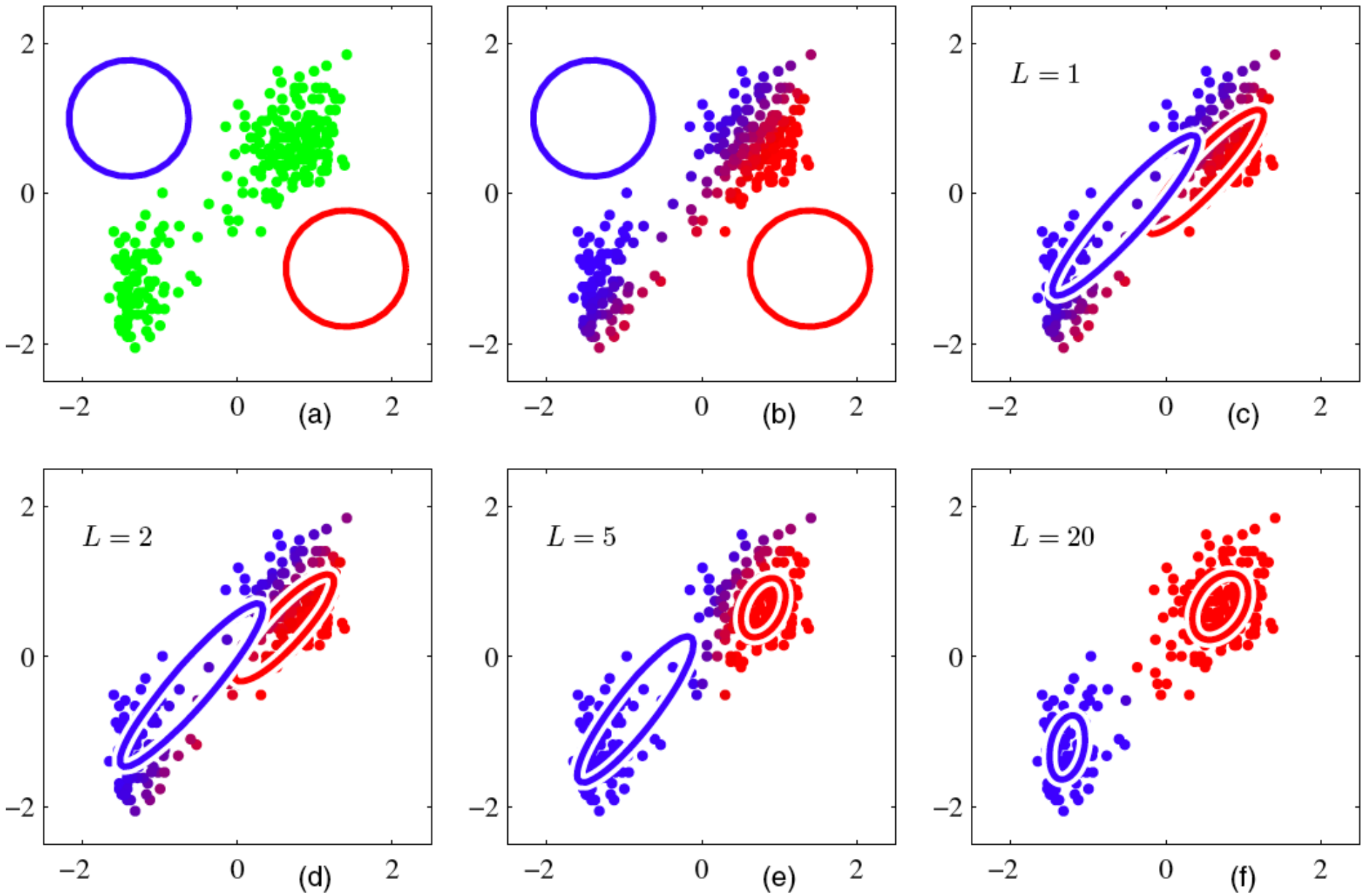
  - **M-step**: Re-estimate the parameters given current responsibilities

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}$$

$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} r_k^{(n)}$$

  - Evaluate log likelihood and check for convergence

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) \right)$$


2
0
−2
−2
0
2
(a)
(b)
$L = 1$
(c)
$L = 2$
(d)
$L = 5$
(e)
$L = 20$
(f)

What just happened: A review

- The maximum likelihood objective $\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$ was hard to optimize
- The complete data likelihood objective was easy to optimize:

$$\sum_{n=1}^{N} \log p(z^{(n)}, \mathbf{x}^{(n)}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) + \log \pi_k)$$

- We don't know $z^{(n)}$ for each $\mathbf{x}^{(n)}$, so we replaced $\mathbb{I}[z^{(n)} = k]$ with $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$
- Another way of saying this: we replaced $\mathbb{I}[z^{(n)} = k]$ with its **expectation** under $p(z^{(n)}|x^{(n)})$

What just happened: A review

- We ended up with the expected complete data log-likelihood:

$$\sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)}|\mathbf{x}^{(n)})}[\log p(z^{(n)}, \mathbf{x}^{(n)})] = \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} (\log \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) + \log \pi_k)$$

- The EM algorithm alternates between:
  - The E-step: computing the $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ (i.e. the **expectations** $\mathbb{E}_{p(z^{(n)}|\mathbf{x}^{(n)})}[\mathbb{I}[z^{(n)} = k]]$) given the current model parameters π_k, μ_k
  - The M-step: update the model parameters π_k, μ_k to optimize the expected complete data log-likelihood

What just happened: A review

- Why does this make sense? In the next lecture, we'll see how EM is optimizing the observed data log-likelihood $\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$ in a somewhat roundabout fashion
- We'll give a principled justification of the EM algorithm and describe how it can be applied to general latent variable models

Relation to k-Means

- The K-Means Algorithm:
  1. **Assignment step**: Assign each data point to the closest cluster
  2. **Refitting step**: Move each cluster center to the average of the data assigned to it
- The EM Algorithm:
  1. **E-step**: Compute the posterior probability over z given our current model
  2. **M-step**: Maximize the probability that it would generate the data it is currently responsible for.

Further Discussion

- We assumed the covariance of each Gaussian was I to simplify the math. This assumption can be removed, allowing clusters to have different spatial extents
- Possible problems with maximum likelihood objective:
  - **Singularities**: Arbitrarily large likelihood when a Gaussian explains a single point with variance shrinking to zero
  - Non-convex

GMM Recap

- A probabilistic view of clustering - Each cluster corresponds to a different Gaussian.
- Model using **latent variables**.
- General approach, can replace Gaussian with other distributions (continuous or discrete)
- More generally, mixture model are very powerful models, **universal approximator**
- Optimization is done using the **EM** algorithm.