

# CSC 411: Introduction to Machine Learning

## Lecture 16: GMM

Mengye Ren and Matthew MacKay

University of Toronto

# A Generative View of Clustering

- Last time: hard and soft k-means algorithm
- This lecture: statistical formulation of clustering → principled, justification for updates
- 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 to maximize the probability that it would produce exactly the data we observed

# Generative Models Recap

- We model the joint distribution as,

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

- But in unsupervised clustering we do not have the class labels  $z$ .
- What can we do instead?

$$p(\mathbf{x}) = \sum_z p(\mathbf{x}, z) = \sum_z p(\mathbf{x}|z)p(z)$$

- This is a **mixture model**

# Gaussian Mixture Model (GMM)

Most common mixture model: **Gaussian mixture model** (GMM)

- A GMM represents a **distribution** as

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

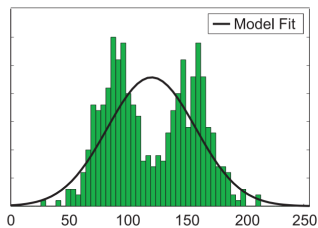
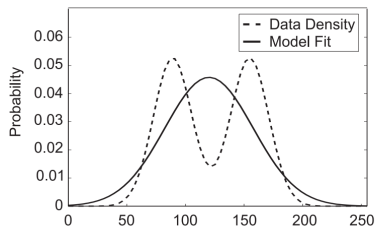
with  $\pi_k$  the **mixing coefficients**, where:

$$\sum_{k=1}^K \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

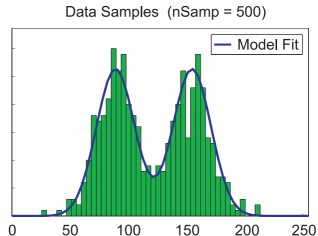
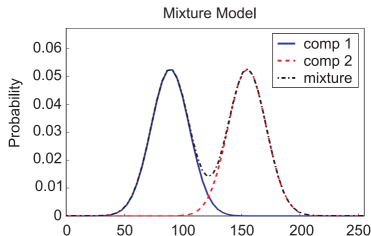
- GMM is a density estimator
- GMMs are **universal approximators of densities** (if you have enough Gaussians). Even diagonal GMMs are universal approximators.
- In general mixture models are very powerful, but harder to optimize

# Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

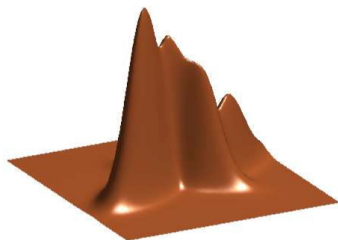
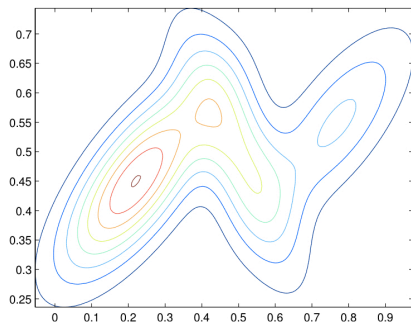
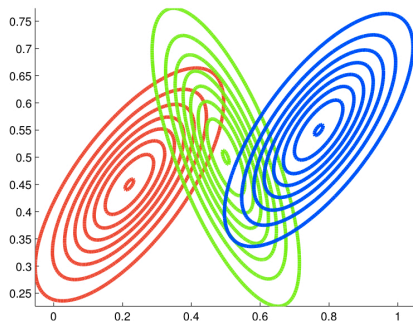


- Now, we are trying to fit a GMM (with  $K = 2$  in this example):



[Slide credit: K. Kutulakos]

# Visualizing a Mixture of Gaussians – 2D Gaussians



# Fitting GMMs: Maximum Likelihood

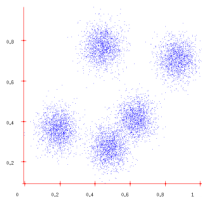
- Maximum likelihood maximizes

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left( \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$

w.r.t  $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$

- Problems:
  - ▶ **Singularities:** Arbitrarily large likelihood when a Gaussian explains a single point
  - ▶ **Identifiability:** Solution is invariant to permutations
  - ▶ Non-convex
- How would you optimize this?
- Can we have a closed form update?
- Don't forget to satisfy the constraints on  $\pi_k$  and  $\Sigma_k$

# Latent Variable



- Our original representation had a hidden (latent) variable  $z$  which would represent which Gaussian generated our observation  $\mathbf{x}$ , with some probability
- Let  $z \sim \text{Categorical}(\boldsymbol{\pi})$  (where  $\pi_k \geq 0$ ,  $\sum_k \pi_k = 1$ )
- Then:

$$p(\mathbf{x}) = \sum_{k=1}^K p(\mathbf{x}, z = k) = \sum_{k=1}^K \underbrace{p(z = k)}_{\pi_k} \underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- This breaks a complicated distribution into simple components - the price is the hidden variable.



# Latent Variable Models

- Some model variables may be unobserved, either at training or at test time, or both
- If occasionally unobserved they are missing, e.g., undefined inputs, missing class labels, erroneous targets
- Variables which are always unobserved are called **latent variables**, or sometimes **hidden variables**
  
- We may want to intentionally introduce latent variables to model complex dependencies between variables – this can actually simplify the model
- Form of divide-and-conquer: use simple parts to build complex models
  
- In a **mixture model**, the identity of the component that generated a given datapoint is a latent variable

# Back to GMM

- A Gaussian mixture distribution:

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

- We had:  $z \sim \text{Categorical}(\boldsymbol{\pi})$  (where  $\pi_k \geq 0$ ,  $\sum_k \pi_k = 1$ )
- Joint distribution:  $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$
- Log-likelihood:

$$\begin{aligned} \ell(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln p(\mathbf{x}^{(n)} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \sum_{n=1}^N \ln \sum_{z^{(n)}=1}^K p(\mathbf{x}^{(n)} | z^{(n)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(n)} | \boldsymbol{\pi}) \end{aligned}$$

- Note: We have a hidden variable  $z^{(n)}$  for every observation
- General problem: sum inside the log
- How can we optimize this?

# Maximum Likelihood

- If we knew  $z^{(n)}$  for every  $x^{(n)}$ , the maximum likelihood problem is easy:

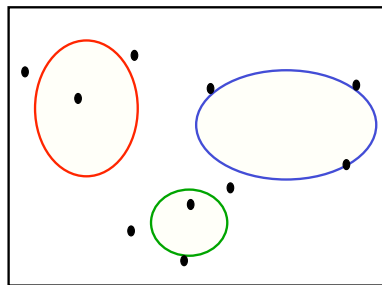
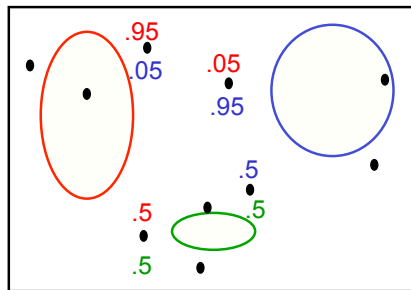
$$\ell(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln p(x^{(n)}, z^{(n)} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln p(\mathbf{x}^{(n)} | z^{(n)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \ln p(z^{(n)} | \boldsymbol{\pi})$$

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

$$\begin{aligned}\mu_k &= \frac{\sum_{n=1}^N \mathbf{1}_{[z^{(n)}=k]} \mathbf{x}^{(n)}}{\sum_{n=1}^N \mathbf{1}_{[z^{(n)}=k]}} \\ \Sigma_k &= \frac{\sum_{n=1}^N \mathbf{1}_{[z^{(n)}=k]} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T}{\sum_{n=1}^N \mathbf{1}_{[z^{(n)}=k]}} \\ \pi_k &= \frac{1}{N} \sum_{n=1}^N \mathbf{1}_{[z^{(n)}=k]}\end{aligned}$$

# Intuitively, How Can We Fit a Mixture of Gaussians?

- Optimization uses the **Expectation Maximization algorithm**, which alternates between two steps:
  1. **E-step**: Compute the posterior probability over  $z$  given our current model - i.e. how much do we think each Gaussian generates each datapoint.
  2. **M-step**: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.



- The K-Means Algorithm:
  1. **Assignment step:** Assign each data point to the closest cluster
  2. **Refitting step:** Move each cluster center to the center of gravity 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.

- Elegant and powerful method for finding maximum likelihood solutions for models with latent variables

## 1. E-step:

- ▶ In order to adjust the parameters, we must first solve the inference problem: Which Gaussian generated each datapoint?
- ▶ We cannot be sure, so it's a distribution over all possibilities.

$$\gamma_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)}; \pi, \mu, \Sigma)$$

## 2. M-step:

- ▶ Each Gaussian gets a certain amount of posterior probability for each datapoint.
- ▶ We fit each Gaussian to the weighted datapoints
- ▶ We can derive closed form updates for all parameters

# GMM E-Step: Responsibilities

Lets see how it works on GMM:

- Conditional probability (using Bayes rule) of  $z$  given  $\mathbf{x}$

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

- $\gamma_k$  can be viewed as the **responsibility** of cluster  $k$  towards  $\mathbf{x}$

- Once we computed  $\gamma_k^{(i)} = p(z^{(i)} = k | \mathbf{x}^{(i)})$  we can compute the expected likelihood

$$\begin{aligned} & \mathbb{E}_{P(z^{(i)} | \mathbf{x}^{(i)})} \left[ \sum_i \log(P(\mathbf{x}^{(i)}, z^{(i)} | \Theta)) \right] \\ &= \sum_i \sum_k \gamma_k^{(i)} \left( \log(P(z^i = k | \Theta)) + \log(P(\mathbf{x}^{(i)} | z^{(i)} = k, \Theta)) \right) \\ &= \sum_i \sum_k \gamma_k^{(i)} \left( \log(\pi_k) + \log(\mathcal{N}(\mathbf{x}^{(i)}; \mu_k, \Sigma_k)) \right) \\ &= \sum_k \sum_i \gamma_k^{(i)} \log(\pi_k) + \sum_k \sum_i \gamma_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \mu_k, \Sigma_k)) \end{aligned}$$

- We need to fit  $k$  Gaussians, just need to weight examples by  $\gamma_k$



- Need to optimize

$$\sum_k \sum_i \gamma_k^{(i)} \log(\pi_k) + \sum_k \sum_i \gamma_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \mu_k, \Sigma_k))$$

- Solving for  $\mu_k$  and  $\Sigma_k$  is like fitting  $k$  separate Gaussians but with weights  $\gamma_k^{(i)}$ .
- Solution is similar to what we have already seen:

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

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

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

# EM Algorithm for GMM

- **Initialize** the means  $\mu_k$ , covariances  $\Sigma_k$  and mixing coefficients  $\pi_k$
- Iterate until convergence:
  - ▶ **E-step**: Evaluate the responsibilities given current parameters

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

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

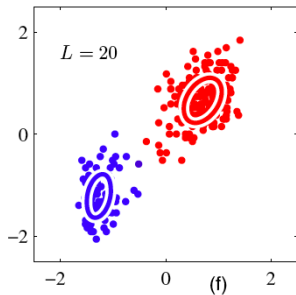
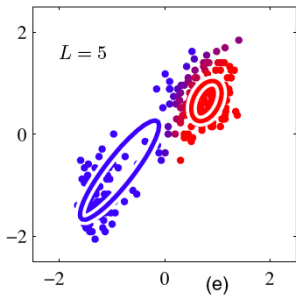
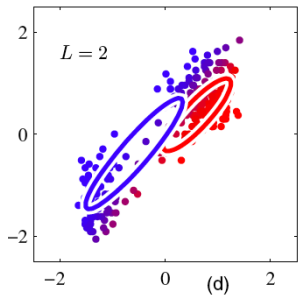
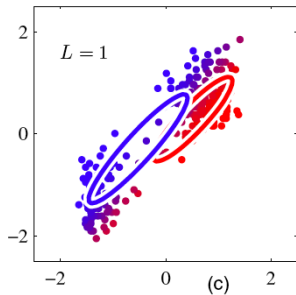
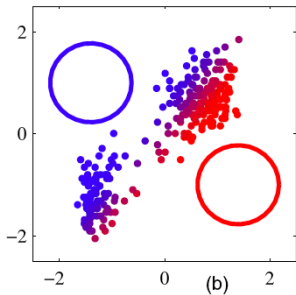
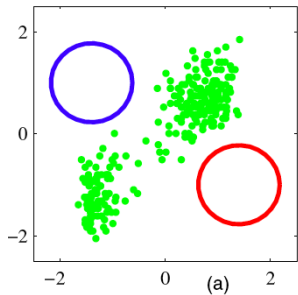
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} \mathbf{x}^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

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

- ▶ Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left( \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$



# Mixture of Gaussians vs. K-means

- EM for mixtures of Gaussians is just like a soft version of K-means, with **fixed priors and covariance**
- Instead of hard assignments in the E-step, we do **soft assignments** based on the softmax of the squared Mahalanobis distance from each point to each cluster.
- Each center moved by **weighted means** of the data, with weights given by soft assignments
- In K-means, weights are 0 or 1

- 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.

# Where does EM come from?

- In the next lecture, we'll see a principled justification of the EM algorithm and describe how it can be applied to general latent variable models