



**Xi'an Jiaotong-Liverpool University**

**西交利物浦大学**

# **INT305 Machine Learning**

## **Lecture 10**

### **k-Means and EM Algorithm**

**Siyue Yu**

**Department Intelligence Science**

**[siyue.yu02@xjtu.edu.cn](mailto:siyue.yu02@xjtu.edu.cn)**

# Today

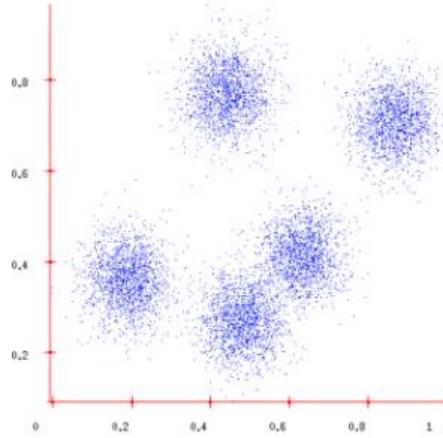
---

- Today's lecture:
  - First, introduce K-means, a simple algorithm for **clustering**, i.e. grouping data points into clusters
  - Then, we will reformulate clustering as a latent variable model, apply the EM algorithm

# Clustering

---

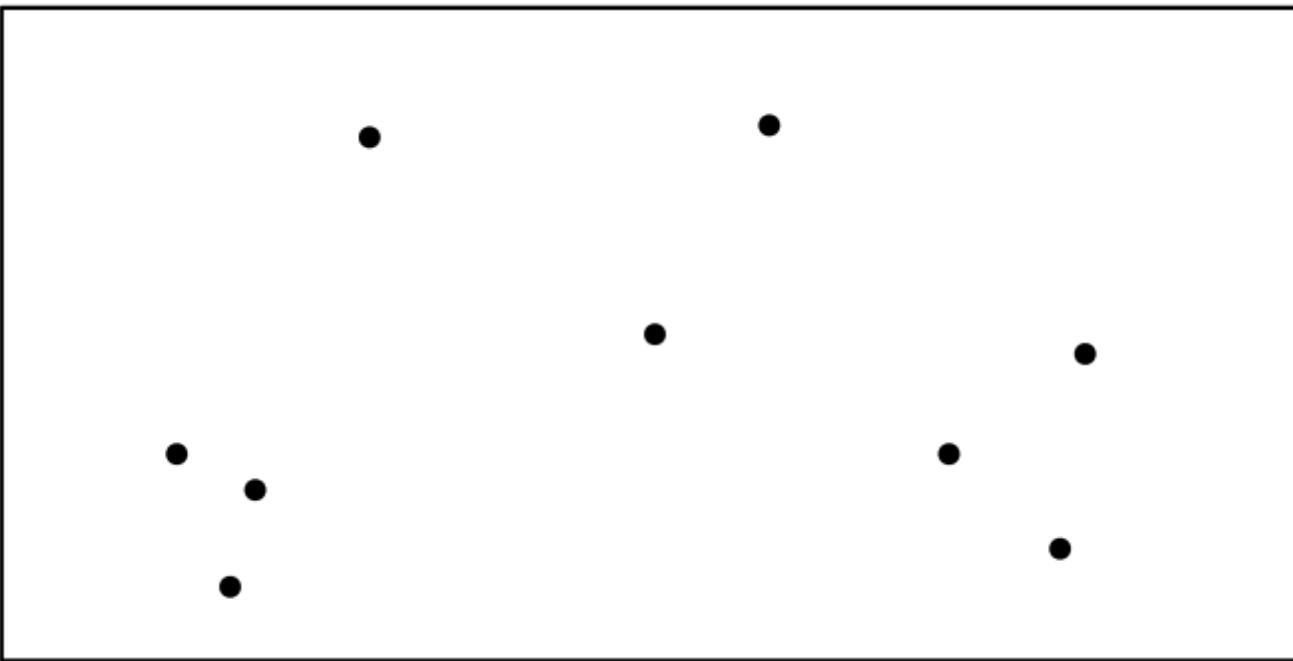
- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:



- Grouping data points into clusters, **with no observed labels**, is called **clustering**. It is an unsupervised learning technique.
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)
  - But topics are never observed (unsupervised).

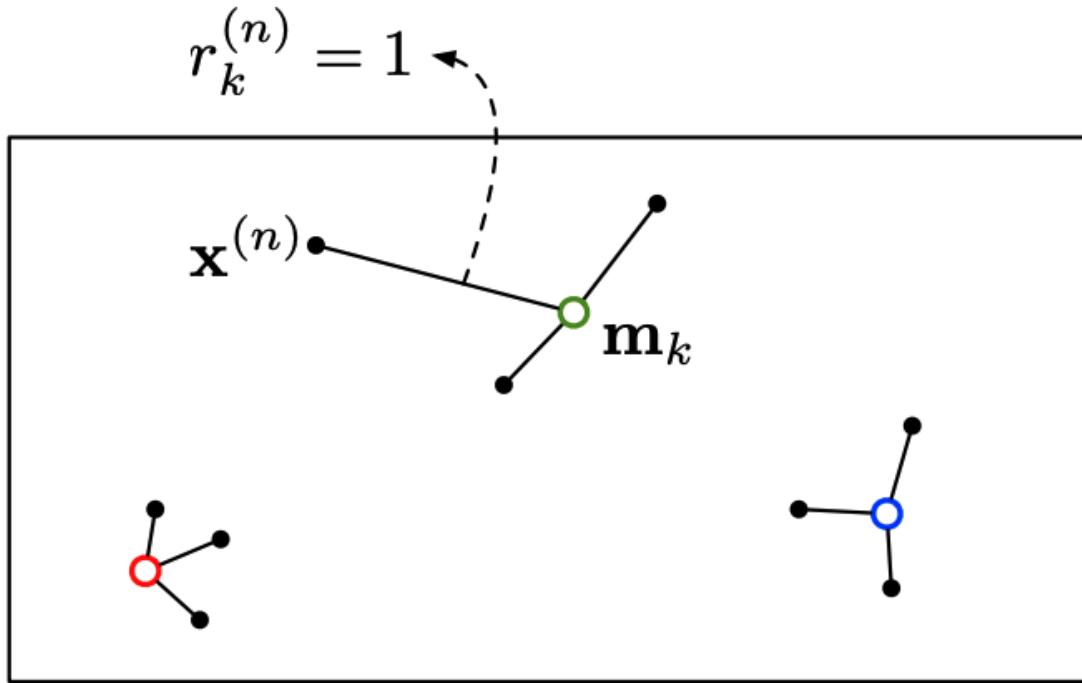
# Clustering problem

---



- Assume the data  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$  lives in a Euclidean space,  $\mathbf{x}^{(n)} \in \mathbb{R}^D$ .
- Assume each data point belongs to one of K clusters.
- Assume the data points from same cluster are similar, i.e. close in Euclidean distance.
- How can we identify those clusters (data points that belong to each cluster)? Let's formulate as an optimization problem.

# K-means Objective



**K-means Objective:** Find cluster centers  $\{\mathbf{m}_k\}_{k=1}^K$  and assignments  $\{\mathbf{r}^{(n)}\}_{n=1}^N$  to minimize the sum of squared distances of data points  $\{\mathbf{x}^{(n)}\}$  to their assigned centers.

- Data sample  $n = 1, \dots, N$ :  $\mathbf{x}^{(n)} \in \mathbb{R}^D$  (observed),
- Cluster center  $k = 1, \dots, K$ :  $\mathbf{m}_k \in \mathbb{R}^D$  (not observed),
- Responsibilities: Cluster assignment for sample  $n$ :  $\mathbf{r}^{(n)} \in \mathbb{R}^K$  1-of-K encoding (not observed)

# K-means Objective

---

- **K-means Objective:** Find cluster centers  $\{\mathbf{m}_k\}_{k=1}^K$  and assignments  $\{\mathbf{r}^{(n)}\}_{n=1}^N$  to minimize the sum of squared distances of data points  $\{\mathbf{x}^{(n)}\}$  to their assigned centers.
  - Data sample  $n = 1, \dots, N$ :  $\mathbf{x}^{(n)} \in \mathbb{R}^D$  (observed),
  - Cluster center  $k = 1, \dots, K$ :  $\mathbf{m}_k \in \mathbb{R}^D$  (not observed),
  - Responsibilities: Cluster assignment for sample  $n$ :  $\mathbf{r}^{(n)} \in \mathbb{R}^K$  1-of-K encoding (not observed)
- Mathematically:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} J(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}) = \min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

where  $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)} \text{ is assigned to cluster } k]$ , i.e.,  $\mathbf{r}^{(n)} = [0, \dots, 1, \dots, 0]^\top$

- Finding an optimal solution is an NP-hard problem!

# K-means Objective

---

- Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \underbrace{\sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2}_{\text{distance between } x^{(n)} \text{ and its assigned cluster center}}$$

- Since  $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)} \text{ is assigned to cluster } k]$ , i.e.,  $\mathbf{r}^{(n)} = [0, \dots, 1, \dots, 0]^T$
- inner sum is over  $K$  terms but only one of them is non-zero.
- E.g. say sample  $\mathbf{x}^{(n)}$  is assigned to cluster  $k = 3$ , then

$$\mathbf{r}^n = [0, 0, 1, 0, \dots]$$

$$\sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2 = \|\mathbf{m}_3 - \mathbf{x}^{(n)}\|^2$$

# How to optimize? Alternating Minimization

---

Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

- Problem is hard when minimizing jointly over the parameters  $\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}$
- But note that if we fix one and minimize over the other, then it becomes easy.

# How to optimize? Alternating Minimization

---

Optimization problem:

$$\min_{\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

- Note:

- If we fix the centers  $\{\mathbf{m}_k\}$  then we can easily find the optimal assignments  $\{\mathbf{r}^{(n)}\}$  for each sample  $n$

$$\min_{\mathbf{r}^{(n)}} \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

- Assign each point to the cluster with the nearest center

$$r_k^{(n)} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}^{(n)} - \mathbf{m}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

- E.g. if  $\mathbf{x}^{(n)}$  is assigned to cluster  $\hat{k}$

$$\mathbf{r}^{(n)} = \underbrace{[0, 0, \dots, 1, \dots, 0]^\top}_{\text{Only } \hat{k}\text{-th entry is 1}}$$

# Alternating Minimization

---

- Likewise, if we fix the assignments  $\{\mathbf{r}^{(n)}\}$  then can easily find optimal centers  $\{\mathbf{m}_k\}$
- Set each cluster's center to the average of its assigned data points: For  $l = 1, 2, \dots, K$

$$\begin{aligned} 0 &= \frac{\partial}{\partial \mathbf{m}_l} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2 \\ &= 2 \sum_{n=1}^N r_l^{(n)} (\mathbf{m}_l - \mathbf{x}^{(n)}) \quad \Rightarrow \quad \mathbf{m}_l = \frac{\sum_n r_l^{(n)} \mathbf{x}^{(n)}}{\sum_n r_l^{(n)}} \end{aligned}$$

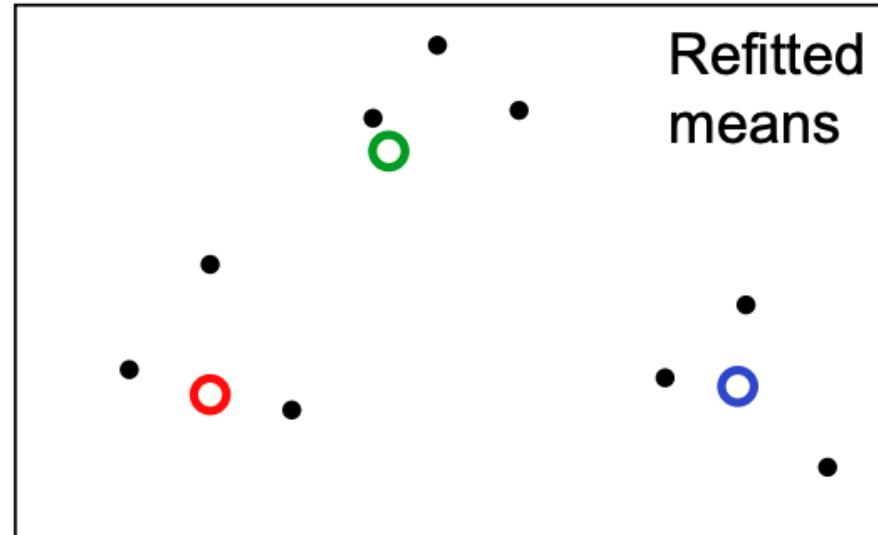
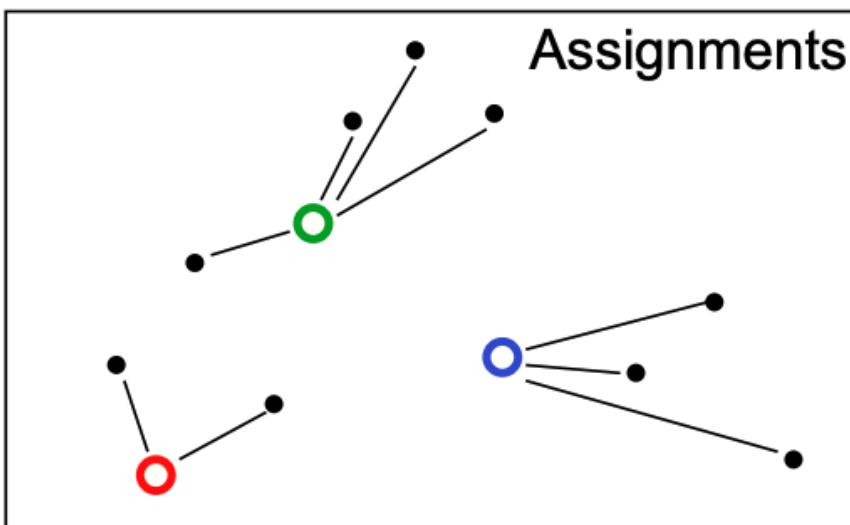
- Let's alternate between minimizing  $J(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\})$  with respect to  $\{\mathbf{m}_k\}$  and  $\{\mathbf{r}^{(n)}\}$
- This is called **alternating minimization**

# K-means Algorithm

---

High level overview of algorithm:

- **Initialization:** randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - **Assignment step:** Assign each data point to the closest cluster
  - **Refitting step:** Move each cluster center to the mean of the data assigned to it



# K-means Algorithm

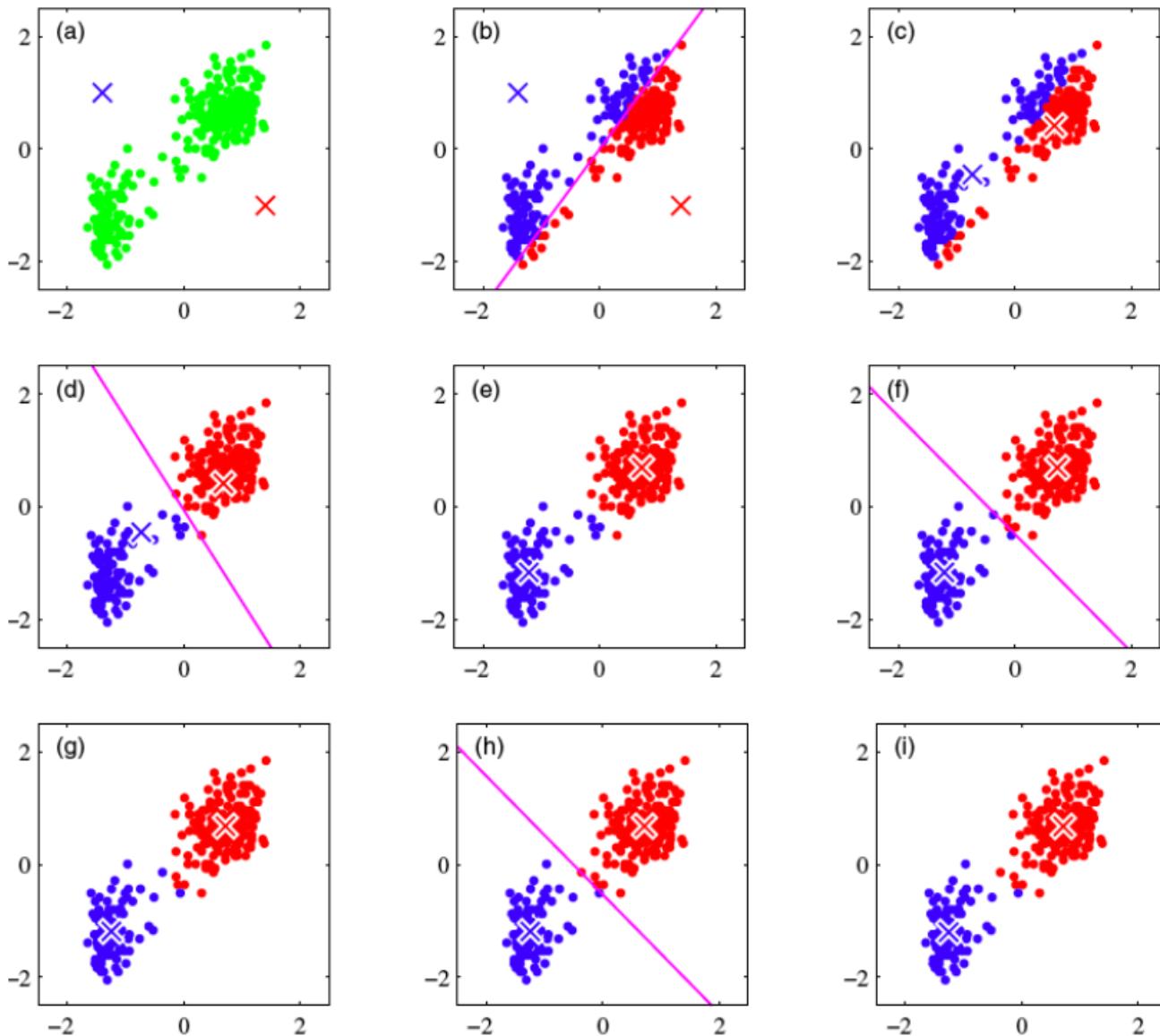


Figure from Bishop

Simple demo: <http://syskall.com/kmeans.js/>

# The K-means Algorithm

---

- **Initialization:** Set K cluster means  $\mathbf{m}_1, \dots, \mathbf{m}_K$  to random values
- Repeat until convergence (until assignments do not change):
  - Assignment: Optimize  $J$  w.r.t.  $\{\mathbf{r}\}$ : Each data point  $\mathbf{x}^{(n)}$  assigned to nearest center

$$\hat{k}^{(n)} = \arg \min_k \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2$$

and **Responsibilities** (1-hot or 1-of-K encoding)

$$r_k^{(n)} = \mathbb{I}[\hat{k}^{(n)} = k] \text{ for } k = 1, \dots, K$$

- **Refitting:** Optimize  $J$  w.r.t.  $\{\mathbf{m}\}$ : Each center is set to mean of data assigned to it

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}.$$

# K-means for Vector Quantization

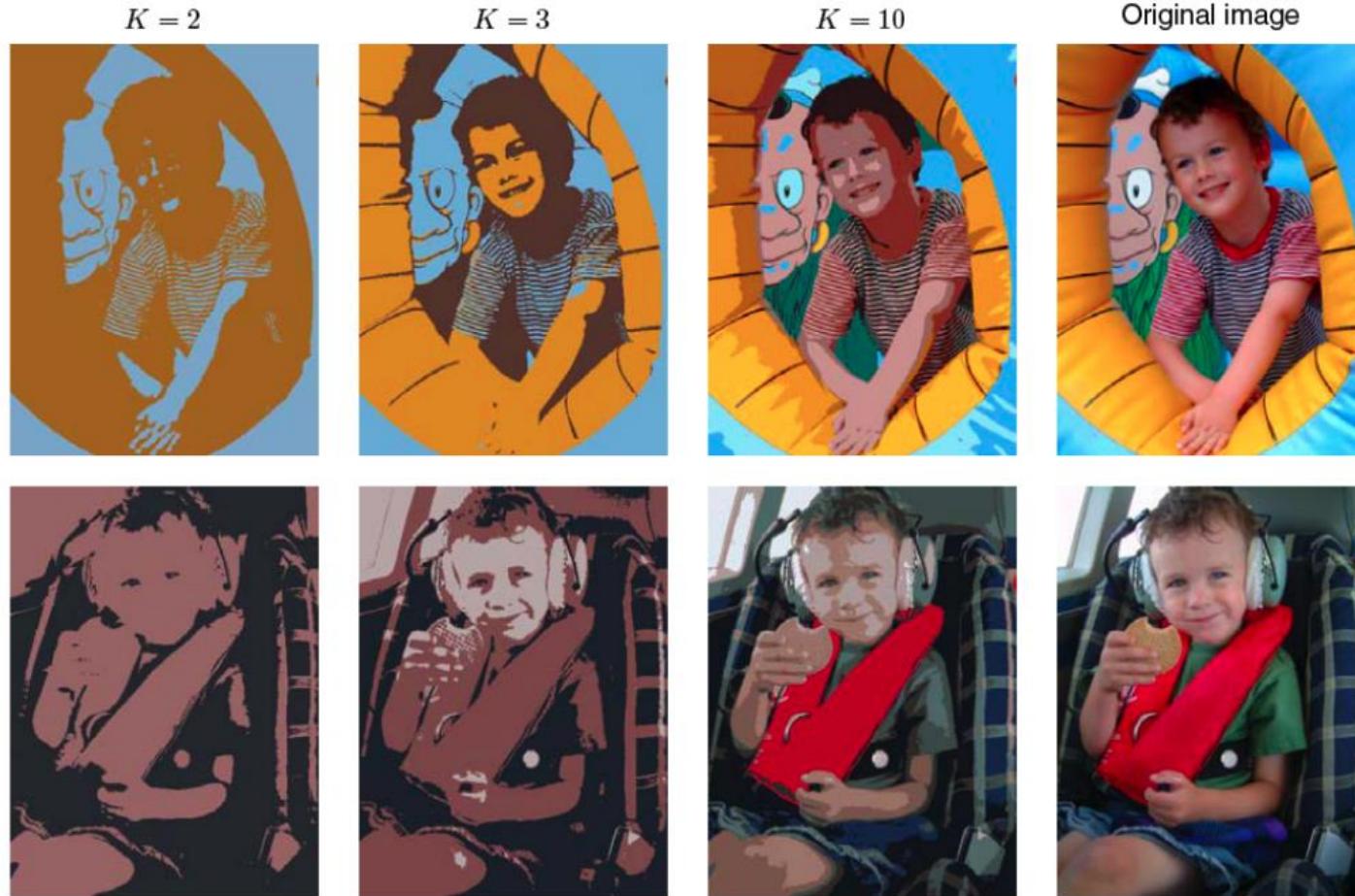
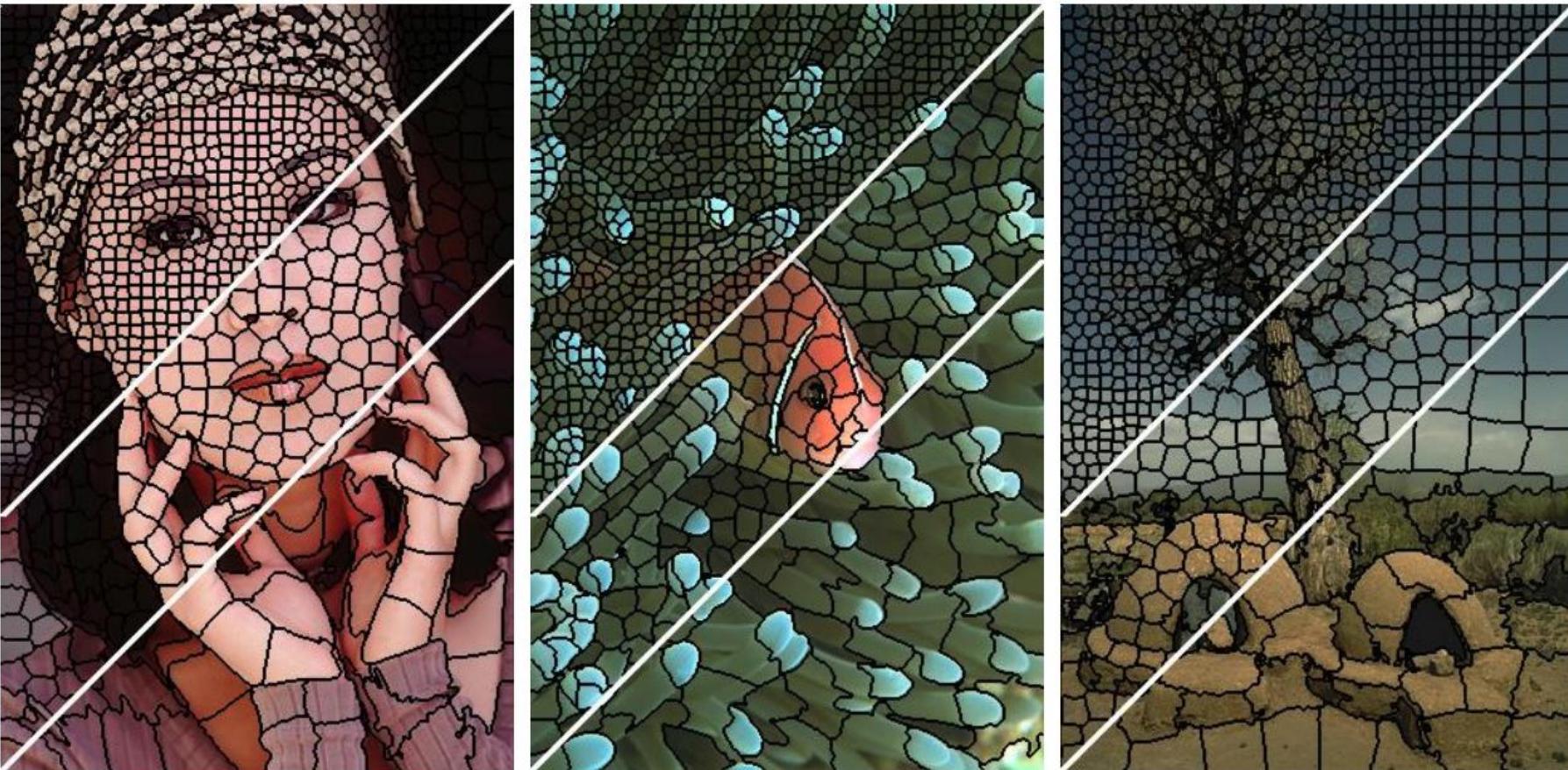


Figure from Bishop

- Given image, construct “dataset” of pixels represented by their RGB pixel intensities
- Run k-means, replace each pixel by its cluster center

# K-means for Image Segmentation

---



- Given image, construct “dataset” of pixels, represented by their RGB pixel intensities and grid locations
- Run k-means (with some modifications) to get superpixels

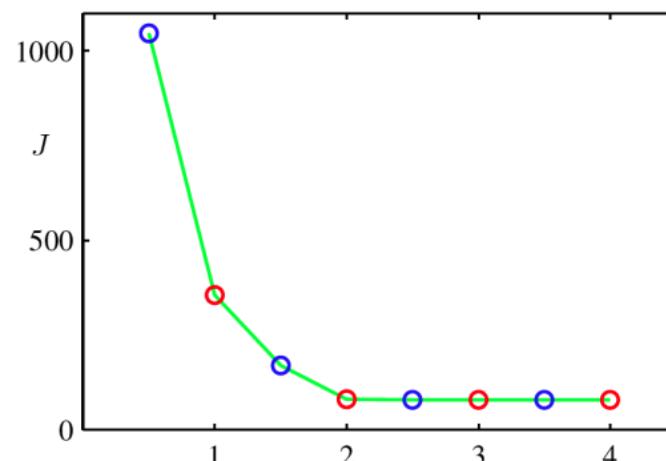
# Questions about K-means

---

- Why does update set  $\mathbf{m}_k$  to mean of assigned points?
- What if we used a different distance measure?
- How can we choose the best distance?
- How to choose  $K$ ?
- Will it converge?

# Why K-means Converges

- K-means algorithm reduces the cost at each iteration.
  - Whenever an assignment is changed, the sum squared distances  $J$  of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved,  $J$  is reduced.
- **Test for convergence**: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).
- This will always happen after a finite number of iterations, since the number of possible cluster assignments is finite



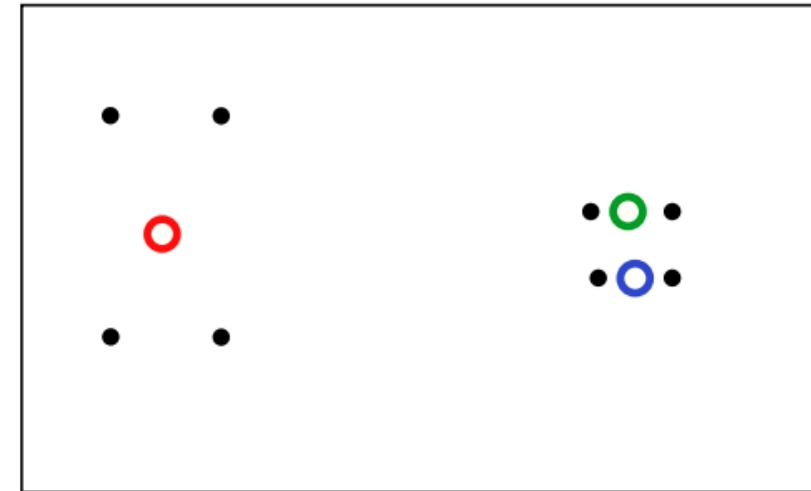
- K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.

# Local Minima

---

- The objective  $J$  is non-convex (so coordinate descent on  $J$  is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points

A bad local optimum



# Soft K-means

---

- Instead of making hard assignments of data points to clusters, we can make **soft assignments**. One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
  - Allows a cluster to use more information about the data in the refitting step.
  - How do we decide on the soft assignments?
  - We already saw this in multi-class classification:
    - 1-of- $K$  encoding vs softmax assignments

# Soft K-means Algorithm

---

- **Initialization:** Set K means  $\{\mathbf{m}_k\}$  to random values
- Repeat until convergence (measured by how much  $J$  changes):
  - **Assignment:** Each data point  $n$  given soft “degree of assignment” to each cluster mean  $k$ , based on responsibilities

$$r_k^{(n)} = \frac{\exp[-\beta \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2]}{\sum_j \exp[-\beta \|\mathbf{m}_j - \mathbf{x}^{(n)}\|^2]}$$

$$\implies \mathbf{r}^{(n)} = \text{softmax}(-\beta \{\|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2\}_{k=1}^K)$$

- **Refitting:** Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

# Questions about Soft K-means

---

Some remaining issues

- How to set  $\beta$ ?
- Clusters with unequal weight and width?

These aren't straightforward to address with K-means. Instead, in the sequel, we'll reformulate clustering using a generative model.

# A Generative View of Clustering

---

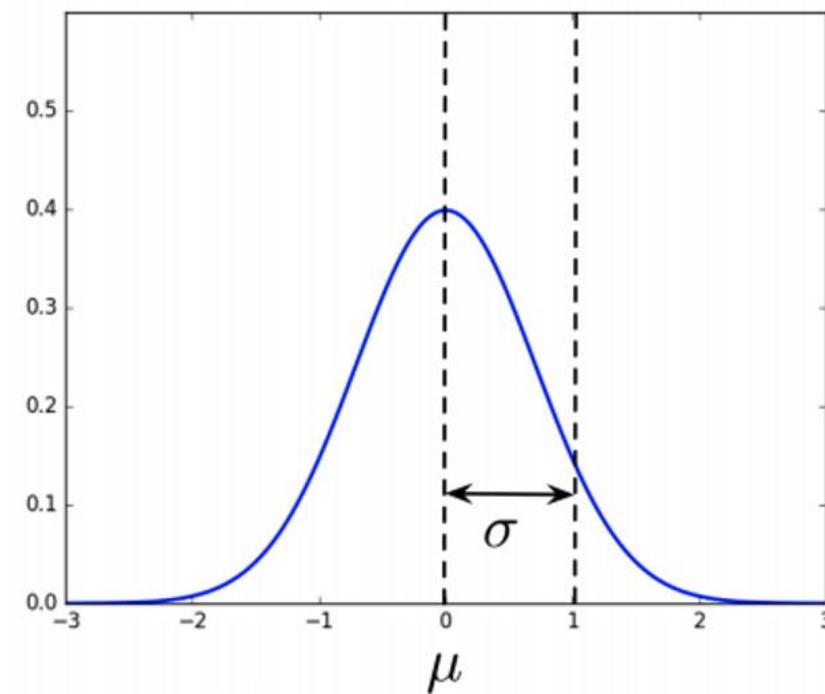
- Next: probabilistic formulation of clustering
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we adjust the model parameters using maximum likelihood i.e. to maximize the probability that it would produce exactly the data we observed

# Univariate Gaussian distribution

- Recall the **Gaussian**, or **normal**, distribution:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

- The central Limit Theorem says that sums of lots of independent random variables are approximately Gaussian.
- In machine learning, we use Gaussians a lot because they make the calculations easy.



# Multivariate Data

---

- Multiple measurements (sensors)
- $D$  inputs / features / attributes
- $N$  instances / observations / examples

$$\mathbf{X} = \begin{bmatrix} [\mathbf{x}^{(1)}]^\top \\ [\mathbf{x}^{(2)}]^\top \\ \vdots \\ [\mathbf{x}^{(N)}]^\top \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_D^{(N)} \end{bmatrix}$$

# Multivariate Mean and Covariance

---

- Mean

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}] = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_d \end{pmatrix}$$

- Covariance

$$\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_D^2 \end{pmatrix}$$

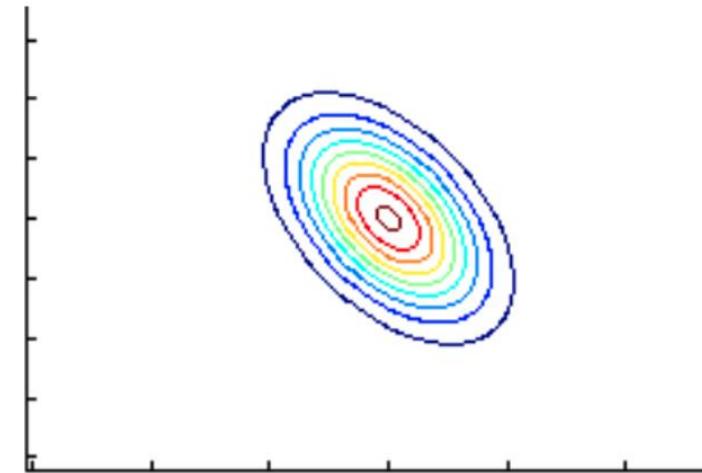
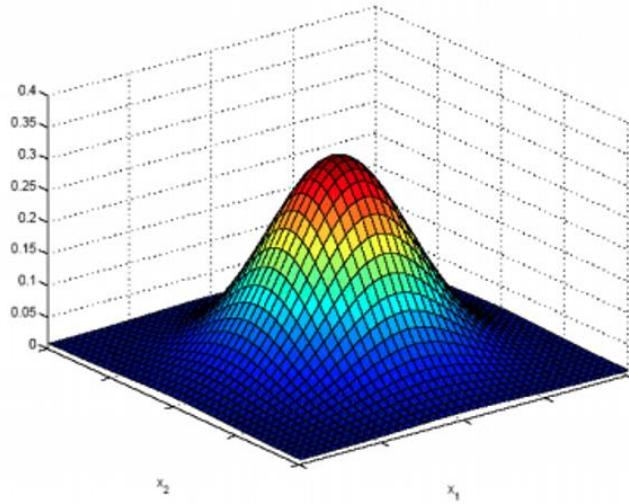
- The statistics ( $\boldsymbol{\mu}$  and  $\boldsymbol{\Sigma}$ ) uniquely define a **multivariate Gaussian** (or **multivariate Normal**) distribution, denoted  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$  or  $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ 
  - This is not true for distributions in general!

# Multivariate Gaussian Distribution

---

- Normally distributed variable  $x \sim \mathcal{N}(\mu, \Sigma)$  has distribution:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$



# The Generative Model

---

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

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

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

# Clusters from Generative Model

---

- This defines joint distribution  $p(z, \mathbf{x}) = p(z)p(\mathbf{x}|z)$  with parameters  $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^K$
- The marginal of  $\mathbf{x}$  is given by  $p(\mathbf{x}) = \sum_z p(z, \mathbf{x})$
- $p(z = k|\mathbf{x})$  can be computed using Bayes rule

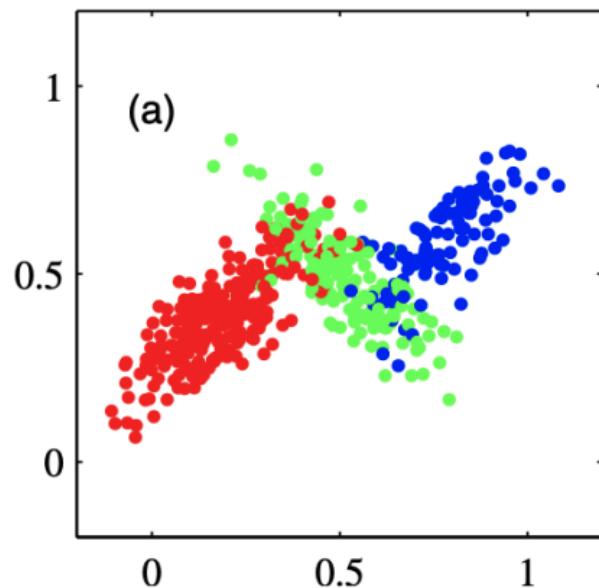
$$p(z = k|\mathbf{x}) = \frac{p(\mathbf{x} | z = k)p(z = k)}{p(\mathbf{x})}$$

and tells us the probability  $\mathbf{x}$  came from the  $k^{\text{th}}$  cluster

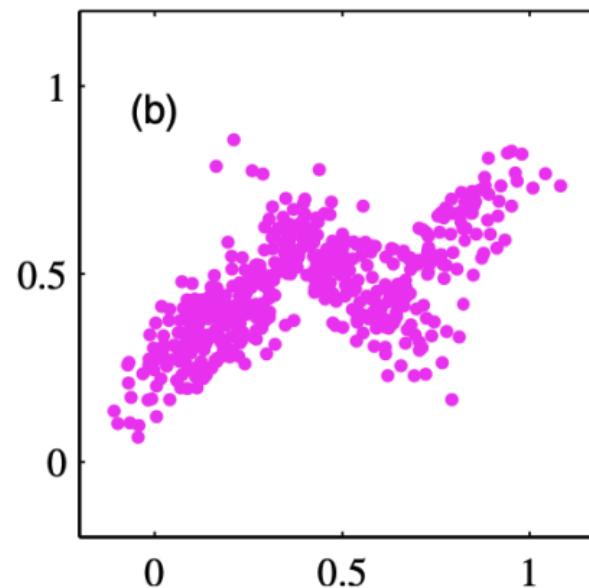
# The Generative Model

---

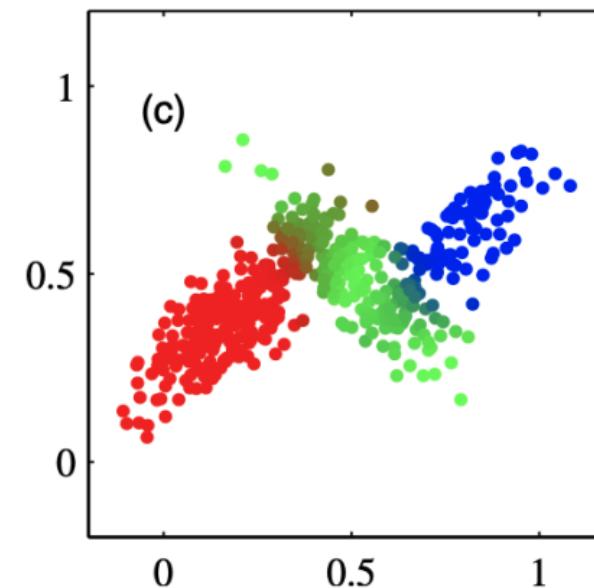
- 500 points drawn from a mixture of 3 Gaussians.



a) Samples from  $p(\mathbf{x} | z)$



b) Samples from the marginal  $p(\mathbf{x})$



c) Responsibilities  $p(z | \mathbf{x})$

# Maximum Likelihood with Latent Variables

---

- How should we choose the parameters  $\{\pi_k, \mu_k\}_{k=1}^K$ ?
- 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)

---

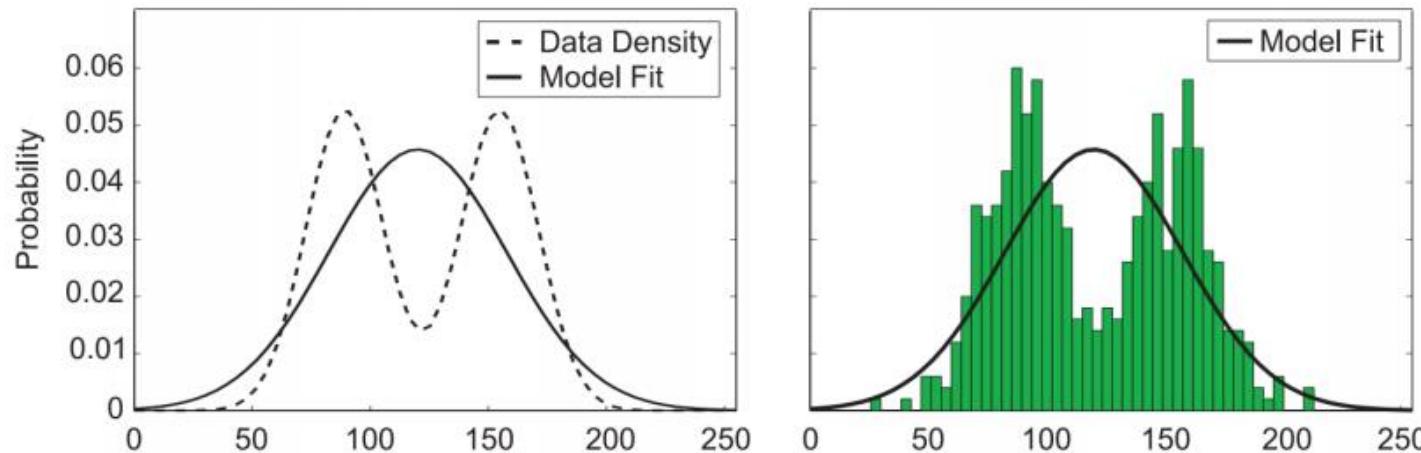
What is  $p(x)$ ?

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

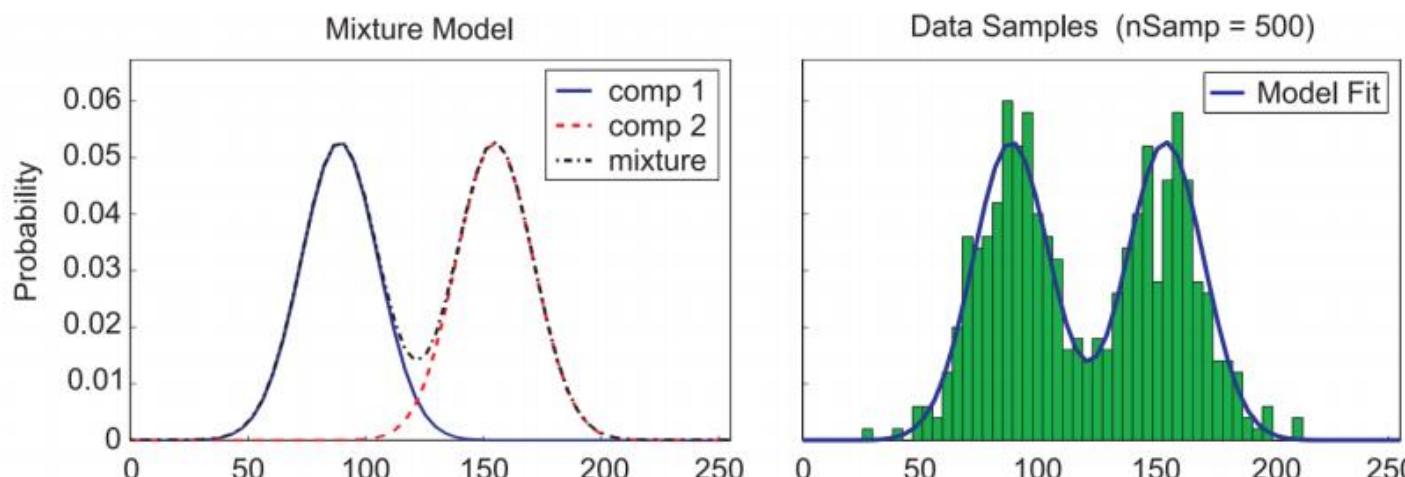
- This distribution is an example of a Gaussian Mixture Model (GMM), and  $\pi_k$  are known as the mixing coefficients
- In general, we would have different covariance for each cluster, i.e.,  $p(\mathbf{x}|z=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ . For this lecture, we assume  $\boldsymbol{\Sigma}_k = \mathbf{I}$  for simplicity.

# Visualizing a Mixture of Gaussians– 1D Gaussians

- If you fit a Gaussian to data:



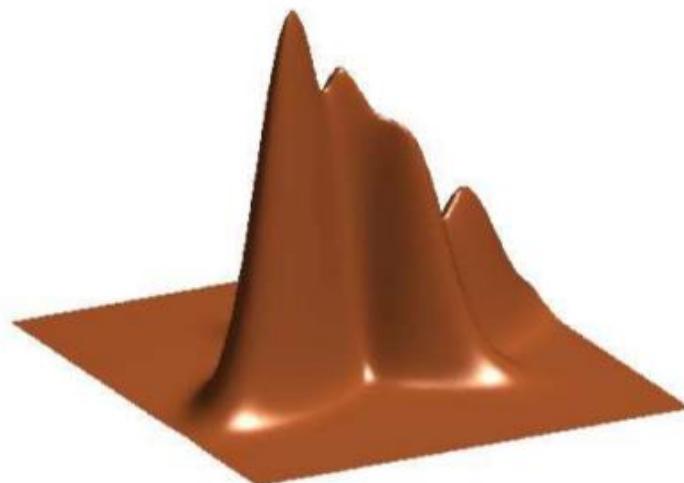
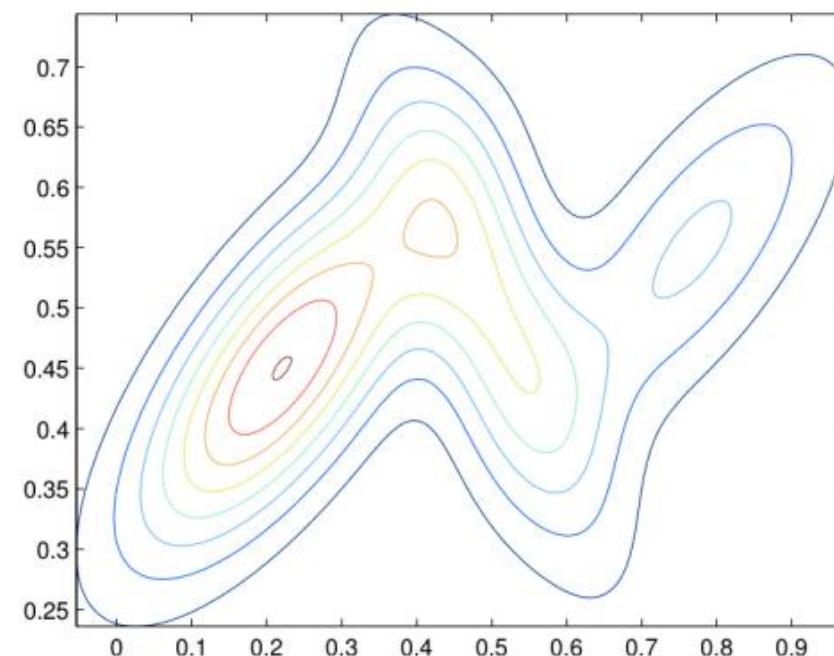
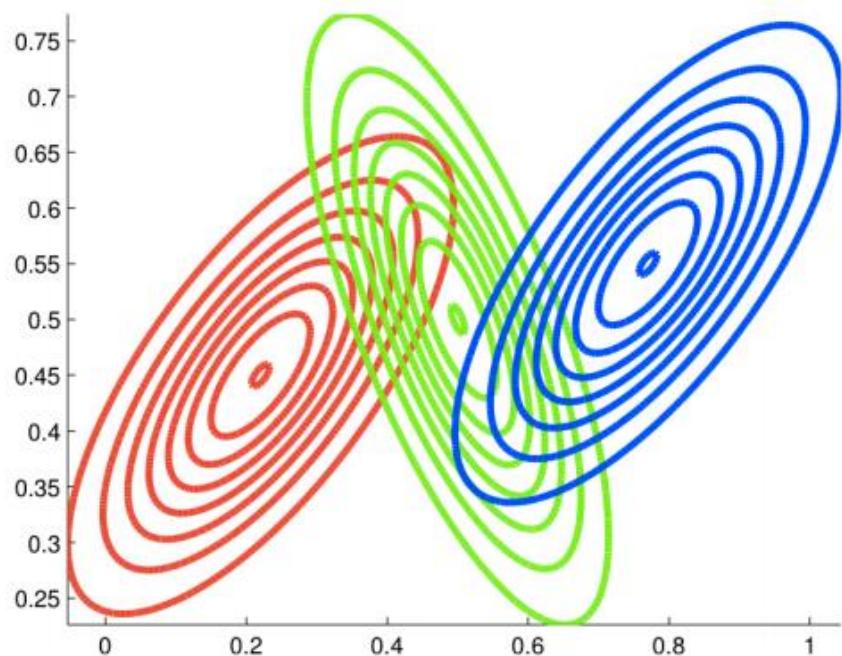
- Now, we are trying to fit a GMM (with K = 2 in this example):



[Slide credit: K. Kutulakos]

# Visualizing a Mixture of Gaussians– 2D Gaussians

---



# Fitting GMMs: Maximum Likelihood

---

Maximum likelihood objective:

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

- How would you optimize this w.r.t. parameters  $\{\pi_k, \boldsymbol{\mu}_k\}$ ?
  - 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:

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

# Maximum Likelihood

---

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

- We have been optimizing something similar for Naive bayes classifiers
- By maximizing  $\log p(\mathcal{D}_{\text{complete}})$ , we would get this:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^N \mathbb{I}[z^{(n)} = k] \mathbf{x}^{(n)}}{\sum_{n=1}^N \mathbb{I}[z^{(n)} = k]} = \text{class means}$$

$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^N \mathbb{I}[z^{(n)} = k] = \text{class proportions}$$

# Maximum Likelihood

---

- We haven't observed the cluster assignments  $z^{(n)}$ , but we can compute  $p(z^{(n)} | \mathbf{x}^{(n)})$  using Bayes rule
- Conditional probability (using Bayes rule) of  $z$  given  $\mathbf{x}$

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

# Maximum Likelihood

---

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

- We don't know the cluster assignments  $\mathbb{I}[z^{(n)} = k]$ , but we know their expectation  $\mathbb{E}[\mathbb{I}[z^{(n)} = k] | \mathbf{x}^{(n)}] = p(z^{(n)} = k | \mathbf{x}^{(n)})$
- If we plug in  $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$  for  $\mathbb{I}[z^{(n)} = k]$ , we get:

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

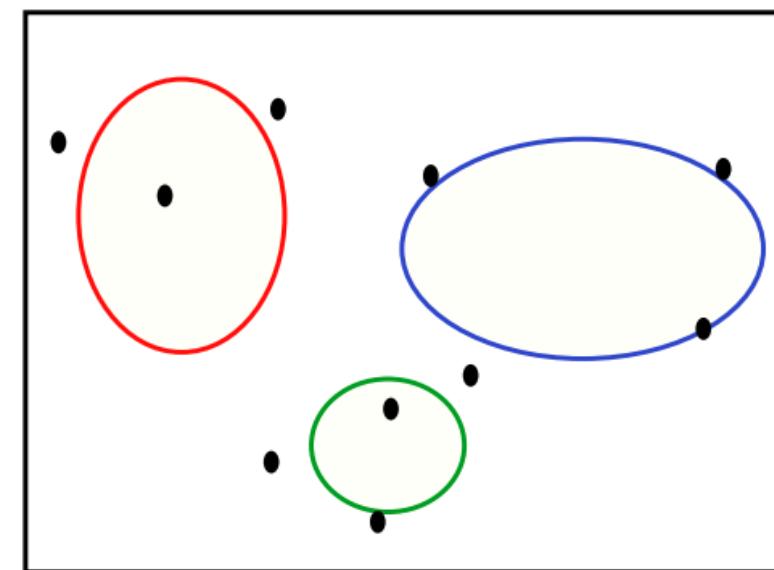
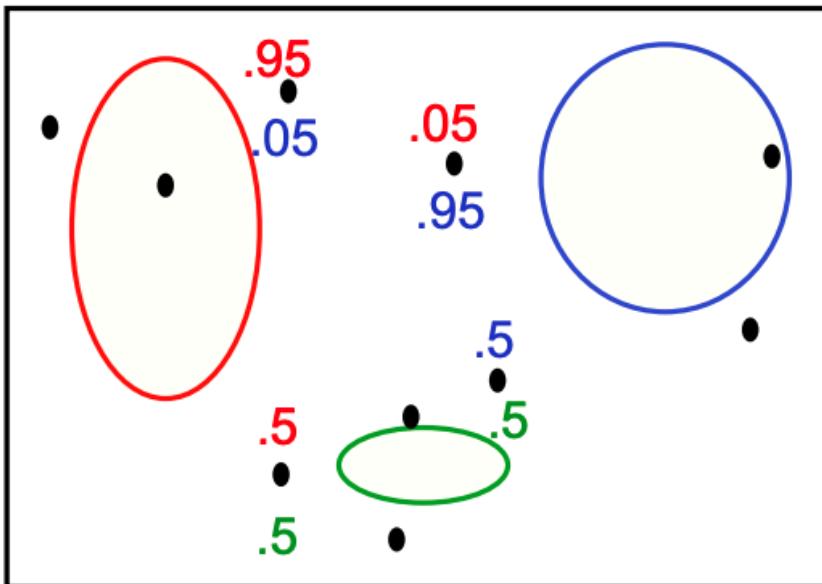
- This is still easy to optimize! Solution is similar to what we have seen:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^N r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^N r_k^{(n)}} \quad \hat{\pi}_k = \frac{\sum_{n=1}^N r_k^{(n)}}{N}$$

- Note: this only works if we treat  $r_k^{(n)} = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_k, \mathbf{I})}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}^{(n)} | \boldsymbol{\mu}_j, \mathbf{I})}$  as fixed.

# How Can We Fit a Mixture of Gaussians?

- This motivates the **Expectation-Maximization** algorithm, which alternates between two steps:
  - E-step:** Compute the posterior probabilities  $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$  our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.
  - 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  $\hat{\mu}_k$  and mixing coefficients  $\hat{\pi}_k$
- Iterate until convergence:

➤ E-step: Evaluate the responsibilities  $r_k^{(n)}$  given current parameters

$$r_k^{(n)} = p(z^{(n)}=k|\mathbf{x}^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)}|\hat{\mu}_k, \mathbf{I})}{\sum_{j=1}^K \hat{\pi}_j \mathcal{N}(\mathbf{x}^{(n)}|\hat{\mu}_j, \mathbf{I})} = \frac{\hat{\pi}_k \exp\{-\frac{1}{2}\|\mathbf{x}^{(n)} - \hat{\mu}_k\|^2\}}{\sum_{j=1}^K \hat{\pi}_j \exp\{-\frac{1}{2}\|\mathbf{x}^{(n)} - \hat{\mu}_j\|^2\}}$$

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

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

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

➤ Evaluate log likelihood and check for convergence

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

## 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:

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

- We don't know  $z^{(n)}$ 's (they are latent), so we replaced  $\mathbb{I}[z^{(n)} = k]$  with responsibilities  $r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)})$
- That is: we replaced  $\mathbb{I}[z^{(n)} = k]$  with its expectation under  $p(z^{(n)} | \mathbf{x}^{(n)})$  (E-step).

# What just happened: A review

---

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

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

which we maximized over parameters  $\{\pi_k, \boldsymbol{\mu}_k\}_k$  (M-step)

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

# 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.