CSC 311: Introduction to Machine Learning

Lecture 11 - K-Means and EM Algorithm

Rahul G. Krishnan

University of Toronto, Fall 2023

Outline

1. K-Means for Clustering
2. Gaussian Mixture Models
3. Expectation-Maximization (E-M)
4. Why EM Works (Optional)

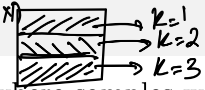
Overview

- In the previous lecture, we covered PCA, Autoencoders and Matrix Factorization—all unsupervised learning algorithms.
  - Each algorithm can be used to approximate high dimensional data using some lower dimensional form.
- Those methods made an interesting assumption that data depends on some latent variables that are never observed. Such models are called latent variable models.
  - For PCA, these correspond to the code vectors (representation).
- Today:
  - K-means, a simple algorithm for clustering, i.e. grouping data points into clusters
  - Reformulate clustering as a latent variable model and apply the EM algorithm

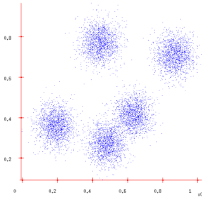
1. K-Means for Clustering
2. Gaussian Mixture Models
3. Expectation-Maximization (E-M)
4. Why EM Works (Optional)

Clustering

$X \in \mathbb{R}^{N\times D}$



- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:
- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.



- Clustering: grouping data points into clusters, with no observed labels. 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).

K-means Intuition

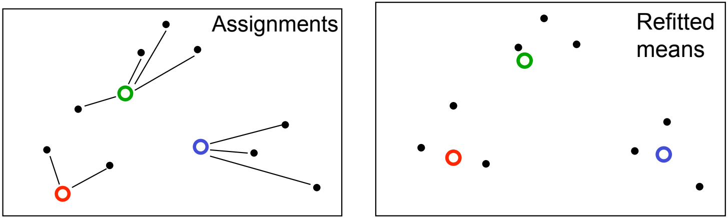
There are k clusters, and each point is close to its cluster center, or mean (the mean of points in the cluster).

How do we compute the cluster assignments?

- Given the cluster assignments, we could easily compute the cluster centers.
- Given the cluster centers, we could easily compute the cluster assignments.
- Chicken and egg problem!
- Simple heuristic - start randomly and alternate between the two!

K-Means

- Randomly initialize cluster centers
- Alternate between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster center to the mean of its members.



K-Means Example

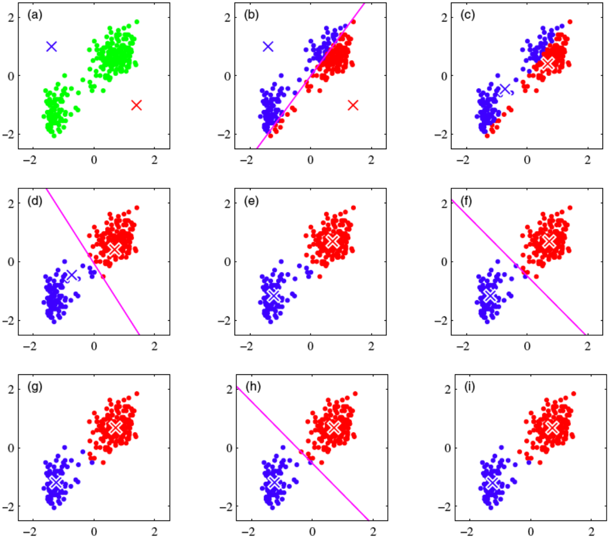
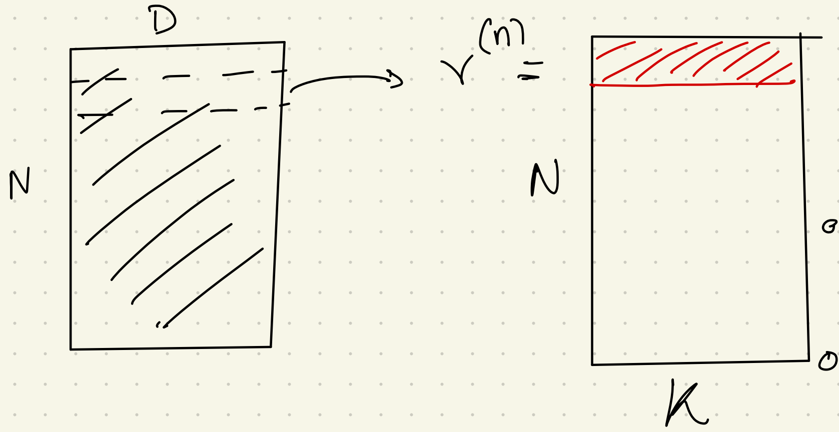


Figure from Bishop

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

What parameters do we need for Kmeans?

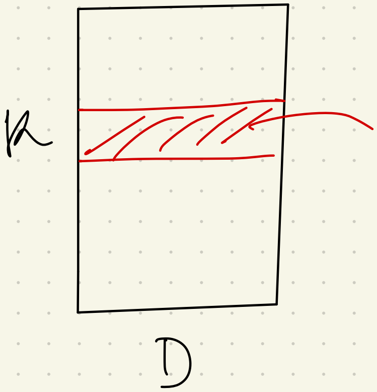
$X \in \mathbb{R}^{N \times D}$



Responsibility matrix

- each row is a one hot vector
- tells us which cluster a point belongs to.

Matrix of Means



$m_K \in \mathbb{R}^D$ nepresents the mean of the k^{th} cluster.

What is K-means Optimizing?

① distance b/w $x^{(n)}$ to kth center

② 0 if $x^{(n)}$ not in kth cluster
distance if $x^{(n)}$ in kth cluster

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

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

$$\text{s.t. } \sum_k r_k^{(n)} = 1, \forall n, \quad \text{where} \quad r_k^{(n)} \in \{0,1\}, \forall k, n$$

where $r_k^{(n)} = 1$ means that $\mathbf{x}^{(n)}$ is assigned to cluster k (with center $\mathbf{m}_k$)

- Finding the exact optimum can be shown to be NP-hard.
- K-means can be seen as block coordinate descent on this objective (analogous to ALS for matrix completion)
  - Assignment step = minimize w.r.t. $\{r_k^{(n)}\}$
  - Refitting step = minimize w.r.t. $\{\mathbf{m}_k\}$

Alternating Minimization

Optimization problem:

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

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

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

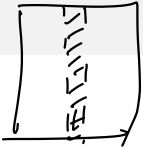
Assign each point to the cluster with the nearest center

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

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

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

Alternating Minimization



Likewise, if we fix the assignments $\{\mathbf{r}^{(n)}\}$ then can easily find optimal centers $\{\mathbf{m}_k\}$

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

- sum of pts in l

- # pts in cluster l

K-Means simply alternates between minimizing w.r.t. assignments and centers. This is an instance of alternating minimization, or block coordinate descent.

The K-means Algorithm

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

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

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

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

Why K-means Converges

- K-means algorithm reduces the cost at each iteration.

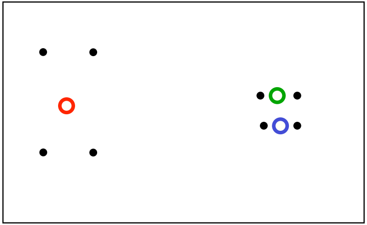
- If the assignments do not change in the assignment step, we have converged (to at least a local minimum).

- Convergence will happen after a finite number of iterations, since the number of possible cluster assignments is finite

Local Minima

- The objective J is non-convex.
- Coordinate descent on J is not guaranteed to converge to the global minimum.
- Nothing prevents k-means getting stuck at local minima.
- We could try many random starting points

A bad local optimum



K-means for Vector Quantization

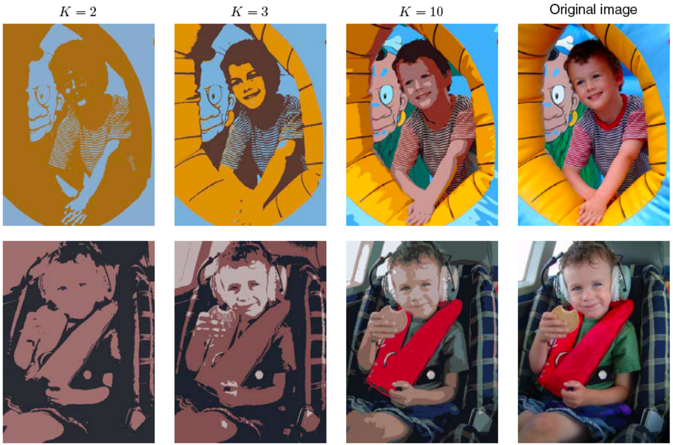
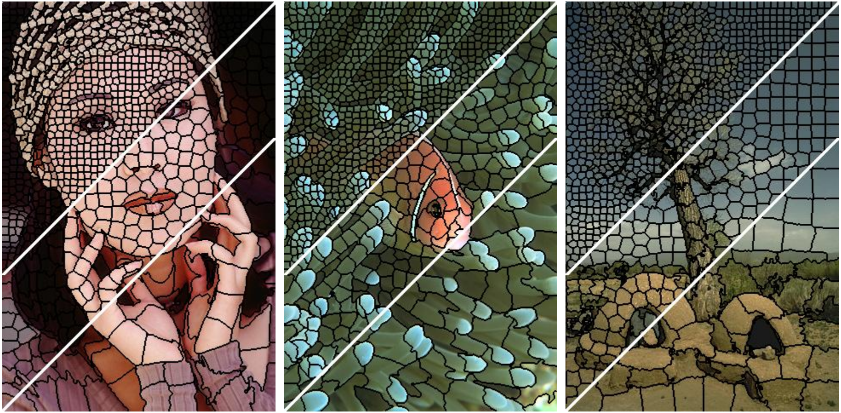


Figure from Bishop

- 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

Soft K-means

- Instead of making hard assignments of data points to clusters, we can make soft assignments.
- For example, one cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
- This 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

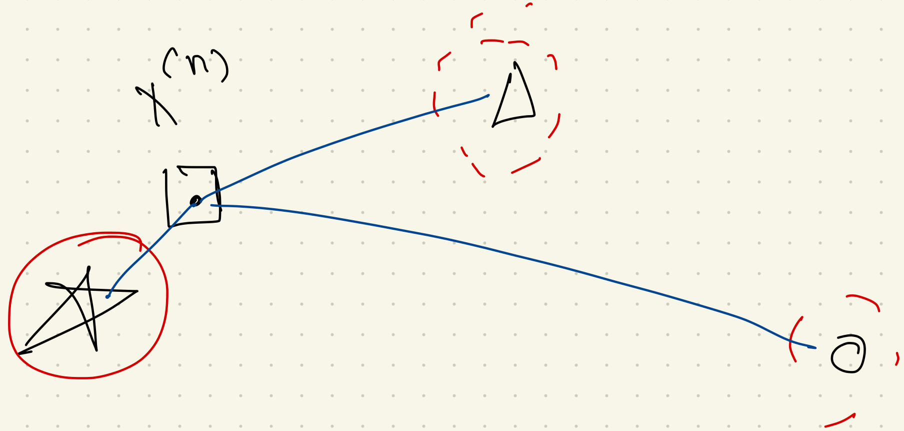
$$r_k^{(n)} = \frac{\exp[-\beta\|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2]}{\sum_j \exp[-\beta\|\mathbf{m}_j - \mathbf{x}^{(n)}\|^2]}$$

- why we multiply by $-\beta$

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

  - Refitting: Cluster centers are adjusted to match sample means of datapoints they are responsible for:

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



	☆	△	○
	5	10	100
$\beta = 1$	-5	-10	-100
	0.8	0.15	0.05

Questions about Soft K-means

Some remaining issues

- How to set β?
- Clusters with unequal weight and width?

These aren't straightforward to address with K-means.

Instead, we'll reformulate clustering using a generative model.

As $\beta \to \infty$, soft k-Means becomes k-Means! (Exercise)

1. K-Means for Clustering
2. Gaussian Mixture Models
3. Expectation-Maximization (E-M)
4. Why EM Works (Optional)

A Generative View of Clustering

- What if the data don't look like spherical blobs?
  - elongated clusters
  - discrete data
- **Remainder of this lecture:** formulating clustering as a probabilistic model
  - specify assumptions about how the observations relate to latent variables
  - use an algorithm called E-M to (approximtely) maximize the likelihood of the observations
- This lets us generalize clustering to non-spherical centers or to non-Gaussian observation models (as in this week's tutorial).
- This lecture is when probabilistic modeling starts to shine!

Generative Models Recap

cat(K) = $\pi_1 \; . \; . \; . \; \pi_k$

prob. that $z == k$

$\sum \pi_k = 1$

- Recall generative (Bayes) classifiers:

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

  - We fit $p(t)$ and $p(\mathbf{x} \,|\, t)$ using labeled data.
- If t is never observed, we call it a latent variable, or hidden variable, and generally denote it with z instead.
  - The things we *can* observe (i.e. $\mathbf{x}$) are called observables.
- By marginalizing out z, we get a density over the observables:

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

- This is called a latent variable model.
- If $p(z)$ is a categorial distribution, this is a mixture model, and different values of z correspond to different components.

Gaussian Mixture Model (GMM)

Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a distribution as

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

  with π_k the mixing coefficients, where:

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

- This defines a density over $\mathbf{x}$, so we can fit the parameters using maximum likelihood. We're try to match the data density of $\mathbf{x}$ as closely as possible.
  - This is a hard optimization problem (and the focus of this lecture).
- GMMs are **universal approximators of densities** (analogously to our universality result for MLPs). Even diagonal GMMs are universal approximators.

Gaussian Mixture Model (GMM)

Story of how the data came to be

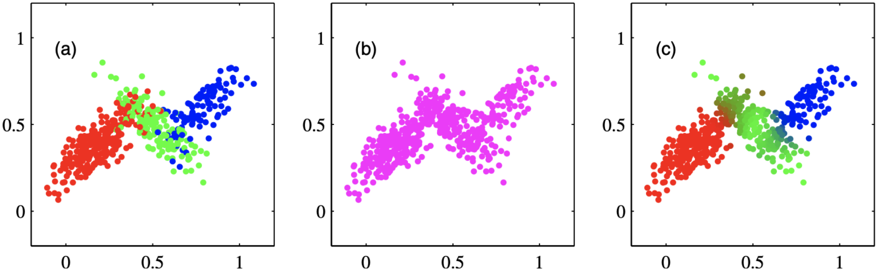
- We can also write the model as a generative process:

For $i = 1, \dots, N$:

$$z^{(i)} \sim \text{Categorical}(\boldsymbol{\pi})$$
$$\mathbf{x}^{(i)} \mid z^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_{z^{(i)}}, \boldsymbol{\Sigma}_{z^{(i)}})$$

The Generative Model

- 500 points drawn from a mixture of 3 Gaussians.



a) Samples from $p(\mathbf{x} \mid z)$ b) Samples from the marginal $p(\mathbf{x})$ c) Responsibilities $p(z \mid \mathbf{x})$

Maximum Likelihood with Latent Variables

- How should we choose the parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$?
- 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:

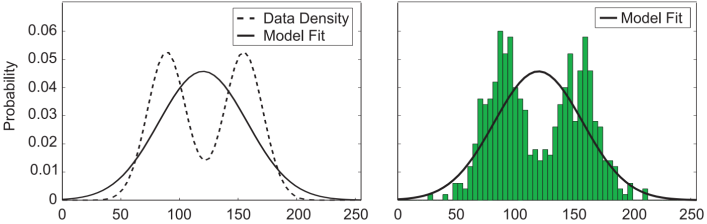
$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$$

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

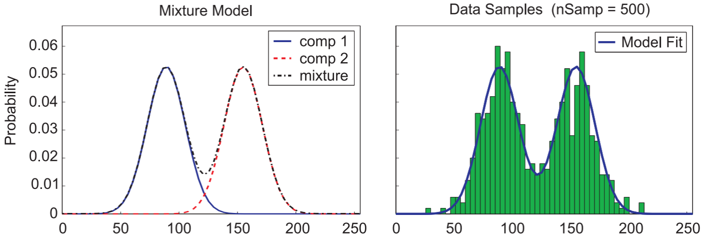
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z = k, \mathbf{x}) = \sum_{k=1}^{K} p(z = k) p(\mathbf{x} | z = k)$$

Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

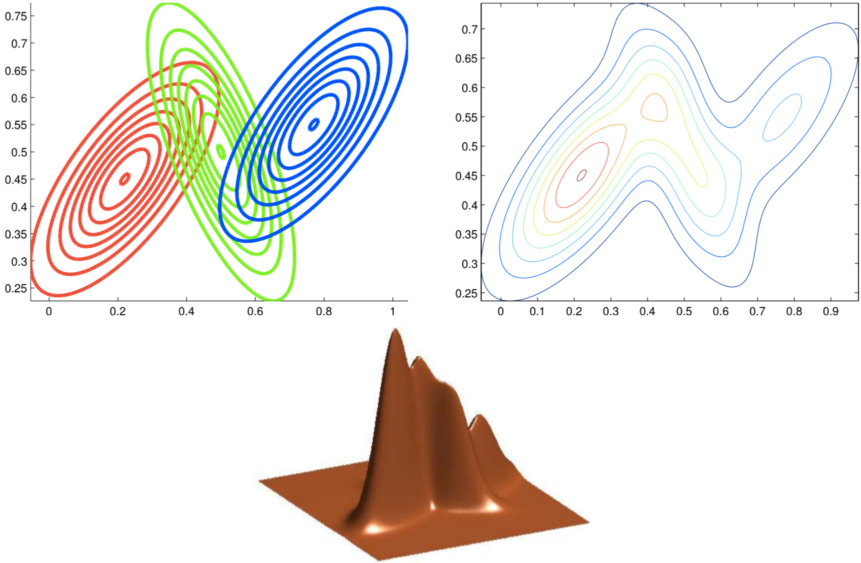


- Now, we are trying to fit a GMM (with $K = 2$ in this example):



[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians



1. K-Means for Clustering
2. Gaussian Mixture Models
3. Expectation-Maximization (E-M)
4. Why EM Works (Optional)

Fitting GMMs: Maximum Likelihood

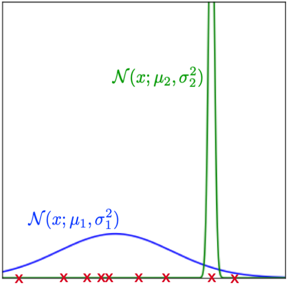
- Some shorthand notation: let $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ denote the full set of model parameters. Let $\mathbf{X} = \{\mathbf{x}^{(i)}\}$ and $\mathbf{Z} = \{z^{(i)}\}$.
- Maximum likelihood objective:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

- In general, no closed-form solution
- Not identifiable: solution is invariant to permutations
- Challenges in optimizing this using gradient descent?
  - Non-convex (due to permutation symmetry)
  - Need to enforce non-negativity constraint on π_k and PSD constraint on $\boldsymbol{\Sigma}_k$
  - Derivatives w.r.t. $\boldsymbol{\Sigma}_k$ are expensive/complicated.
- We need a different approach!

Fitting GMMs: Maximum Likelihood

- **Warning:** you don't want the global maximum. You can achieve arbitrarily high training likelihood by placing a small-variance Gaussian component on a training example.
- This is known as a singularity.



Latent Variable Models: Inference

- If we knew the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, we could infer which component a data point $\mathbf{x}^{(i)}$ probably belongs to by inferring its latent variable $z^{(i)}$.
- This is just posterior inference, which we do using Bayes' Rule:

$$\Pr(z^{(i)} = k \,|\, \mathbf{x}^{(i)}) = \frac{\Pr(z = k)\, p(\mathbf{x} \,|\, z = k)}{\sum_\ell \Pr(z = \ell)\, p(\mathbf{x} \,|\, z = \ell)}$$

- Just like Naïve Bayes, GDA, etc. at test time.

Latent Variable Models: Learning

- If we somehow knew the latent variables for every data point, we could simply maximize the joint log-likelihood.

$$
\begin{aligned}
\log p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \log p(z^{(i)}) + \log p(\mathbf{x}^{(i)} \mid z^{(i)}).
\end{aligned}
$$

- This is just like GDA at training time. Our formulas from Week 8, written in a suggestive notation:

$$
\begin{aligned}
\pi_k &= \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)} \\
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\
r_k^{(i)} &= \mathbb{1}[z^{(i)} = k]
\end{aligned}
$$

Latent Variable Models

- But we *don't* know the $z^{(i)}$, so we need to marginalize them out. Now the log-likelihood is more awkward.

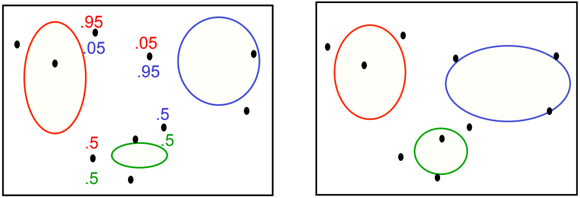
$$
\begin{aligned}
\log p(\mathbf{X}; \boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \log \sum_{z^{(i)}=1}^{K} p(\mathbf{x}^{(i)} \mid z^{(i)}; \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\})\, p(z^{(i)} \mid \boldsymbol{\pi})
\end{aligned}
$$

- Problem: the log is outside the sum, so things don't simplify.
- We have a chicken-and-egg problem, just like with K-Means!
  - Given $\boldsymbol{\theta}$, inferring the $z^{(i)}$ is easy.
  - Given the $z^{(i)}$, learning $\boldsymbol{\theta}$ (with maximum likelihood) is easy.
  - Doing both simultaneously is hard.
- Can you guess the algorithm?

Intuitively, How Can We Fit a Mixture of Gaussians?

- We use the Expectation-Maximization algorithm, which alternates between two steps:

  1. Expectation step (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. Maximization step (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.



Expectation Maximization for GMM Overview

1. E-step: Assign the responsibility $r_k^{(i)}$ of component k for data point i using the posterior probability:

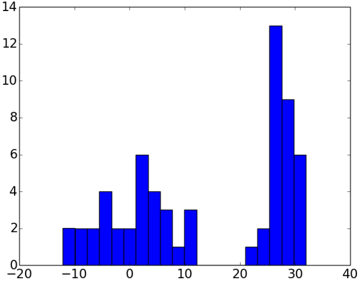
$$r_k^{(i)} = \Pr(z^{(i)} = k \,|\, \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

2. M-step: Apply the maximum likelihood updates, where each component is fit with a weighted dataset. The weights are proportional to the responsibilities.

$$\begin{aligned}
\pi_k &= \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)} \\
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top
\end{aligned}$$

Example

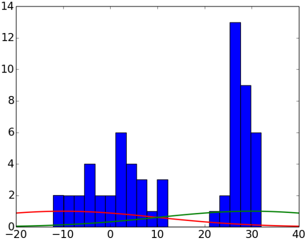
- Suppose we recorded a bunch of temperatures in March for Toronto and Miami, but forgot to record which was which, and they're all jumbled together.



- Let's try to separate them out using a mixture of Gaussians and E-M.

Example

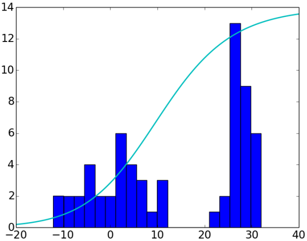
Random initialization



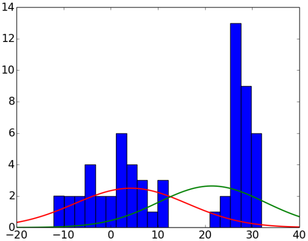
Example

Step 1:

E-step



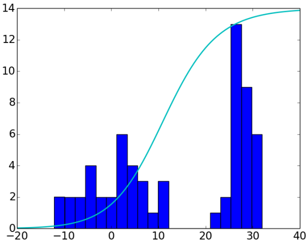
M-step



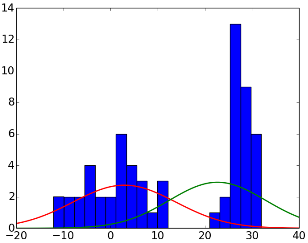
Example

Step 2:

E-step



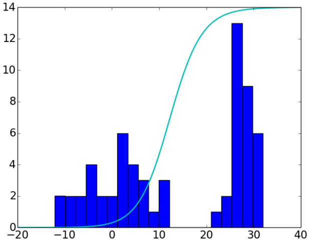
M-step



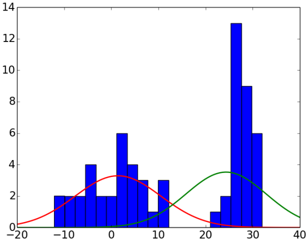
Example

Step 3:

E-step



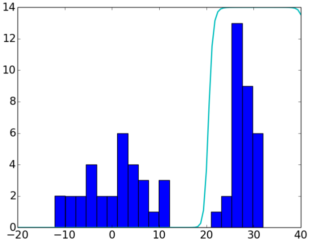
M-step



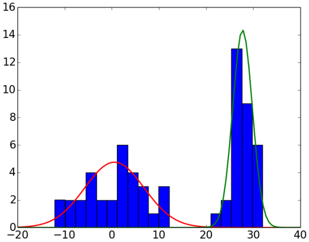
Example

Step 10:

E-step

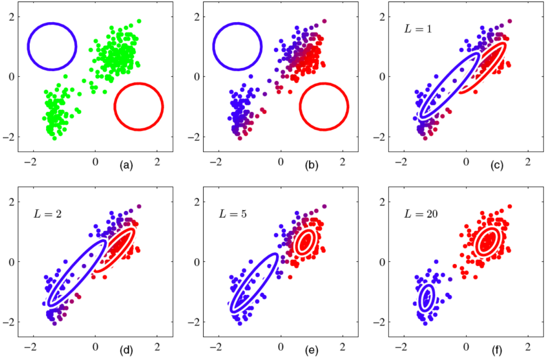


M-step



Expectation-Maximization

- EM for Multivariate Gaussians:



- In tutorial, you will fit a mixture of Bernoullis model.

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.
- Can you find the similarities between the soft k-Means algorithm and EM algorithm with shared covariance $\frac{1}{\beta}\mathbf{I}$?
- Both rely on alternating optimization methods and can suffer from bad local optima.

1. K-Means for Clustering
2. Gaussian Mixture Models
3. Expectation-Maximization (E-M)
4. Why EM Works (Optional)

Jensen's Inequality (optional)

- Recall: if a function f is convex, then

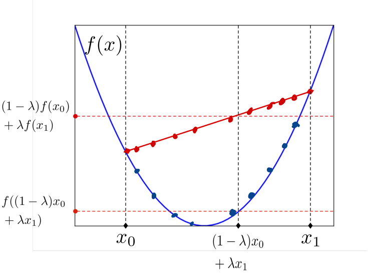
$$\overset{\text{LHS}}{f\left(\sum_i \lambda_i \mathbf{x}_i\right)} \leq \overset{\text{RHS}}{\sum_i \lambda_i f(\mathbf{x}_i)},$$

  where $\{\lambda_i\}$ are such that each $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$.

- If we treat the λ_i as the parameters of a categorical distribution, $\lambda_i = \Pr(\mathbf{X} = \mathbf{x}_i)$, this can be rewritten as:

$$\boxed{f(\mathbb{E}[\mathbf{X}]) \leq \mathbb{E}[f(\mathbf{X})].}$$

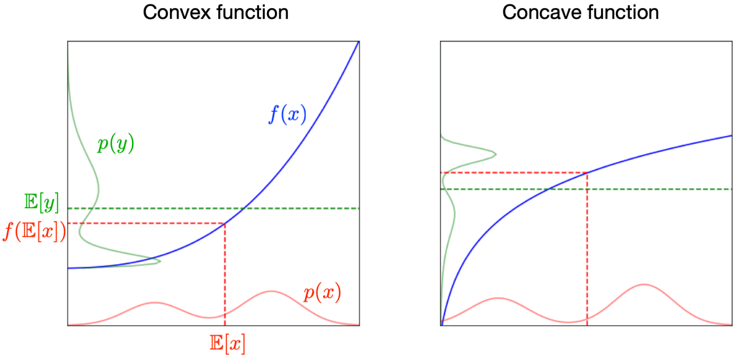
- This is known as Jensen's Inequality. It holds for continuous distributions as well.



Jensen's Inequality (optional)

- A function $f(\mathbf{x})$ is concave if $-f(\mathbf{x})$ is convex. In this case, we flip Jensen's Inequality:

$$f(\mathbb{E}[\mathbf{X}]) \geq \mathbb{E}[f(\mathbf{X})].$$



- When would you expect the inequality to be tight?

Where does EM come from? (optional)

- Recall: the log-likelihood function is awkward because it has a summation inside the log:

$$\log p(\mathbf{X};\boldsymbol{\theta}) = \sum_i \log(p(\mathbf{x}^{(i)};\boldsymbol{\theta})) = \sum_i \log\left(\sum_{z^{(i)}} p(\mathbf{x}^{(i)}, z^{(i)};\boldsymbol{\theta})\right)$$

- Introduce a new distribution $q(z^{(i)})$ (we'll see what this is shortly):

$$\log p(\mathbf{X};\boldsymbol{\theta}) = \sum_i \log\left(\sum_{z^{(i)}} q(z^{(i)})\,\frac{p(\mathbf{x}^{(i)}, z^{(i)};\boldsymbol{\theta})}{q(z^{(i)})}\right)$$

(multiply & divide by $q(z^{(i)})$)

$$= \sum_i \log \mathbb{E}_{q(z^{(i)})}\left[\frac{p(\mathbf{x}^{(i)}, z^{(i)};\boldsymbol{\theta})}{q(z^{(i)})}\right]$$

- Notice that log is a concave function. So we can use Jensen's Inequality to push the log inwards, obtaining the variational lower bound:

$$\log p(\mathbf{X};\boldsymbol{\theta}) \geq \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)};\boldsymbol{\theta})}{q(z^{(i)})}\right] \triangleq \mathcal{L}(q,\boldsymbol{\theta})$$

Where does EM come from? (optional)

- Just derived a lower bound on the log-likelihood:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) \geq \sum_i \mathbb{E}_{q(z^{(i)})} \left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] \triangleq \mathcal{L}(q, \boldsymbol{\theta})$$

- Simplifying the right-hand-side:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_i \mathbb{E}_{q(z^{(i)})}[\log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})] - \underbrace{\mathbb{E}_{q(z^{(i)})}[\log q(z^{(i)})]}_{\text{constant w.r.t. } \boldsymbol{\theta}}$$

- The expected log-probability will turn out to be nice.

Where does EM come from? (optional)

- Everything so far holds for any choice of q. But what should we actually pick?
- Jensen's inequality gives a lower bound on the log-likelihood, so the best we can achieve is to make the bound tight (i.e. equality).
- Denote the current parameters as $\boldsymbol{\theta}^{\text{old}}$.
- It turns out the posterior probability $p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})$ is a very good choice for q. Plugging it in to the lower bound:

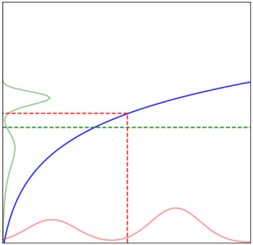
$$\begin{aligned}
\sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{\text{old}})}{q(z^{(i)})}\right] &= \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{\text{old}})}{p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})}\right] \\
&= \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})\right] \\
&= \sum_i \log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}}) \\
&= \log p(\mathbf{X}; \boldsymbol{\theta}^{\text{old}})
\end{aligned}$$

- Equality achieved!

Where does EM come from? (optional)

An aside:

- How could you pick $q(z^{(i)}) = p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})$ if you didn't already know the answer?
- Observe: if f is strictly concave, then Jensen's inequality becomes an equality exactly when the random variable X is determinisic.



- Hence, to solve

$$\log \mathbb{E}_{q(z^{(i)})} \left[\frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] = \mathbb{E}_{q(z^{(i)})} \left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right],$$

we should set $q(z^{(i)}) \propto p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})$.

Where does EM come from? (optional)

- **E-step:** compute the responsibilities using Bayes' Rule:

$$r_k^{(i)} \triangleq q(z^{(i)} = k) = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})$$

- Rewriting the variational lower bound in terms of the responsibilities:

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) = & \sum_i \sum_k r_k^{(i)} \log \Pr(z^{(i)} = k; \boldsymbol{\pi}) \\ & + \sum_i \sum_k r_k^{(i)} \log p(\mathbf{x}^{(i)} \mid z^{(i)} = k; \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}) \\ & + \text{const}\end{aligned}$$

- **M-step:** maximize $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, giving $\boldsymbol{\theta}^{\text{new}}$. This can be done analytically, and gives the parameter updates we saw previously.
- The two steps are guaranteed to improve the log-likelihood:

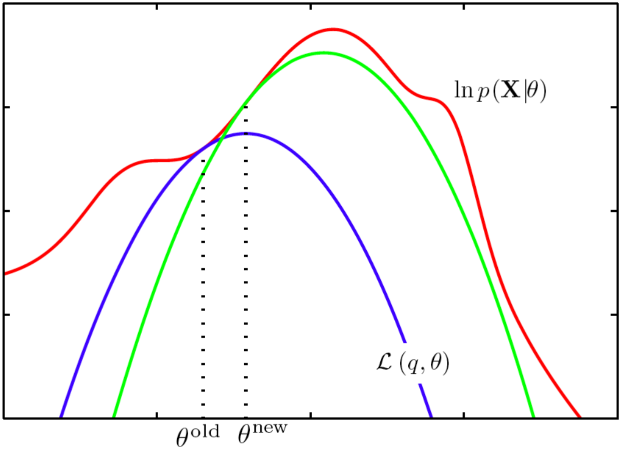
$$\log p(\mathbf{X}; \boldsymbol{\theta}^{\text{new}}) \geq \mathcal{L}(q, \boldsymbol{\theta}^{\text{new}}) \geq \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}}) = \log p(\mathbf{X}; \boldsymbol{\theta}^{\text{old}}).$$

EM: Recap (optional)

Recap of EM derivation:

- We're trying to maximize the log-likelihood $\log p(\mathbf{X}; \boldsymbol{\theta})$.
- The exact log-likelihood is awkward, but we can use Jensen's Inequality to lower bound it with a nicer function $\mathcal{L}(q, \boldsymbol{\theta})$, the variatonal lower bound, which depends on a choice of q.
- The **E-step** chooses q to make the bound tight at the current parameters $\boldsymbol{\theta}^{\text{old}}$. Mechanistically, this means computing the responsibilities $r_k^{(i)} = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})$.
- The **M-step** maximizes $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, giving $\boldsymbol{\theta}^{\text{new}}$. For GMMs, this can be done analytically.
- The combination of the E-step and M-step is guaranteed to improve the true log-likelihood.

Visualization of the EM Algorithm (optional)



- The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values.

GMM E-Step: Responsibilities (optional)

Lets see how it works on GMM:

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

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

GMM E-Step (optional)

- Once we computed $r_k^{(i)} = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)})$ we can compute the expected likelihood

$$
\begin{aligned}
& \mathbb{E}_{p(z^{(i)} \mid \mathbf{x}^{(i)})} \left[\sum_i \log(p(\mathbf{x}^{(i)}, z^{(i)} \mid \boldsymbol{\theta})) \right] \\
= & \sum_i \sum_k r_k^{(i)} \Big(\log(\Pr(z^{(i)} = k \mid \boldsymbol{\theta})) + \log(p(\mathbf{x}^{(i)} \mid z^{(i)} = k, \boldsymbol{\theta})) \Big) \\
= & \sum_i \sum_k r_k^{(i)} \Big(\log(\pi_k) + \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \Big) \\
= & \sum_k \sum_i r_k^{(i)} \log(\pi_k) + \sum_k \sum_i r_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))
\end{aligned}
$$

- We need to fit k Gaussians, just need to weight examples by r_k

GMM M-Step (optional)

- Need to optimize

$$\sum_k \sum_i r_k^{(i)} \log(\pi_k) + \sum_k \sum_i r_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))$$

- Solving for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ is like fitting k separate Gaussians but with weights $r_k^{(i)}$.
- Solution is similar to what we have already seen:

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} \mathbf{x}^{(i)} \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T \\
\pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{i=1}^{N} r_k^{(N)}
\end{aligned}$$

EM Algorithm for GMM (optional)

- Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

$$r_k^{(i)} = p(z^{(i)} \,|\, \mathbf{x}^{(i)}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)} \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

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

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} \mathbf{x}^{(i)} \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{i=1}^{N} r_k^{(i)}
\end{aligned}$$

  - Evaluate log likelihood and check for convergence

$$\log p(\mathbf{X} \,|\, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(i)} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

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 models are very powerful models, i.e. universal distribution approximators
- Optimization is done using the EM algorithm.