CSC 2515: Introduction to Machine Learning

Lecture 10: K-Means and EM Algorithm

Amir-massoud Farahmand[1]

University of Toronto and Vector Institute

[1] Credit for slides goes to many members of the ML Group at the U of T, and beyond, including (recent past): Roger Grosse, Murat Erdogdu, Richard Zemel, Juan Felipe Carrasquilla, Emad Andrews, and myself.

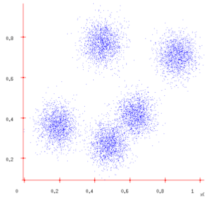
Table of Contents

Overview

- We have covered PCA, which was an unsupervised learning algorithm.
  - Its main purpose was to reduce the dimension of the data.
  - In practice, even though data is very high dimensional, it can be well represented in low dimensions.
- This method relies on an assumption that data depends on some latent variables, which are not observed. Such models are called latent variable models.
  - For PCA, these corresponds to the code vectors (representation).
  - Today's lecture: K-means, a simple algorithm for clustering, i.e., grouping data points into clusters
  - Today's lecture: Reformulate clustering as a latent variable model, apply the Expectation-Maximization (EM) algorithm

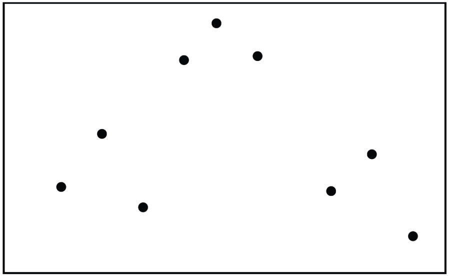
Clustering Problem

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

Clustering Problem



- Assume that the data points $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ live in an Euclidean space, i.e., $\mathbf{x}^{(n)} \in \mathbb{R}^D$.
- Assume that each data point belongs to one of the K clusters
- Assume that the data points from the same cluster are similar, i.e., close in Euclidean distance.
- How can we identify those clusters and the data points that belong to each cluster?

K-means Objective

Let's formulate this as an optimization problem

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

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

  where $r_k^{(n)} = \mathbb{I}\{\mathbf{x}^{(n)}$ is assigned to cluster $k\}$, that is, $\mathbf{r}^{(n)} = [0, \ldots, 1, \ldots, 0]^\top$.
- Finding an optimal solution is an NP-hard problem!

K-means Objective

- Optimization problem:

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

- Since $r_k^{(n)} = \mathbb{I}\{\mathbf{x}^{(n)} \text{ is assigned to cluster } k\}$ (e.g., $\mathbf{r}^{(n)} = [0, \ldots, 1, \ldots, 0]^\top$), the inner sum is over K terms but only one of them is non-zero.
- For example, if data point $\mathbf{x}^{(n)}$ is assigned to cluster $k = 3$, then $\mathbf{r}^n = [0, 0, 1, 0, ...]$ and

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

How to Optimize? Alternating Minimization

Optimization problem:

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

- Problem is hard when minimizing jointly over the parameters $\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}$.
- But if we fix one and minimize over the other, then it becomes easy.
- Idea: We can alternate between optimizing $\mathbf{r}$ (assignments) and $\mathbf{m}$ (centres).
- Doesn't guarantee the same solution!

Alternating Minimization (Optimizing Assignments)

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 centres $\{\mathbf{m}_k\}$, we can easily find the optimal assignments $\{\mathbf{r}^{(n)}\}$ for each sample n

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

    - Assign each point to the cluster with the nearest centre

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

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

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

Alternating Minimization (Optimizing Centres)

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

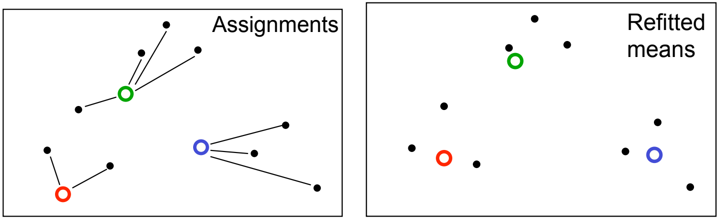
$$
\begin{aligned}
0 =& \frac{\partial}{\partial \mathbf{m}_l} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2 \\
=& 2 \sum_{n=1}^{N} r_l^{(n)} (\mathbf{m}_l - \mathbf{x}^{(n)}) \qquad \Longrightarrow \ \mathbf{m}_l = \frac{\sum_n r_l^{(n)} \mathbf{x}^{(n)}}{\sum_n r_l^{(n)}}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

- 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 centres
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster centre to the mean of the data assigned to it



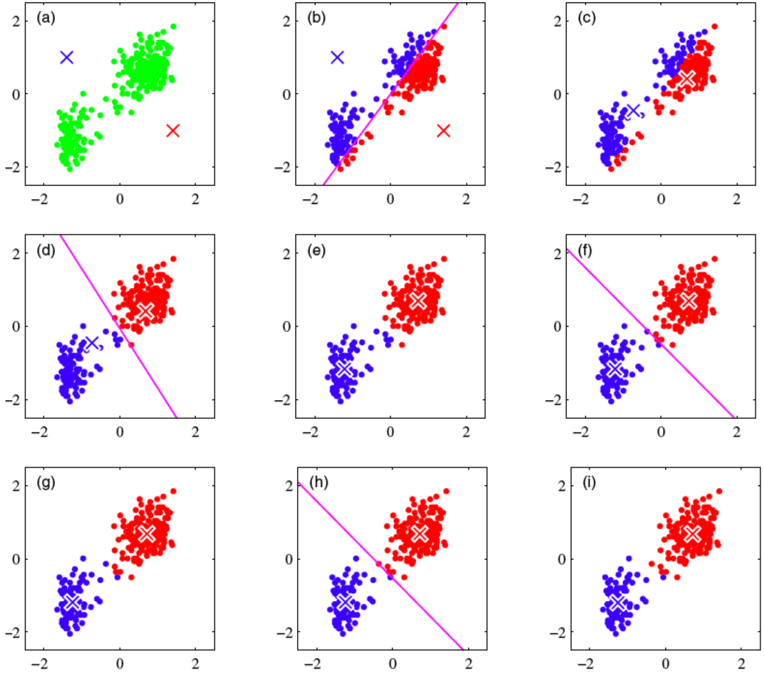


Figure from Bishop

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

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 J w.r.t. $\{\mathbf{r}\}$: Each data point $\mathbf{x}^{(n)}$ is assigned to nearest centre

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

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

$$r_k^{(n)} = \mathbb{I}\{\hat{k}^{(n)} = k\} \;\; \text{for} \;\; k = 1, .., K$$

  - Refitting: Optimize J w.r.t. $\{\mathbf{m}\}$: Each centre 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

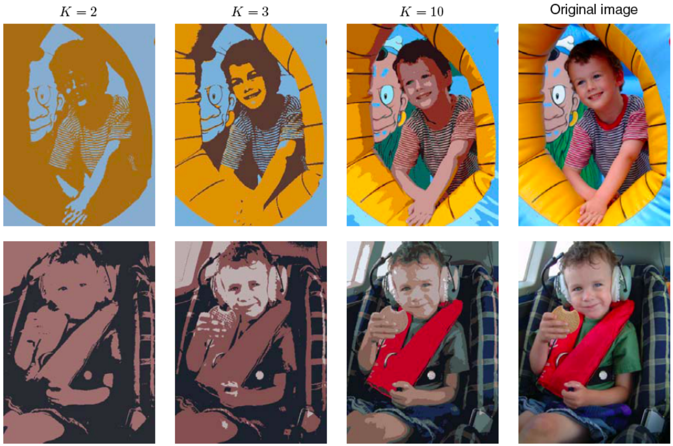


Figure from Bishop

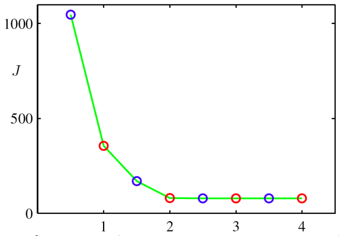
- Given image, construct "dataset" of pixels represented by their RGB pixel intensities
- Run K-means, replace each pixel by its cluster centre

Questions about K-means

- 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

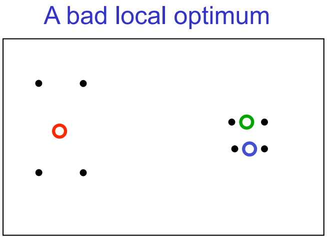
- K-means algorithm reduces the cost at each iteration.
  - Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centres is reduced.
  - Whenever a cluster centre is moved, J is reduced.
- Test for convergence: If the assignments do not change in the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



- 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



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 0.7 for a datapoint and another may have a responsibility of 0.3.
  - 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

$$r_k^{(n)} = \frac{\exp(-\beta\|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2)}{\sum_{j=1}^{K} \exp(-\beta\|\mathbf{m}_j - \mathbf{x}^{(n)}\|^2)}$$

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

  - Refitting: Model parameters (i.e., centre 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 β?
- Clusters with unequal weight and width?

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

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

A Generative View of Clustering

- Next: 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 will 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

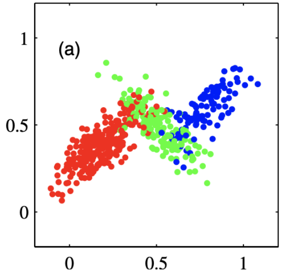
- This defines joint distribution $p(z, \mathbf{x}) = p(z)p(\mathbf{x}|z)$ with parameters $\{(\pi_k, \boldsymbol{\mu}_k)\}_{k=1}^K$
- The marginal of $\mathbf{x}$ is given by $p(\mathbf{x}) = \sum_z p(z, \mathbf{x})$
- $p(z = k|\mathbf{x})$ can be computed using Bayes rule

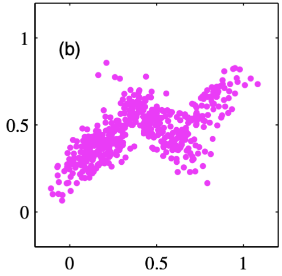
$$p(z = k|\mathbf{x}) = \frac{p(\mathbf{x} \mid z = k)p(z = k)}{p(\mathbf{x})}.$$

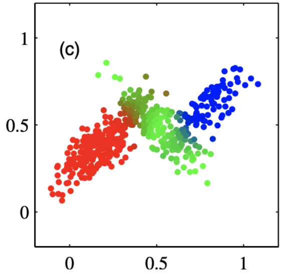
This tells us the probability that $\mathbf{x}$ comes from the k^{th} cluster.

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 $\{(\pi_k, \boldsymbol{\mu}_k)\}_{k=1}^{K}$?
- Maximum likelihood principle: choose parameters to maximize the 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)

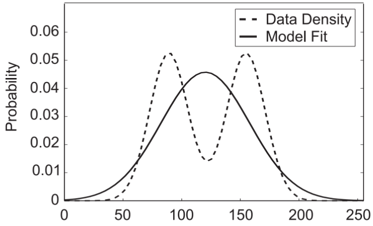
What is $p(\mathbf{x})$?

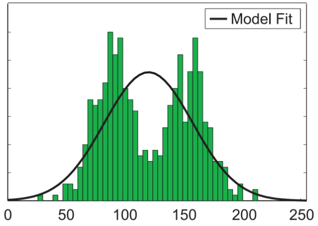
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z=k)p(\mathbf{x}|z=k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I})$$

- This distribution is an example of a Gaussian Mixture Model (GMM), and π_k are known as the mixing coefficients
- In general, we would have different covariance for each cluster, i.e., $p(\mathbf{x} \mid z=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. For this lecture, we assume that $\boldsymbol{\Sigma}_k = \mathbf{I}$ for simplicity.
- If we allow arbitrary covariance matrices, GMMs are **universal approximators of densities** (if you have enough Gaussians). Even diagonal GMMs are universal approximators.

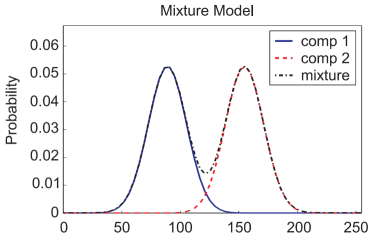
Visualizing a Mixture of Gaussians – 1D Gaussians

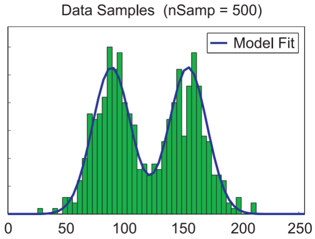
- If you fit one Gaussian distribution to data:





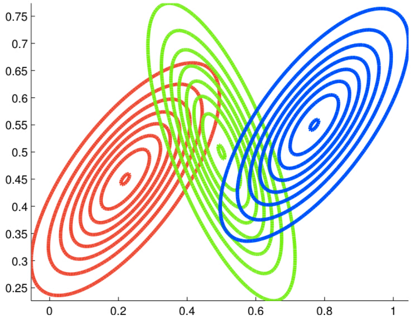
- Now, we are trying to fit a GMM with $K = 2$:

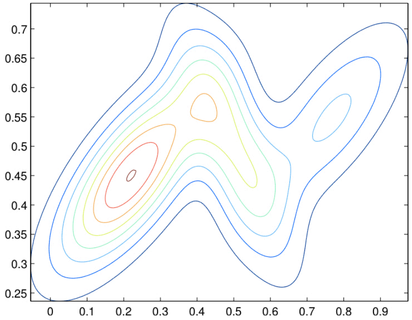


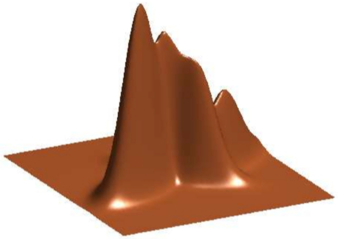


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs: Maximum Likelihood

Maximum likelihood objective:

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

- How would you optimize this w.r.t. parameters $\boldsymbol{\theta} = \{(\pi_k, \boldsymbol{\mu}_k)\}$?
  - 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 would be easy:

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

Maximum Likelihood

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

- We have been optimizing something similar for the Naïve Bayes classifiers.
- By maximizing $\log p(\mathcal{D}_{\text{complete}} \mid \boldsymbol{\theta})$, we would get this:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \frac{\sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\} \, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\}} = \text{class means} \\ \hat{\pi}_k &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\{z^{(n)} = k\} = \text{class proportions} \end{aligned}$$

Maximum Likelihood

- But we haven't observed the cluster assignments $z^{(n)}$.
- We can compute $p(z^{(n)}|\mathbf{x}^{(n)})$ using Bayes rule, given our current estimate $\boldsymbol{\theta}^{\text{old}}$.
- 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 $\mathbb{I}\{z^{(n)}=k\}$ (they are our latent variables), but we know their expectation w.r.t. the current estimate $\boldsymbol{\theta}^{\text{old}}$:
$$\mathbb{E}[\mathbb{I}\{z^{(n)}=k\} \mid \mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}}] = p(z^{(n)}=k|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}}).$$
- So we consider
$$\sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)}|\mathbf{x}^{(n)};\boldsymbol{\theta}^{\text{old}})}\left[\sum_{k=1}^{K} \mathbb{I}\{z^{(n)} = k\}(\log \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k)\right] =$$
$$\sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)}(\log \mathcal{N}(\mathbf{x}^{(n)}|\boldsymbol{\mu}_k, \mathbf{I}) + \log \pi_k),$$
with $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$.

Maximum Likelihood

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

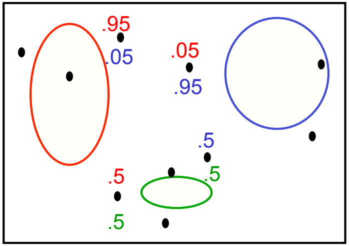
- This is still easy to optimize! Solution is similar to what we have seen:

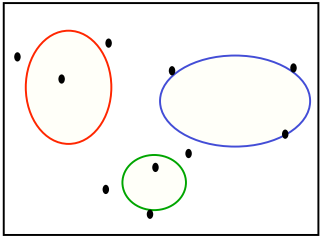
$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} r_k^{(n)}} \qquad \hat{\pi}_k = \frac{\sum_{n=1}^{N} r_k^{(n)}}{N}$$

- 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, i.e., depending on $\boldsymbol{\theta}^{\text{old}}$.

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 $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ 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 $\hat{\boldsymbol{\mu}}_k$ and mixing coefficients $\hat{\pi}_k$
- Iterate until convergence:
  - E-step: Evaluate the responsibilities $r_k^{(n)}$ given current parameters

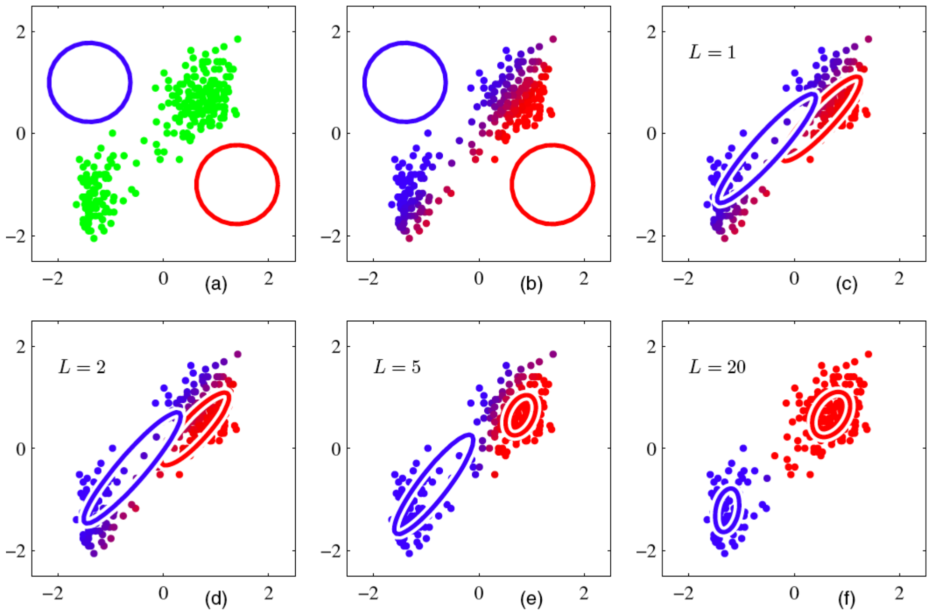
$$r_k^{(n)} = p(z^{(n)} = k | \mathbf{x}^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_k, \mathbf{I})}{\sum_{j=1}^K \hat{\pi}_j \mathcal{N}(\mathbf{x}^{(n)} | \hat{\boldsymbol{\mu}}_j, \mathbf{I})} = \frac{\hat{\pi}_k \exp(-\frac{1}{2}\|\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_k\|^2)}{\sum_{j=1}^K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

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

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k &= \frac{1}{N_k} \sum_{n=1}^N r_k^{(n)} \mathbf{x}^{(n)} \\ \hat{\pi}_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^N r_k^{(n)} \end{aligned}$$

  - Evaluate log likelihood and check for convergence

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



(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:

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

- 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:

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

  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. expectations $\mathbb{E}[\mathbb{I}\{z^{(n)} = k\}|\mathbf{x}^{(n)}]$) given the current model parameters $(\pi_k, \boldsymbol{\mu}_k)$
  - 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 centre 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.

Further Discussion

- We assumed that the covariance of each Gaussian was $\mathbf{I}$ to simplify the math. This assumption can be removed, allowing clusters to have different spatial spreads. The resulting algorithm is still very simple.
- Possible problems with maximum likelihood objective:
  - Singularities: Arbitrarily large likelihood when a Gaussian explains a single point with variance shrinking to zero
  - Non-convex
- EM is more general than what was covered in this lecture. Here, EM algorithm is used to find the optimal parameters under the GMMs.

GMM Recap

- A probabilistic view of clustering. Each cluster corresponds to a different Gaussian.
- Model using latent variables.
- General approach. We 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.