ML4 B&I: Introduction to Machine Learning

Lecture 7- Clustering Algorithms

Murat A. Erdogdu

Vector Institute, Fall 2022

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.

Overview

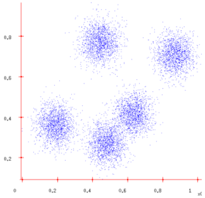
- In the previous lecture, we covered PCA, Autoencoders and Matrix Factorization—all unsupervised learning algorithms.
  - Each algorithm can be used to approximate high dimensional data using some lower dimensional form.
- Those methods made an interesting assumption that data depends on some latent variables that are never observed. Such models are called latent variable models.
  - For PCA, these correspond to the code vectors (representation).

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'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

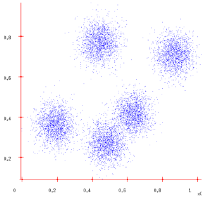
Clustering

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



Clustering

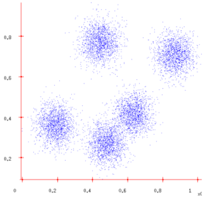
- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:



- Such a distribution is multimodal, since it has multiple modes.

Clustering

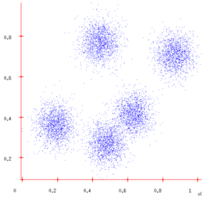
- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:



- Such a distribution is multimodal, since it has multiple modes.
- Grouping data points into clusters, with no observed labels, is called clustering. It is an unsupervised learning technique.

Clustering

- 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.
- 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).

K-means intuition

- K-means assumes there are k clusters, and each point is close to its cluster center, or mean (the mean of points in the cluster).
  - If we knew the cluster assignment, we could easily compute the centers.
  - If we knew the centers, we could easily compute cluster assignment.
  - Chicken and egg problem!

K-means intuition

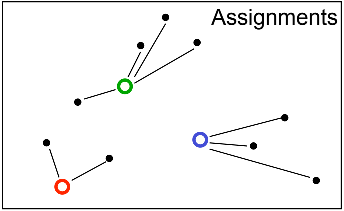
- K-means assumes there are k clusters, and each point is close to its cluster center, or mean (the mean of points in the cluster).
    - If we knew the cluster assignment, we could easily compute the centers.
    - If we knew the centers, we could easily compute cluster assignment.
    - Chicken and egg problem!
- Very simple (and useful) heuristic - start randomly and alternate between the two!

K-means

- Initialization: randomly initialize cluster centers

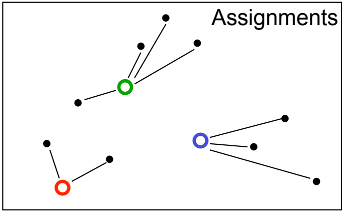
K-means

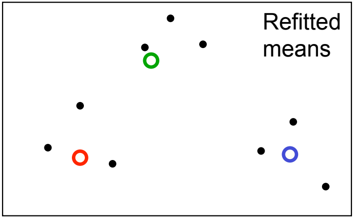
- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster



K-means

- 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 center of gravity of the data assigned to it





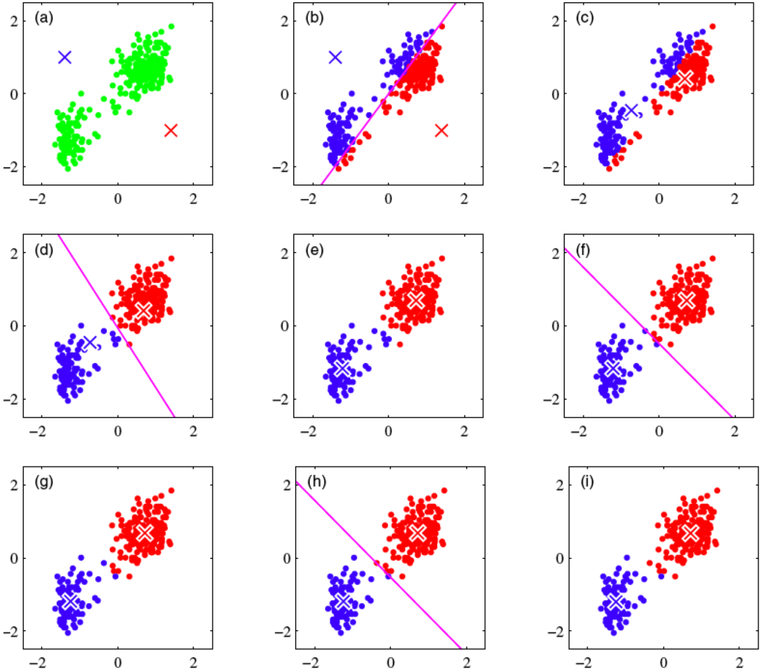


Figure from Bishop

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

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):

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: Each data point $\mathbf{x}^{(n)}$ assigned to nearest center

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

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: 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)

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

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: 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)

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

  - Refitting: 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 improves the clustering at each iteration.
  - Whenever an assignment is changed, the sum of squared distances of data points from their assigned cluster centers is reduced.

Why K-means Converges

- K-means algorithm improves the clustering at each iteration.
  - Whenever an assignment is changed, the sum of squared distances of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, the above sum is reduced.

Why K-means Converges

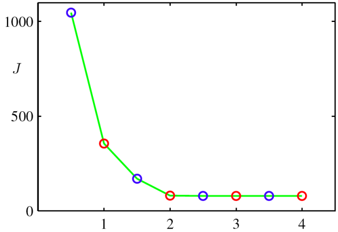
- K-means algorithm improves the clustering at each iteration.
  - Whenever an assignment is changed, the sum of squared distances of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, the above sum is reduced.
- **Test for convergence**: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).

Why K-means Converges

- K-means algorithm improves the clustering at each iteration.
  - Whenever an assignment is changed, the sum of squared distances of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, the above sum 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

Why K-means Converges

- K-means algorithm improves the clustering at each iteration.
  - Whenever an assignment is changed, the sum of squared distances of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, the above sum 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

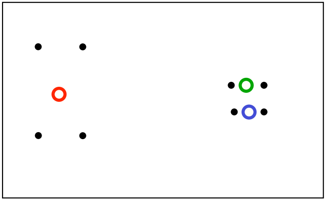


- Sum of squared distances after each assignment step (blue) and refitting step (red). The algorithm converged after the third step.

Local Minima

- This is a non-convex algorithm so it 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



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

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.

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.

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?

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

Soft K-means Algorithm

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Repeat until convergence :

Soft K-means Algorithm

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Repeat until convergence :
  - Assignment: Each data point n given soft "degree of assignment" to each cluster mean k, based on responsibilities

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

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

Soft K-means Algorithm

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Repeat until convergence :
  - 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 β?

Questions about Soft K-means

Some remaining issues

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

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'll reformulate clustering using a generative model.

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

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?

Hard cases – unequal spreads, non-circular spreads, in-between points

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

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 according to a maximum likelihood criteria.

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:

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 $p(z = k) = \pi_k$

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 $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_z, \mathbf{I})$

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 $\{1, \ldots, K\}$ 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 $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^{K}$

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

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

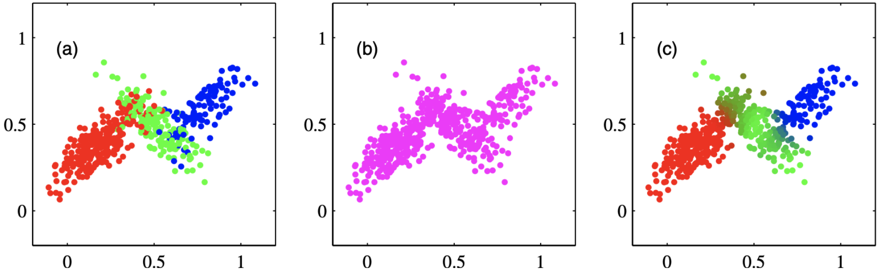
- $p(z = k|\mathbf{x})$ can be computed using Bayes rule

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

and tells us the probability $\mathbf{x}$ came from the k^{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 $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^{K}$?
- As usual, define a cost, minimize it over parameters

Maximum Likelihood with Latent Variables

- How should we choose the parameters $\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^{K}$?
- As usual, define a cost, minimize it over parameters
- We don't observe the cluster assignments z, we only see the data $\mathbf{x}$

Gaussian Mixture Model (GMM)

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

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

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

Gaussian Mixture Model (GMM)

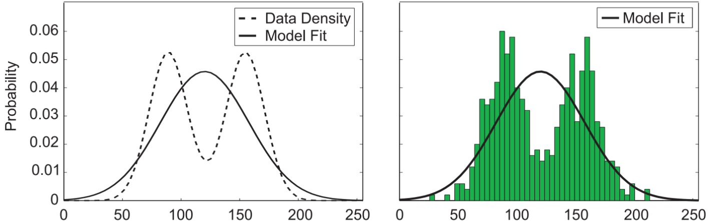
What is $p(\mathbf{x})$?

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

- 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., $p(\mathbf{x}\,|\,z=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. For this lecture, we assume $\boldsymbol{\Sigma}_k = \mathbf{I}$ 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.

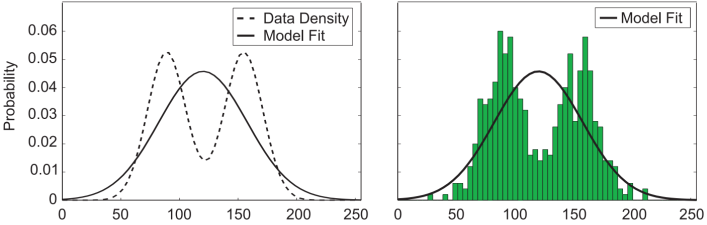
Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

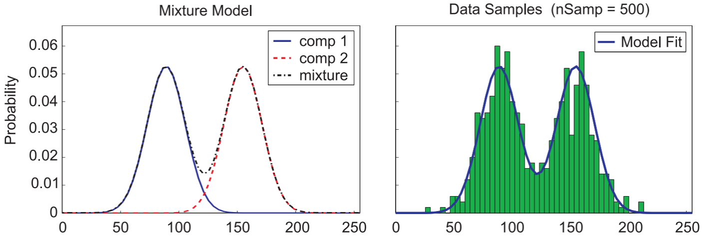


[Slide credit: K. Kutulakos]

- If you fit a Gaussian to data:

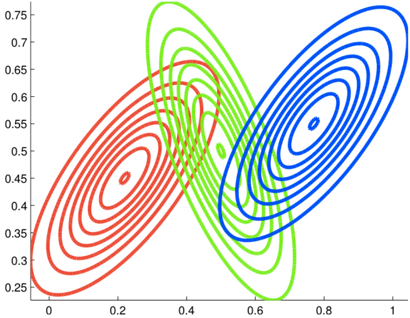


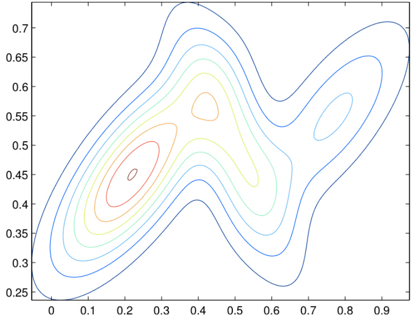
- Now, we are trying to fit a GMM (with $K = 2$ in this example):

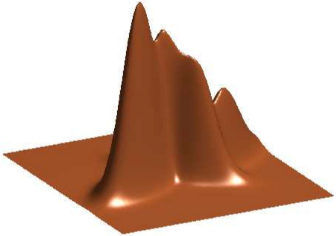


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs

Fitting a GMM is equivalent to minimizing the cost:

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

Fitting GMMs

Fitting a GMM is equivalent to minimizing the cost:

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

- 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

Fitting GMMs

Fitting a GMM is equivalent to minimizing the cost:

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

- 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?

Fitting GMMs

Fitting a GMM is equivalent to minimizing the cost:

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

- 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

- We will alternate as in the previous part.
- If we observed the cluster assignments $z^{(n)}$'s:
- By minimizing $-\log p(\mathcal{D}_{\text{complete}})$, 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 their expectations!
- Conditional expectation (using Bayes rule) of z given $\mathbf{x}$

$$\mathbb{E}[\mathbb{I}[z^{(n)} = k] | \mathbf{x}] \quad =$$

Maximum Likelihood

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

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

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 $\mathbb{E}[\mathbb{I}[z^{(n)} = k] \mid \mathbf{x}^{(n)}] = p(z^{(n)} = k|\mathbf{x}^{(n)})$.

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 $\mathbb{E}[\mathbb{I}[z^{(n)}=k] \mid \mathbf{x}^{(n)}] = p(z^{(n)}=k|\mathbf{x}^{(n)})$.
- This is still easy to optimize! Solution is similar to what we have seen:

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

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 $\mathbb{E}[\mathbb{I}[z^{(n)} = k] \mid \mathbf{x}^{(n)}] = p(z^{(n)} = k|\mathbf{x}^{(n)})$.
- 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.

How Can We Fit a Mixture of Gaussians?

- This motivates the Expectation-Maximization algorithm, which alternates between two steps:

How Can We Fit a Mixture of Gaussians?

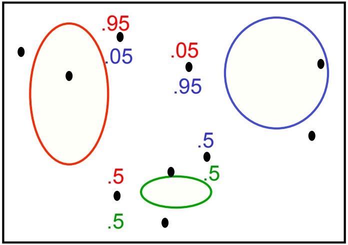
- This motivates the Expectation-Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ given our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.

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 (equivalent to the previous minimization problem).

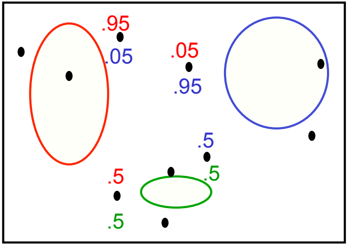
How Can We Fit a Mixture of Gaussians?

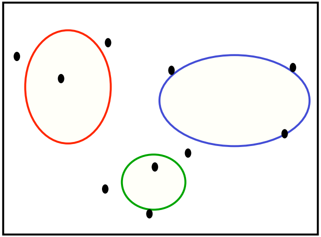
- This motivates the Expectation-Maximization algorithm, which alternates between two steps:
  1. E-step: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ given our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.
  2. M-step: 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 (equivalent to the previous minimization problem).



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 (equivalent to the previous minimization problem).





EM Algorithm for GMM

- Initialize the means $\hat{\boldsymbol{\mu}}_k$ and mixing coefficients $\hat{\pi}_k$

EM Algorithm for GMM

- Initialize the means $\hat{\boldsymbol{\mu}}_k$ and mixing coefficients $\hat{\pi}_k$
- Iterate until convergence:

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

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

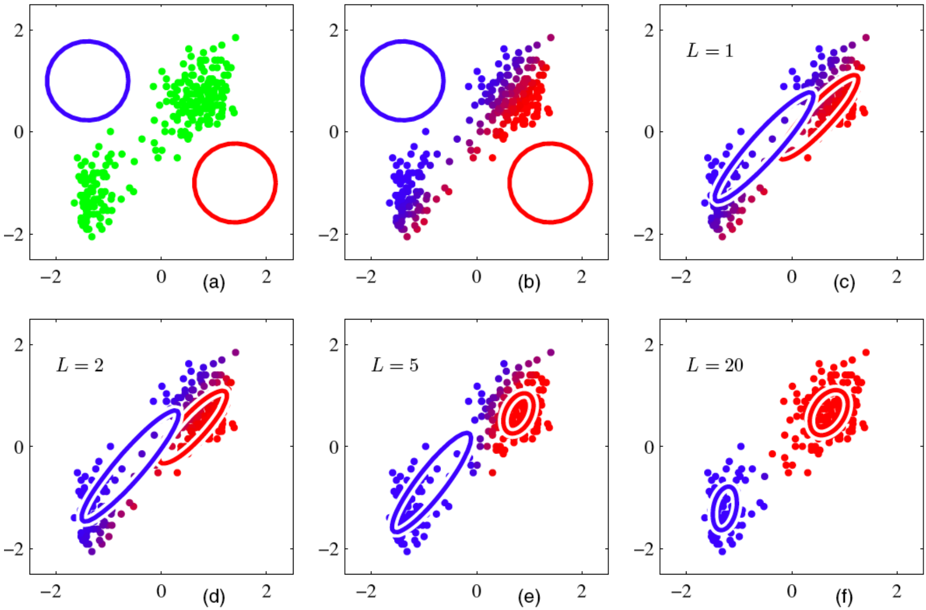
EM Algorithm for GMM

- 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

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



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

What just happened: A review

- The GMM cost $-\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$ was hard to optimize

What just happened: A review

- The GMM cost $-\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$ was hard to optimize
- We made it easy by assuming cluster assignments are known:

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

What just happened: A review

- The GMM cost $-\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$ was hard to optimize
- We made it easy by assuming cluster assignments are known:

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

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

What just happened: A review

- We ended up with the expected GMM cost 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 center to the average of the data assigned to it

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.

Further Discussion

- We assumed the covariance of each Gaussian was I to simplify the math. This assumption can be removed, allowing clusters to have different spatial extents. The resulting algorithm is still very simple.

Further Discussion

- We assumed the covariance of each Gaussian was I to simplify the math. This assumption can be removed, allowing clusters to have different spatial extents. The resulting algorithm is still very simple.
- Problem with the previous minimization:
  - 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.

GMM Recap

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

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)

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

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.

Conclusion

- We covered two clustering algorithms.
  - k-Means algorithm.
  - EM-algorithm.
- Tomorrow, we will conclude the course with a brief lecture on Fairness in ML.