Lecture 8: Introduction to Unsupervised Learning, K-means, Gaussian Mixture Model (GMM), and EM algorithm

Tao LIN

SoE, Westlake University

October 28, 2025

WESTLAKE UNIVERSITY | SCHOOL OF ENGINEERING

1. Unsupervised Learning
   - Introduction to Unsupervised Learning
   - Unsupervised Representation Learning & Generation
   - Clustering and K-means Methods
   - Gaussian Mixture Model (GMM)
   - Expectation-Maximization Algorithm

Reading materials & Reference

Reading materials:

- Chapter 10, Stanford CS 229 Lecture Notes, `https://cs229.stanford.edu/notes2022fall/main_notes.pdf`
- EPFL, CS-433 Machine Learning, `https://github.com/epfml/ML_course`

Table of Contents

1 Unsupervised Learning
- Introduction to Unsupervised Learning
- Unsupervised Representation Learning & Generation
- Clustering and K-means Methods
- Gaussian Mixture Model (GMM)
- Expectation-Maximization Algorithm

Table of Contents

1. Unsupervised Learning
   - Introduction to Unsupervised Learning
   - Unsupervised Representation Learning & Generation
   - Clustering and K-means Methods
   - Gaussian Mixture Model (GMM)
   - Expectation-Maximization Algorithm

Questions

Questions

- How can systems learn when there are no labels available?

Questions

- How can systems learn when there are no labels available?
- How to learn a meaningful internal representation for data examples?

Questions

- How can systems learn when there are no labels available?

- How to learn a meaningful internal representation for data examples?
  - I.e., to represent them in a way that reflects the semantic structure of the overall collection of input patterns?

Questions

- How can systems learn when there are no labels available?

- How to learn a meaningful internal representation for data examples?
  - I.e., to represent them in a way that reflects the semantic structure of the overall collection of input patterns?

This question is the central focus of unsupervised learning.

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

Unsupervised learning seems to play an important role in how living beings learn.

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

Unsupervised learning seems to play an important role in how living beings learn.

Variants of it seem to be more common in the brain than supervised learning.

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

Unsupervised learning seems to play an important role in how living beings learn.

Variants of it seem to be more common in the brain than supervised learning.

Two main directions in unsupervised learning are

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

Unsupervised learning seems to play an important role in how living beings learn.

Variants of it seem to be more common in the brain than supervised learning.

Two main directions in unsupervised learning are

- representation learning and

In unsupervised learning, our data consists only of features (or inputs) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, vectors in $\mathbb{R}^D$, and there are **no outputs** y_n available.

Unsupervised learning seems to play an important role in how living beings learn.

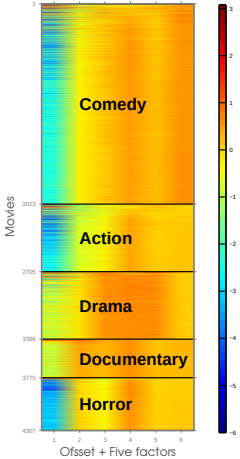
Variants of it seem to be more common in the brain than supervised learning.

Two main directions in unsupervised learning are

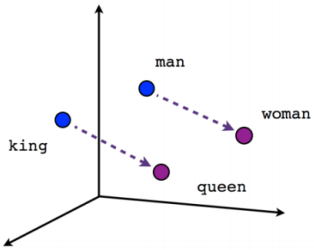
- representation learning and
- density estimation & generative models

Example

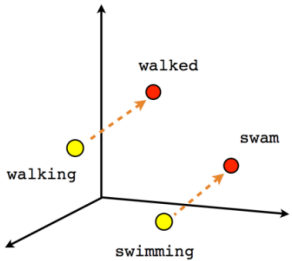
Given ratings of movies and viewers, we use matrix factorization to extract useful features (see e.g. Netflix Prize).



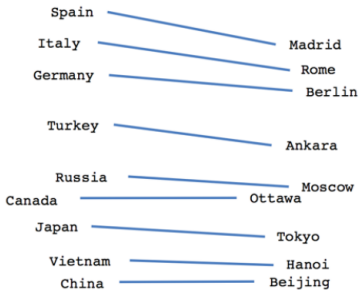
Learning word-representations using matrix-factorizations, `word2vec` (Mikolov et al. 2013).



Male-Female



Verb tense



Country-Capital

Given voting patterns of regions across Switzerland, we use PCA to extract useful features (Etter et al. 2014).

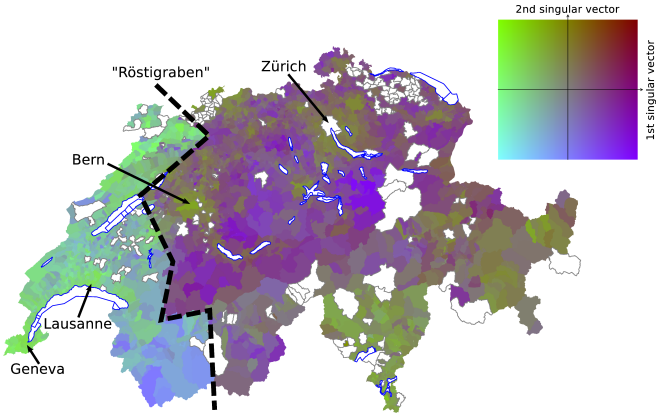
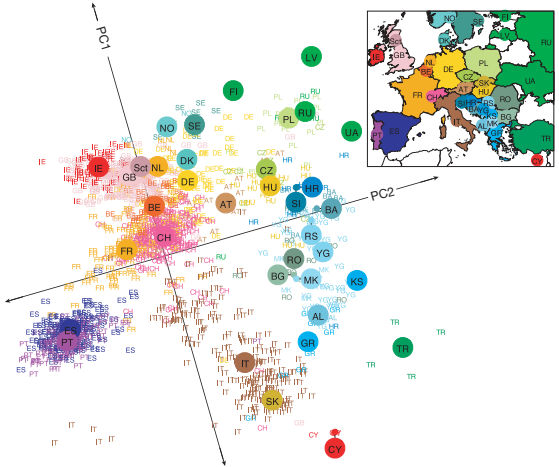


Figure 9: Voting patterns of Swiss municipalities. The color of a municipality is assigned using its location in Figure 8 and the color gradient shown in the upper right corner. Two municipalities with similar colors have similar voting patterns. The *Röstigraben*, corresponding to the cultural difference between French-speaking municipalities and German-speaking ones, is clearly visible from the difference in voting patterns. Regions shown in white are lakes or municipalities for which some vote results are missing (due to a merging of municipalities, for example). A more detailed map can be found online [2].

PCA Example 2: Genes mirror geography



Nature 2008, `http://dx.doi.org/10.1038/nature07331`

Examples for Clustering

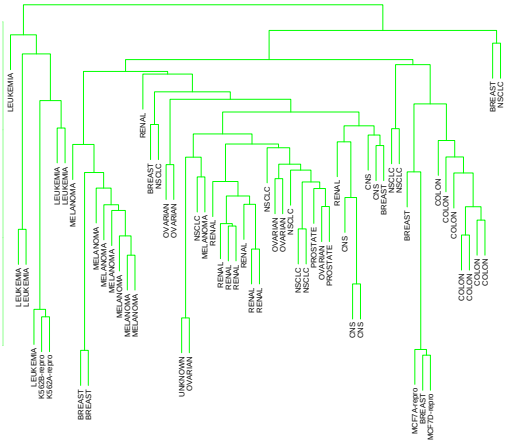


FIGURE 14.12. *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

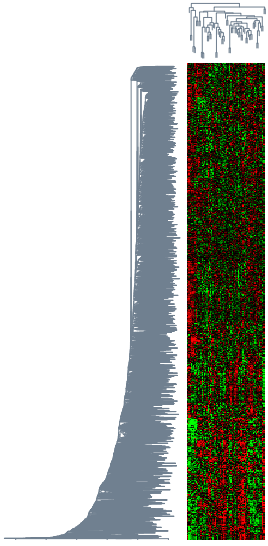
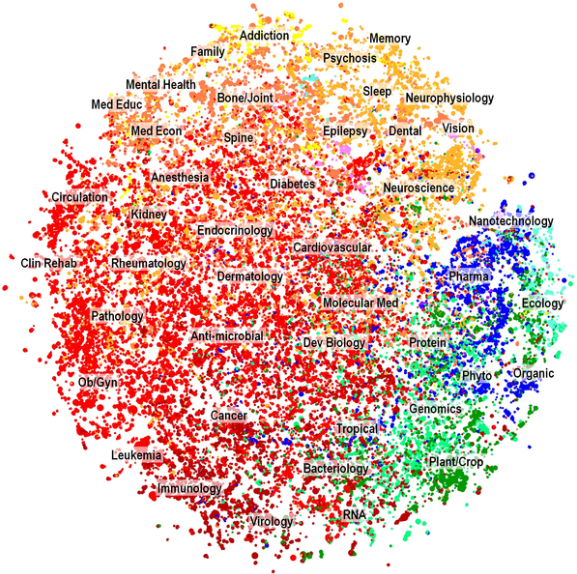


FIGURE 14.14. *DNA microarray data: average linkage hierarchical clustering has been applied independently to the rows (genes) and columns (samples), de-*

Clustering more than two million biomedical publications (Kevin Boyack et.al. 2011)



Clustering articles published in Science (Blei & Lafferty 2007)

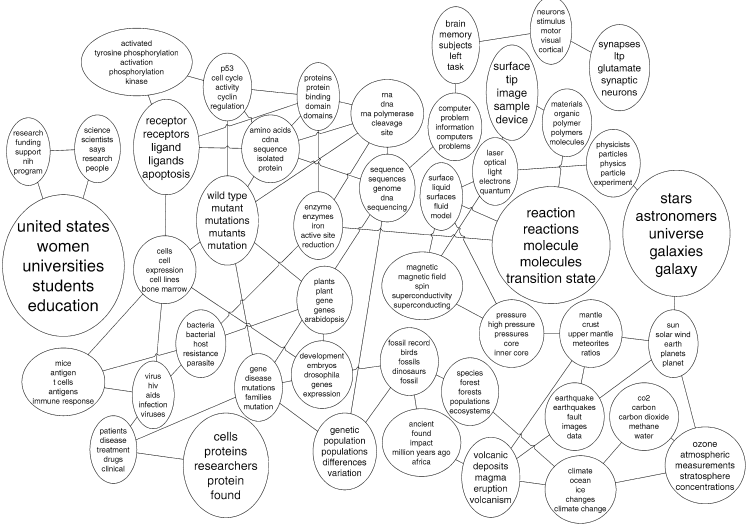


FIG. 2. *A portion of the topic graph learned from* 16,351 *OCR articles from Science* (1990–1999). *Each topic node is labeled with its five most probable phrases and has font proportional to its popularity in the corpus.* (*Phrases are found by permutation test.*) *The full model can be found in* http://www.cs.cmu.edu/~lemur/science/ *and on STATLIB.*

Table of Contents

1 Unsupervised Learning
- Introduction to Unsupervised Learning
- Unsupervised Representation Learning & Generation
- Clustering and K-means Methods
- Gaussian Mixture Model (GMM)
- Expectation-Maximization Algorithm

How does it work?

Define a unsupervised or self-supervised loss function, for

How does it work?

Define a unsupervised or self-supervised loss function, for

- Compression & Reconstruction (e.g. Auto-Encoder)

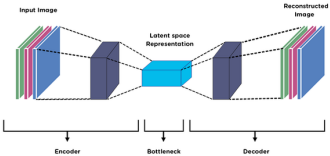


image: Deepak Birla

How does it work?

Define a unsupervised or self-supervised loss function, for

- Compression & Reconstruction (e.g. Auto-Encoder)

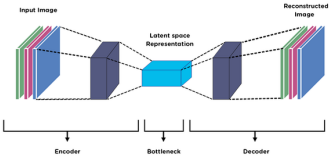


image: Deepak Birla

- Consistency & Contrastive Learning (e.g. Noise-contrastive estimation)

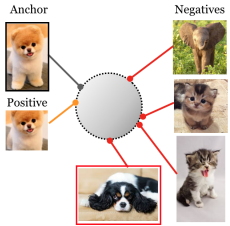


image: arxiv.org/pdf/2004.11362

Examples

(G = can be used as a generative model)

Examples

(G = can be used as a generative model)

- Generation (e.g. Auto-Encoder, Gaussian Mixture Model)

Examples

(G = can be used as a generative model)

- Generation (e.g. Auto-Encoder, Gaussian Mixture Model)
- Auto-Encoders (G)
  Invertible networks, learned compression, normalizing flows

Examples

(G = can be used as a generative model)

- Generation (e.g. Auto-Encoder, Gaussian Mixture Model)
- Auto-Encoders (G)
  Invertible networks, learned compression, normalizing flows
- Representation Learning
  e.g. images, text, time-series, video.
  Combining unsupervised representation learning (pre-training) with supervised learning

Examples

(G = can be used as a generative model)

- Generation (e.g. Auto-Encoder, Gaussian Mixture Model)
- Auto-Encoders (G)
  Invertible networks, learned compression, normalizing flows
- Representation Learning
  e.g. images, text, time-series, video.
  Combining unsupervised representation learning (pre-training) with supervised learning
- Language Models & Sequence Models (G)
  text generation, or sequence continuation, BERT video, audio & timeseries (auto-regressive, contrastive, ...)

- Generative Adversarial Networks (GAN) (G)
  see also *predictability minimization*

"A Style-Based Generator Architecture for Generative Adversarial Networks", CVPR 2019

- Contrastive image-language pretraining (CLIP)
  learns a joint representation space for images and text using contrastive learning

- Diffusion models (G)
  new state-of-the-art in image generation; these models can be adapted to generate images from text prompts (e.g., DALL-E 2, Stable Diffusion, Midjourney)

Source: Stable Diffusion model, https://stability.ai/blog/stable-diffusion-public-releas

Table of Contents

1 Unsupervised Learning
- Introduction to Unsupervised Learning
- Unsupervised Representation Learning & Generation
- Clustering and K-means Methods
- Gaussian Mixture Model (GMM)
- Expectation-Maximization Algorithm

Clustering

Clusters are groups of points whose inter-point distances are small compared to the distances outside the cluster.

Clustering

Clusters are groups of points whose inter-point distances are small compared to the distances outside the cluster.

The goal of clustering is to

Clustering

Clusters are groups of points whose inter-point distances are small compared to the distances outside the cluster.

The goal of clustering is to

- find "prototype" points $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $z_n \in \{1, 2, \ldots, K\}$ for all $n = 1, 2, \ldots, N$ data vectors $\mathbf{x}_n \in \mathbb{R}^D$ of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

K-means clustering

Assume the number of clusters K is known.

$$\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z},\boldsymbol{\mu}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \quad \text{s.t.} \quad \boldsymbol{\mu}_k \in \mathbb{R}^D,\ z_{nk} \in \{0,1\},\ \sum_{k=1}^{K} z_{nk} = 1, \tag{1}$$

$$\text{where } \mathbf{z}_n = [z_{n1}, z_{n2}, \ldots, z_{nK}]^\top \tag{2}$$

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]^\top \tag{3}$$

$$\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K]^\top \tag{4}$$

K-means clustering

Assume the number of clusters K is known.

$$\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z},\boldsymbol{\mu}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \quad \text{s.t.} \quad \boldsymbol{\mu}_k \in \mathbb{R}^D,\ z_{nk} \in \{0,1\},\ \sum_{k=1}^{K} z_{nk} = 1, \tag{1}$$

$$\text{where } \mathbf{z}_n = [z_{n1}, z_{n2}, \ldots, z_{nK}]^\top \tag{2}$$

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]^\top \tag{3}$$

$$\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K]^\top \tag{4}$$

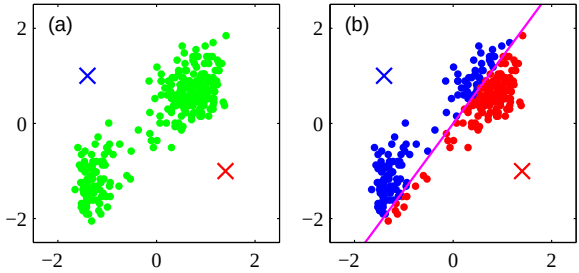
Is this optimization problem easy?

K-means Methods

Algorithm: Initialize $\boldsymbol{\mu}_k \, \forall k$, then iterate:

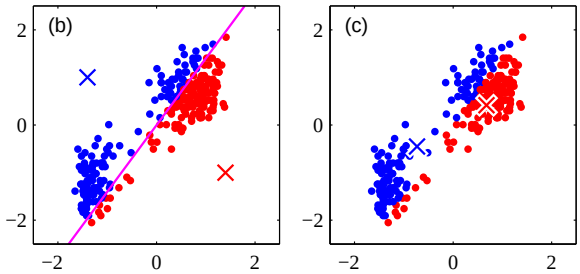
1. **Step 1:** For all n, compute $\mathbf{z}_n$ given $\boldsymbol{\mu}$.
2. **Step 2:** For all k, compute $\boldsymbol{\mu}_k$ given $\mathbf{z}$.

Step 1: For all n, compute $\mathbf{z}_n$ given $\boldsymbol{\mu}$.



$$z_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_{j=1,2,\ldots,K} \left\| \mathbf{x}_n - \boldsymbol{\mu}_j \right\|_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Step 2: For all k, compute $\boldsymbol{\mu}_k$ given $\mathbf{z}$.



Take derivative w.r.t. $\boldsymbol{\mu}_k$ to get:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} z_{nk}} \tag{6}$$

Hence, the name 'K-means'.

Summary of K-means

Initialize $\boldsymbol{\mu}_k \, \forall k$, then iterate:

1. For all n, compute $\mathbf{z}_n$ given $\boldsymbol{\mu}$.

$$z_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \left\| \mathbf{x}_n - \boldsymbol{\mu}_j \right\|_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

2. For all k, compute $\boldsymbol{\mu}_k$ given $\mathbf{z}$.

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk} \mathbf{x}_n}{\sum_{n=1}^{N} z_{nk}} \tag{8}$$

Convergence to a local optimum is assured since each step decreases the cost.

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \; \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \; \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \ \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \ \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

$\mathcal{L}$ must monotonically decrease, and the value of $\mathcal{L}$ must converge.

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \ \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \ \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

$\mathcal{L}$ must monotonically decrease, and the value of $\mathcal{L}$ must converge.

Remark:

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \ \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \ \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

$\mathcal{L}$ must monotonically decrease, and the value of $\mathcal{L}$ must converge.

Remark:

- It is possible for K-means to oscillate between a few different clustering—i.e., a few different values for $\mathbf{z}$ and/or $\boldsymbol{\mu}$—that have exactly the same value of $\mathcal{L}$.

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \ \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \ \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

$\mathcal{L}$ must monotonically decrease, and the value of $\mathcal{L}$ must converge.

Remark:

- It is possible for K-means to oscillate between a few different clustering—i.e., a few different values for $\mathbf{z}$ and/or $\boldsymbol{\mu}$—that have exactly the same value of $\mathcal{L}$.
- $\mathcal{L}$ is a convex function?

Coordinate Descent view of K-means method

K-means is a coordinate descent algorithm, where, to find $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu})$, we start with some $\boldsymbol{\mu}^{(0)}$ and repeat the following:

$$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}} \ \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}^{(t)}) \tag{9}$$

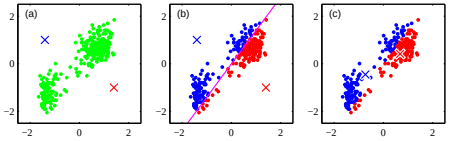
$$\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}} \ \mathcal{L}(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}) \tag{10}$$

$\mathcal{L}$ must monotonically decrease, and the value of $\mathcal{L}$ must converge.

Remark:

- It is possible for K-means to oscillate between a few different clustering—i.e., a few different values for $\mathbf{z}$ and/or $\boldsymbol{\mu}$—that have exactly the same value of $\mathcal{L}$.
- $\mathcal{L}$ is a convex function?
  NO! So coordinate descent on $\mathcal{L}$ is not guaranteed to converge to the global minimum.

K-means for the “old-faithful” dataset (Bishop’s Figure 9.1)

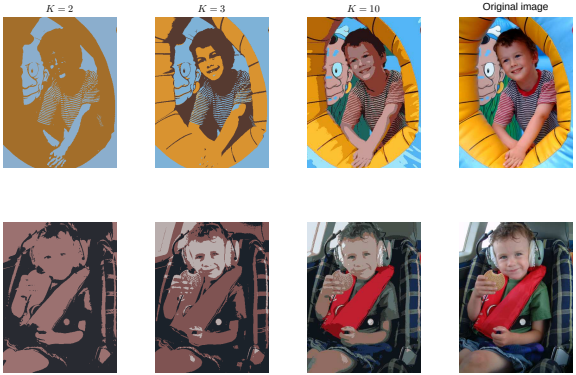


(a) Iteration 0 (b) Iteration 1 (c) Iteration 1

(d) Iteration 2 (e) Iteration 2 (f) Iteration 3

(g) Iteration 3 (h) Iteration 4 (i) Iteration 4

Data compression for images (this is also known as vector quantization).



Probabilistic model for K-means

Recall the objective

$$\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z},\boldsymbol{\mu}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \tag{11}$$

$$= \left\|\mathbf{X}^\top - \mathbf{M}\mathbf{Z}^\top\right\|_{\text{Frob}}^2 \tag{12}$$

$$\text{s.t. } \boldsymbol{\mu}_k \in \mathbb{R}^D, \tag{13}$$

$$z_{nk} \in \{0,1\}, \sum_{k=1}^{K} z_{nk} = 1. \tag{14}$$

Issues with K-means method

Issues with K-means method

1. Computation can be heavy for large N, D and K, i.e., $\mathcal{O}(NDK)$.

Issues with K-means method

1. Computation can be heavy for large N, D and K, i.e., $\mathcal{O}(NDK)$.
2. Clusters are forced to be spherical (e.g. cannot be elliptical).

Issues with K-means method

1. Computation can be heavy for large N, D and K, i.e., $\mathcal{O}(NDK)$.
2. Clusters are forced to be spherical (e.g. cannot be elliptical).
3. Each example can belong to only one cluster (“hard” cluster assignments).

Table of Contents

1 Unsupervised Learning
- Introduction to Unsupervised Learning
- Unsupervised Representation Learning & Generation
- Clustering and K-means Methods
- Gaussian Mixture Model (GMM)
- Expectation-Maximization Algorithm

Gaussian Mixture Model: Introduction

- Gaussian Mixture Models (GMMs) are a probabilistic model for density estimation

Gaussian Mixture Model: Introduction

- Gaussian Mixture Models (GMMs) are a probabilistic model for density estimation
- GMMs assume data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters

Gaussian Mixture Model: Introduction

- Gaussian Mixture Models (GMMs) are a probabilistic model for density estimation
- GMMs assume data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters
- Each data point $\mathbf{x}_n$ belongs to one of K clusters with some probability

Gaussian Mixture Model: Introduction

- Gaussian Mixture Models (GMMs) are a probabilistic model for density estimation
- GMMs assume data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters
- Each data point $\mathbf{x}_n$ belongs to one of K clusters with some probability
- The model is defined by the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, \boldsymbol{\pi}\}$
  - $\boldsymbol{\mu}_k$ - Means of each Gaussian component
  - $\boldsymbol{\Sigma}_k$ - Covariance matrices
  - $\boldsymbol{\pi}$ - Mixture weights (probabilities of each component)

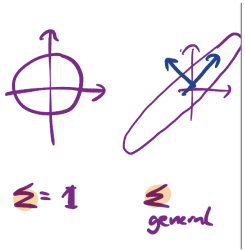
Recall the issues of K-means method:

Recall the issues of K-means method:

1. K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.

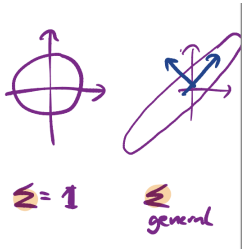
Recall the issues of K-means method:

1. K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.



Recall the issues of K-means method:

1. K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.



2. Each example in K-means can only belong to one cluster, but this may not always be a good choice, e.g. for data points that are near the “border".

Both of these problems are solved by using Gaussian Mixture Models.

Clustering with Gaussians

- Issue: K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.

Clustering with Gaussians

- Issue: K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.
- It can be resolved by using full covariance matrices $\mathbf{\Sigma}_k$ instead of *isotropic* covariances.

$$p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{\Sigma}, \mathbf{z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} [\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)]^{\mathbf{z}_{nk}}$$

Clustering with Gaussians

- Issue: K-means forces the clusters to be *spherical*, but sometimes it is desirable to have *elliptical* clusters.
- It can be resolved by using full covariance matrices $\mathbf{\Sigma}_k$ instead of *isotropic* covariances.

$$p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{\Sigma}, \mathbf{z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} [\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)]^{\mathbf{z}_{nk}}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{D \times K}$, $\mathbf{\Sigma} \in \mathbb{R}^{D \times D \times K}$, and $\boldsymbol{\pi} \in \mathbb{R}^{K}$.

Soft-clustering

- Issue: Each example in K-means can only belong to one cluster, but this may not always be a good choice, e.g. for data points that are near the “border".

Soft-clustering

- Issue: Each example in K-means can only belong to one cluster, but this may not always be a good choice, e.g. for data points that are near the “border".
- It can be resolved by defining z_n to be a random variable.

Soft-clustering

- Issue: Each example in K-means can only belong to one cluster, but this may not always be a good choice, e.g. for data points that are near the “border".
- It can be resolved by defining z_n to be a random variable.
- Specifically, define $z_n \in \{1, 2, \ldots, K\}$ that follows a multi-nomial distribution.

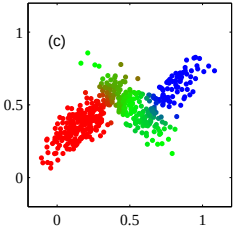
$$p(z_n = k) = \pi_k \quad \text{where} \quad \pi_k > 0\,, \forall k \quad \text{and} \quad \sum_{k=1}^{K} \pi_k = 1 \tag{15}$$

Soft-clustering

- Issue: Each example in K-means can only belong to one cluster, but this may not always be a good choice, e.g. for data points that are near the “border".
- It can be resolved by defining z_n to be a random variable.
- Specifically, define $z_n \in \{1, 2, \ldots, K\}$ that follows a multi-nomial distribution.

$$p(z_n = k) = \pi_k \quad \text{where} \quad \pi_k > 0\,, \forall k \quad \text{and} \quad \sum_{k=1}^{K} \pi_k = 1 \tag{15}$$

- This leads to soft-clustering as opposed to having “hard" assignments.



Gaussian mixture model

Together, the likelihood and the prior define the joint distribution of Gaussian mixture model (GMM):

$$\underbrace{p(\mathbf{X}, \mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}_{joint} \tag{16}$$

$$= \prod_{n=1}^{N} \underbrace{p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma})}_{likelihood} \underbrace{p(\mathbf{z}_n|\boldsymbol{\pi})}_{prior} \tag{17}$$

$$\tag{18}$$

Here,

- $\mathbf{x}_n$ are observed data vectors
- $\mathbf{z}_n$ are *latent* unobserved variables
- the unknown *parameters* are given by $\boldsymbol{\theta} := \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, \boldsymbol{\pi}\}$.

Gaussian mixture model

Together, the likelihood and the prior define the joint distribution of Gaussian mixture model (GMM):

$$\underbrace{p(\mathbf{X}, \mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})}_{joint} \tag{16}$$

$$= \prod_{n=1}^{N} \underbrace{p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma})}_{likelihood} \underbrace{p(\mathbf{z}_n|\boldsymbol{\pi})}_{prior} \tag{17}$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} [\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{\mathbf{z}_{nk}} \prod_{k=1}^{K} [\pi_k]^{\mathbf{z}_{nk}} \tag{18}$$

Here,

- $\mathbf{x}_n$ are observed data vectors
- $\mathbf{z}_n$ are *latent* unobserved variables
- the unknown *parameters* are given by $\boldsymbol{\theta} := \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, \boldsymbol{\pi}\}$.

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

- An advantage of treating $\mathbf{z}_n$ as latent variables instead of *parameters* is that we can *marginalize* them out to get a cost function that does not depend on $\mathbf{z}_n$, i.e. as if $\mathbf{z}_n$ never existed.

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

- An advantage of treating $\mathbf{z}_n$ as latent variables instead of *parameters* is that we can *marginalize* them out to get a cost function that does not depend on $\mathbf{z}_n$, i.e. as if $\mathbf{z}_n$ never existed.
- Specifically, we get the following marginal likelihood by marginalizing $\mathbf{z}_n$ out from the likelihood:

$$p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n) \tag{19}$$

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

- An advantage of treating $\mathbf{z}_n$ as latent variables instead of *parameters* is that we can *marginalize* them out to get a cost function that does not depend on $\mathbf{z}_n$, i.e. as if $\mathbf{z}_n$ never existed.
- Specifically, we get the following marginal likelihood by marginalizing $\mathbf{z}_n$ out from the likelihood:

$$p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n) \tag{19}$$

- Deriving cost functions this way is good for *statistical efficiency* (assuming $D, K \ll N$).

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

- An advantage of treating $\mathbf{z}_n$ as latent variables instead of *parameters* is that we can *marginalize* them out to get a cost function that does not depend on $\mathbf{z}_n$, i.e. as if $\mathbf{z}_n$ never existed.
- Specifically, we get the following marginal likelihood by marginalizing $\mathbf{z}_n$ out from the likelihood:

$$p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n) \tag{19}$$

- Deriving cost functions this way is good for *statistical efficiency* (assuming $D, K \ll N$).
  - Without a latent variable model, the number of parameters grows at rate $\mathcal{O}(N)$.

Marginal likelihood

GMM is a latent variable model with $\mathbf{z}_n$ being the unobserved (latent) variables.

- An advantage of treating $\mathbf{z}_n$ as latent variables instead of *parameters* is that we can *marginalize* them out to get a cost function that does not depend on $\mathbf{z}_n$, i.e. as if $\mathbf{z}_n$ never existed.
- Specifically, we get the following marginal likelihood by marginalizing $\mathbf{z}_n$ out from the likelihood:

$$p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{where} \quad p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n) \tag{19}$$

- Deriving cost functions this way is good for *statistical efficiency* (assuming $D, K \ll N$).
  - Without a latent variable model, the number of parameters grows at rate $\mathcal{O}(N)$.
  - After marginalization, the growth is reduced to $\mathcal{O}(D^2K)$.

Maximum likelihood

To get a maximum (marginal) likelihood estimate of $\boldsymbol{\theta}$, we maximize the following:

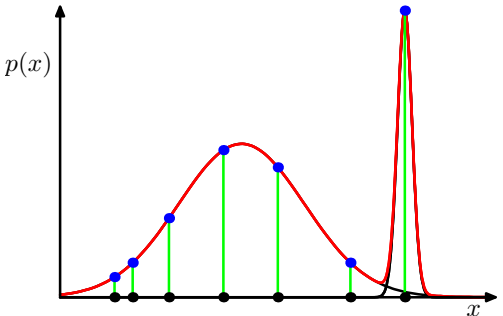
$$\max_{\boldsymbol{\theta}} \underbrace{\sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\log p(\mathbf{x}|\boldsymbol{\theta})}$$

Maximum likelihood

To get a maximum (marginal) likelihood estimate of $\boldsymbol{\theta}$, we maximize the following:

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

Is this cost convex? Identifiable? Bounded?



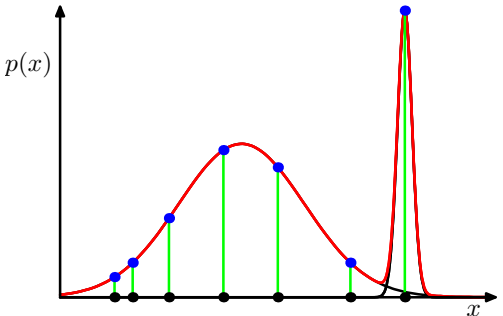
Maximum likelihood

To get a maximum (marginal) likelihood estimate of $\boldsymbol{\theta}$, we maximize the following:

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

Is this cost convex? Identifiable? Bounded?

Exercise:



Maximum likelihood

To get a maximum (marginal) likelihood estimate of $\boldsymbol{\theta}$, we maximize the following:

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

Is this cost convex? Identifiable? Bounded?
Exercise: non-convex, non-unique optima, unbounded.

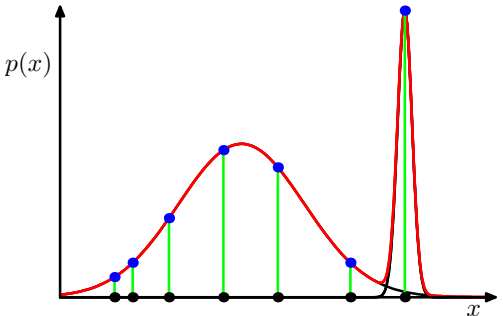


Table of Contents

1 Unsupervised Learning
- Introduction to Unsupervised Learning
- Unsupervised Representation Learning & Generation
- Clustering and K-means Methods
- Gaussian Mixture Model (GMM)
- Expectation-Maximization Algorithm

Motivation of Expectation-Maximization (EM)

Recall the equation of maximum likelihood for GMM:

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{20}$$

Motivation of Expectation-Maximization (EM)

Recall the equation of maximum likelihood for GMM:

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{20}$$

Remarks:

Motivation of Expectation-Maximization (EM)

Recall the equation of maximum likelihood for GMM:

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{20}$$

Remarks:

- Computing maximum likelihood for GMM is difficult, due to .

Motivation of Expectation-Maximization (EM)

Recall the equation of maximum likelihood for GMM:

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{20}$$

Remarks:

- Computing maximum likelihood for GMM is difficult, due to the $\log$ outside the sum.

EM provides an elegant solution by iteratively building & optimizing lower bounds.

Motivation of Expectation-Maximization (EM)

Recall the equation of maximum likelihood for GMM:

$$\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{20}$$

Remarks:

- Computing maximum likelihood for GMM is difficult, due to the $\log$ outside the sum.

EM provides an elegant solution by iteratively building & optimizing lower bounds.

- It uses an iterative two-step procedure, where individual steps usually involve problems that are easy to optimize.

An Overview of EM

- The EM algorithm provides an elegant approach to latent variable estimation problems.
- EM iteratively tries to overcome the challenge of not knowing the hidden variables $\mathbf{z}_n$.

An Overview of EM

- The EM algorithm provides an elegant approach to latent variable estimation problems.
- EM iteratively tries to overcome the challenge of not knowing the hidden variables $\mathbf{z}_n$.
- If we knew the values of the latent variables $\mathbf{z}_n$, maximum likelihood estimation would be simple.

An Overview of EM

- The EM algorithm provides an elegant approach to latent variable estimation problems.
- EM iteratively tries to overcome the challenge of not knowing the hidden variables $\mathbf{z}_n$.
- If we knew the values of the latent variables $\mathbf{z}_n$, maximum likelihood estimation would be simple.
- Since we don't know them, we:

An Overview of EM

- The EM algorithm provides an elegant approach to latent variable estimation problems.
- EM iteratively tries to overcome the challenge of not knowing the hidden variables $\mathbf{z}_n$.
- If we knew the values of the latent variables $\mathbf{z}_n$, maximum likelihood estimation would be simple.
- Since we don't know them, we:
  - E-step: Compute the expected values of the latent variables given current parameters

An Overview of EM

- The EM algorithm provides an elegant approach to latent variable estimation problems.
- EM iteratively tries to overcome the challenge of not knowing the hidden variables $\mathbf{z}_n$.
- If we knew the values of the latent variables $\mathbf{z}_n$, maximum likelihood estimation would be simple.
- Since we don't know them, we:
  - E-step: Compute the expected values of the latent variables given current parameters
  - M-step: Update parameters as if these expected values were the true values

EM algorithm: Summary

Start with $\boldsymbol{\theta}^{(1)}$ and iterate:

EM algorithm: Summary

Start with $\boldsymbol{\theta}^{(1)}$ and iterate:

1. Expectation step: Compute a lower bound to the cost such that it is tight at the previous $\boldsymbol{\theta}^{(t)}$:

$$\mathcal{L}(\boldsymbol{\theta}) \geq \underline{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) \text{ and} \qquad \text{(lower bound on } \mathcal{L}\text{)} \tag{21}$$

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \underline{\mathcal{L}}(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)})\,. \tag{22}$$

EM algorithm: Summary

Start with $\boldsymbol{\theta}^{(1)}$ and iterate:

1. Expectation step: Compute a lower bound to the cost such that it is tight at the previous $\boldsymbol{\theta}^{(t)}$:

$$\mathcal{L}(\boldsymbol{\theta}) \geq \underline{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) \text{ and} \qquad \text{(lower bound on } \mathcal{L}\text{)} \tag{21}$$

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \underline{\mathcal{L}}(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)})\,. \tag{22}$$

2. Maximization step: Update $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \underline{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})\,. \tag{23}$$

EM algorithm: Summary

Start with $\boldsymbol{\theta}^{(1)}$ and iterate:

1. Expectation step: Compute a lower bound to the cost such that it is tight at the previous $\boldsymbol{\theta}^{(t)}$:

$$\mathcal{L}(\boldsymbol{\theta}) \geq \underline{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) \text{ and} \qquad \text{(lower bound on } \mathcal{L}\text{)} \tag{21}$$

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \underline{\mathcal{L}}(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)})\,. \tag{22}$$

2. Maximization step: Update $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \underline{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})\,. \tag{23}$$

Each iteration of EM is guaranteed to increase (or keep the same) the log-likelihood.

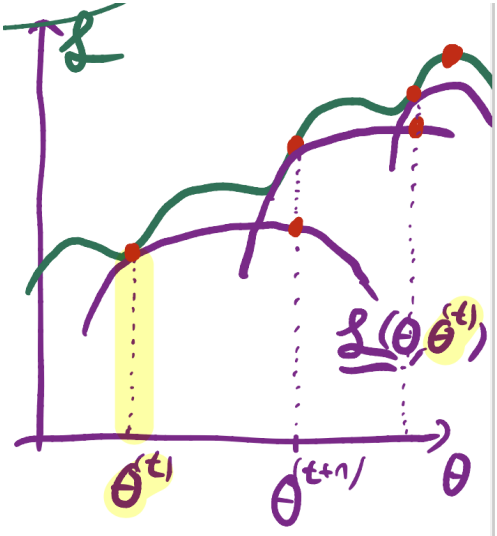


Figure: Visualization of EM iterations: Each bound is tight at the previous parameter value

The Expectation step (E-step)

Exercise (the concavity of log):

The Expectation step (E-step)

Exercise (the concavity of log):
Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$:

$$\log\left(\sum_{k=1}^{K} q_k r_k\right) \geq \sum_{k=1}^{K} q_k \log r_k \tag{24}$$

The Expectation step (E-step)

Exercise (the concavity of log):
Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$:

$$\log\left(\sum_{k=1}^{K} q_k r_k\right) \geq \sum_{k=1}^{K} q_k \log r_k \tag{24}$$

The Expectation step (E-step):

$$\log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \geq \sum_{k=1}^{K} q_{kn} \log \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{q_{kn}} \tag{25}$$

The Expectation step (E-step)

Exercise (the concavity of log):
Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$:

$$\log\left(\sum_{k=1}^{K} q_k r_k\right) \geq \sum_{k=1}^{K} q_k \log r_k \tag{24}$$

The Expectation step (E-step):

$$\log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \geq \sum_{k=1}^{K} q_{kn} \log \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{q_{kn}} \tag{25}$$

with equality when,

$$q_{kn} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}\,. \tag{26}$$

The Expectation step (E-step)

Exercise (the concavity of log):
Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$:

$$\log\left(\sum_{k=1}^{K} q_k r_k\right) \geq \sum_{k=1}^{K} q_k \log r_k \tag{24}$$

The Expectation step (E-step):

$$\log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \geq \sum_{k=1}^{K} q_{kn} \log \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{q_{kn}} \tag{25}$$

with equality when,

$$q_{kn} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}\,. \tag{26}$$

This is precisely the posterior probability $p(\mathbf{z}_n = k | \mathbf{x}_n, \boldsymbol{\theta})$.

The Expectation step (E-step)

Exercise (the concavity of log):
Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$:

$$\log\left(\sum_{k=1}^{K} q_k r_k\right) \geq \sum_{k=1}^{K} q_k \log r_k \tag{24}$$

The Expectation step (E-step):

$$\log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \geq \sum_{k=1}^{K} q_{kn} \log \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{q_{kn}} \tag{25}$$

with equality when,

$$q_{kn} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}\,. \tag{26}$$

This is precisely the posterior probability $p(\mathbf{z}_n = k | \mathbf{x}_n, \boldsymbol{\theta})$. Exercise: Show why the equality holds.

The Evidence Lower Bound (ELBO)

- The procedure we just derived can be formalized as maximizing the Evidence Lower Bound (ELBO)

The Evidence Lower Bound (ELBO)

- The procedure we just derived can be formalized as maximizing the Evidence Lower Bound (ELBO)
- The log-likelihood (evidence) can be decomposed as:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{27}$$

where $\mathcal{L}(q, \boldsymbol{\theta})$ is the ELBO:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{28}$$

The Evidence Lower Bound (ELBO)

- The procedure we just derived can be formalized as maximizing the Evidence Lower Bound (ELBO)
- The log-likelihood (evidence) can be decomposed as:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{27}$$

  where $\mathcal{L}(q, \boldsymbol{\theta})$ is the ELBO:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{28}$$

- Since KL-divergence is always non-negative, $\mathcal{L}(q, \boldsymbol{\theta}) \leq \log p(\mathbf{x}|\boldsymbol{\theta})$

The Evidence Lower Bound (ELBO)

- The procedure we just derived can be formalized as maximizing the Evidence Lower Bound (ELBO)
- The log-likelihood (evidence) can be decomposed as:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{27}$$

  where $\mathcal{L}(q, \boldsymbol{\theta})$ is the ELBO:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{28}$$

- Since KL-divergence is always non-negative, $\mathcal{L}(q, \boldsymbol{\theta}) \leq \log p(\mathbf{x}|\boldsymbol{\theta})$
- ELBO is tight when $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ (the posterior distribution)

The Evidence Lower Bound (ELBO)

- The procedure we just derived can be formalized as maximizing the Evidence Lower Bound (ELBO)
- The log-likelihood (evidence) can be decomposed as:

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{27}$$

  where $\mathcal{L}(q, \boldsymbol{\theta})$ is the ELBO:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{28}$$

- Since KL-divergence is always non-negative, $\mathcal{L}(q, \boldsymbol{\theta}) \leq \log p(\mathbf{x}|\boldsymbol{\theta})$
- ELBO is tight when $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ (the posterior distribution)
- EM alternates between:
  - E-step: Set $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{(t)})$ to make ELBO tight
  - M-step: Maximize ELBO with respect to $\boldsymbol{\theta}$, keeping q fixed

ELBO: Intuition and Properties

- The ELBO can be rewritten in two equivalent forms:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{29}$$

$$= \log p(\mathbf{x}|\boldsymbol{\theta}) - \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{30}$$

ELBO: Intuition and Properties

- The ELBO can be rewritten in two equivalent forms:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{29}$$

$$= \log p(\mathbf{x}|\boldsymbol{\theta}) - \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{30}$$

- First form shows ELBO as:
  - Expected complete-data log-likelihood (energy)
  - Minus entropy of the approximating distribution

ELBO: Intuition and Properties

- The ELBO can be rewritten in two equivalent forms:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{29}$$

$$= \log p(\mathbf{x}|\boldsymbol{\theta}) - \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{30}$$

- First form shows ELBO as:
  - Expected complete-data log-likelihood (energy)
  - Minus entropy of the approximating distribution
- Second form shows ELBO as:
  - Log-likelihood (what we want to maximize)
  - Minus KL-divergence between approximate and true posterior

ELBO: Intuition and Properties

- The ELBO can be rewritten in two equivalent forms:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \tag{29}$$

$$= \log p(\mathbf{x}|\boldsymbol{\theta}) - \mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \tag{30}$$

- First form shows ELBO as:
  - Expected complete-data log-likelihood (energy)
  - Minus entropy of the approximating distribution
- Second form shows ELBO as:
  - Log-likelihood (what we want to maximize)
  - Minus KL-divergence between approximate and true posterior
- ELBO maximization balances:
  - Finding a good fit to the data through $\boldsymbol{\theta}$
  - Finding a good approximation to the posterior through q

The Maximization step (M-step)

Maximizing the lower bound w.r.t. $\boldsymbol{\theta}$ results in

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} q_{kn}^{(t)} \left[\log \pi_k + \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \quad \text{(exercise. why?)} \tag{31}$$

The Maximization step (M-step)

Maximizing the lower bound w.r.t. $\boldsymbol{\theta}$ results in

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} q_{kn}^{(t)} \left[\log \pi_k + \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \quad \text{(exercise. why?)} \tag{31}$$

Differentiating w.r.t. $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^{-1}$, we can get the updates for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$.

$$\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_{n=1}^{N} q_{kn}^{(t)} \mathbf{x}_n}{\sum_{n=1}^{N} q_{kn}^{(t)}} \tag{32}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_{n=1}^{N} q_{kn}^{(t)} (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{n=1}^{N} q_{kn}^{(t)}} \tag{33}$$

The Maximization step (M-step)

Maximizing the lower bound w.r.t. $\boldsymbol{\theta}$ results in

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} q_{kn}^{(t)} \left[\log \pi_k + \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \qquad \text{(exercise. why?)} \tag{31}$$

Differentiating w.r.t. $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^{-1}$, we can get the updates for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$.

$$\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_{n=1}^{N} q_{kn}^{(t)} \mathbf{x}_n}{\sum_{n=1}^{N} q_{kn}^{(t)}} \tag{32}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_{n=1}^{N} q_{kn}^{(t)} (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{n=1}^{N} q_{kn}^{(t)}} \tag{33}$$

For π_k, we use the fact that they sum to 1. Therefore, we add a Lagrangian term, differentiate w.r.t. π_k and set to 0, to get the following update:

$$\pi_k^{(t+1)} := \frac{1}{N} \sum_{n=1}^{N} q_{kn}^{(t)} \tag{34}$$

Summary of EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

Summary of EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

1. E-step: Compute assignments $q_{kn}^{(t)}$ (posterior probabilities):

$$q_{kn}^{(t)} := \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{k=1}^{K} \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})} \tag{35}$$

Summary of EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

1. E-step: Compute assignments $q_{kn}^{(t)}$ (posterior probabilities):

$$q_{kn}^{(t)} := \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{k=1}^{K} \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})} \tag{35}$$

2. Compute the marginal likelihood (cost).

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \tag{36}$$

Summary of EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

1. E-step: Compute assignments $q_{kn}^{(t)}$ (posterior probabilities):

$$q_{kn}^{(t)} := \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})} \tag{35}$$

2. Compute the marginal likelihood (cost).

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \tag{36}$$

3. M-step: Update $\boldsymbol{\mu}_k^{(t+1)}, \boldsymbol{\Sigma}_k^{(t+1)}, \pi_k^{(t+1)}$.

$$\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_{n=1}^N q_{kn}^{(t)} \mathbf{x}_n}{\sum_{n=1}^N q_{kn}^{(t)}} \tag{37}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_{n=1}^N q_{kn}^{(t)} (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{n=1}^N q_{kn}^{(t)}} \tag{38}$$

$$\pi_k^{(t+1)} := \frac{1}{N} \sum_{n=1}^N q_{kn}^{(t)} \tag{39}$$

Summary of EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

1. E-step: Compute assignments $q_{kn}^{(t)}$ (posterior probabilities):

$$q_{kn}^{(t)} := \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})} \tag{35}$$

2. Compute the marginal likelihood (cost).

$$\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \tag{36}$$

3. M-step: Update $\boldsymbol{\mu}_k^{(t+1)}, \boldsymbol{\Sigma}_k^{(t+1)}, \pi_k^{(t+1)}$.

$$\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_{n=1}^N q_{kn}^{(t)} \mathbf{x}_n}{\sum_{n=1}^N q_{kn}^{(t)}} \tag{37}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_{n=1}^N q_{kn}^{(t)} (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{n=1}^N q_{kn}^{(t)}} \tag{38}$$

$$\pi_k^{(t+1)} := \frac{1}{N} \sum_{n=1}^N q_{kn}^{(t)} \tag{39}$$

If we let the covariance be diagonal i.e. $\boldsymbol{\Sigma}_k := \sigma^2 \mathbf{I}$, EM algorithm is same as K-means as $\sigma^2 \to 0$.

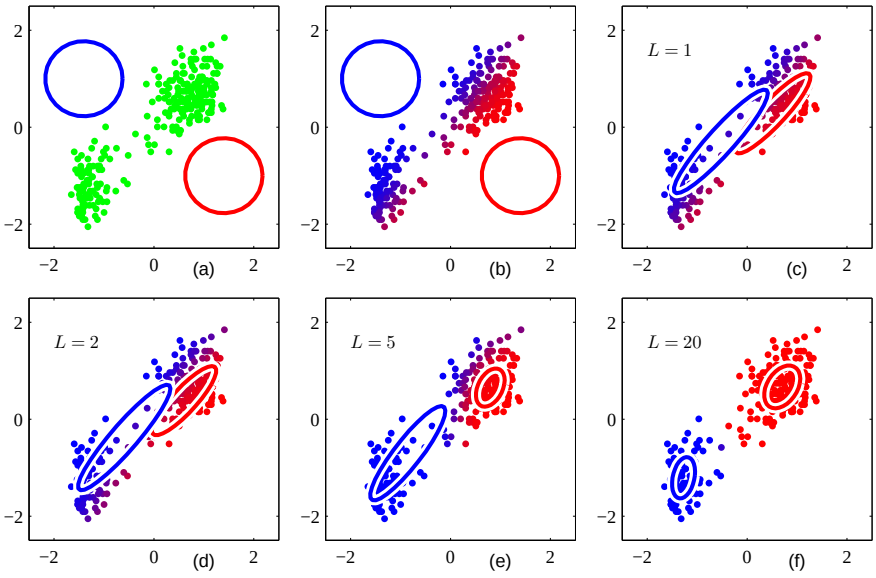
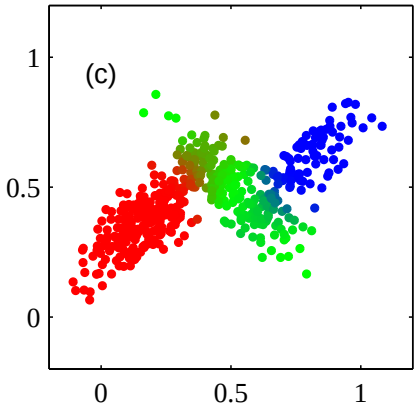


Figure: EM algorithm for GMM: Visualization of the iterative fitting process

Posterior distribution

We now show that $q_{kn}^{(t)}$ is the posterior distribution of the latent variable, i.e. $q_{kn}^{(t)} = p(\mathbf{z}_n = k \mid \mathbf{x}_n, \boldsymbol{\theta}^{(t)})$

$$\underbrace{p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta})}_{\text{joint}} = \underbrace{p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\theta})}_{\text{likelihood}} \underbrace{p(\mathbf{z}_n|\boldsymbol{\theta})}_{\text{prior}} = \underbrace{p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta})}_{\text{posterior}} \underbrace{p(\mathbf{x}_n|\boldsymbol{\theta})}_{\text{marginal likelihood}} \tag{40}$$



The relationship to K-means

- The EM algorithm for GMMs is reminiscent of the K-means clustering algorithm:

The relationship to K-means

- The EM algorithm for GMMs is reminiscent of the K-means clustering algorithm:
- Instead of the "hard" cluster assignments in K-means, EM uses “soft” assignments q_{kn}.

$$q_{kn} = p(z_n = k \mid \mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{41}$$

The relationship to K-means

- The EM algorithm for GMMs is reminiscent of the K-means clustering algorithm:
- Instead of the "hard" cluster assignments in K-means, EM uses “soft” assignments q_{kn}.

$$q_{kn} = p(z_n = k \mid \mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{41}$$

- "Soft" means our assignments are probabilities taking values in $[0, 1]$

The relationship to K-means

- The EM algorithm for GMMs is reminiscent of the K-means clustering algorithm:
- Instead of the "hard" cluster assignments in K-means, EM uses “soft” assignments q_{kn}.

$$q_{kn} = p(z_n = k \mid \mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{41}$$

- "Soft" means our assignments are probabilities taking values in $[0, 1]$
- "Hard" assignments take values in $\{0, 1\}$ or $\{1, \ldots, K\}$

The relationship to K-means

- The EM algorithm for GMMs is reminiscent of the K-means clustering algorithm:
- Instead of the "hard" cluster assignments in K-means, EM uses “soft” assignments q_{kn}.

$$q_{kn} = p(z_n = k \mid \mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{41}$$

- "Soft" means our assignments are probabilities taking values in $[0, 1]$
- "Hard" assignments take values in $\{0, 1\}$ or $\{1, \ldots, K\}$
- Like K-means, EM is also susceptible to local optima, so initializing at several different parameter values may be a good strategy.

EM Convergence Properties

- The EM algorithm is guaranteed to increase the likelihood at each iteration:

$$\mathcal{L}(\boldsymbol{\theta}^{(t+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{42}$$

EM Convergence Properties

- The EM algorithm is guaranteed to increase the likelihood at each iteration:

$$\mathcal{L}(\boldsymbol{\theta}^{(t+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{42}$$

- This guarantees convergence to a local maximum of the likelihood function.

EM Convergence Properties

- The EM algorithm is guaranteed to increase the likelihood at each iteration:

$$\mathcal{L}(\boldsymbol{\theta}^{(t+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{42}$$

- This guarantees convergence to a local maximum of the likelihood function.
- However, like K-means, it may not find the global maximum.

EM Convergence Properties

- The EM algorithm is guaranteed to increase the likelihood at each iteration:

$$\mathcal{L}(\boldsymbol{\theta}^{(t+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{42}$$

- This guarantees convergence to a local maximum of the likelihood function.
- However, like K-means, it may not find the global maximum.
- The rate of convergence depends on the fraction of information about the parameters contained in the latent variables.

EM Convergence Properties

- The EM algorithm is guaranteed to increase the likelihood at each iteration:

$$\mathcal{L}(\boldsymbol{\theta}^{(t+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{42}$$

- This guarantees convergence to a local maximum of the likelihood function.
- However, like K-means, it may not find the global maximum.
- The rate of convergence depends on the fraction of information about the parameters contained in the latent variables.
- In practice, we monitor the change in log-likelihood and stop when it falls below a threshold.

EM and Variational Inference

- The ELBO perspective reveals that EM is a special case of variational inference

EM and Variational Inference

- The ELBO perspective reveals that EM is a special case of variational inference
- Variational inference approximates intractable posterior distributions with a tractable distribution $q(\mathbf{z})$

EM and Variational Inference

- The ELBO perspective reveals that EM is a special case of variational inference
- Variational inference approximates intractable posterior distributions with a tractable distribution $q(\mathbf{z})$
- In standard EM:
  - We set $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{(t)})$ exactly in the E-step
  - This is possible because the posterior is tractable for models like GMM

EM and Variational Inference

- The ELBO perspective reveals that EM is a special case of variational inference
- Variational inference approximates intractable posterior distributions with a tractable distribution $q(\mathbf{z})$
- In standard EM:
  - We set $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{(t)})$ exactly in the E-step
  - This is possible because the posterior is tractable for models like GMM
- In variational EM:
  - The posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ is intractable
  - We restrict $q(\mathbf{z})$ to a simpler family of distributions
  - E-step: Find $q(\mathbf{z})$ that maximizes ELBO (minimizes KL-divergence)
  - M-step: Same as standard EM (maximize ELBO w.r.t. $\boldsymbol{\theta}$)

EM and Variational Inference

- The ELBO perspective reveals that EM is a special case of variational inference
- Variational inference approximates intractable posterior distributions with a tractable distribution $q(\mathbf{z})$
- In standard EM:
  - We set $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{(t)})$ exactly in the E-step
  - This is possible because the posterior is tractable for models like GMM
- In variational EM:
  - The posterior $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ is intractable
  - We restrict $q(\mathbf{z})$ to a simpler family of distributions
  - E-step: Find $q(\mathbf{z})$ that maximizes ELBO (minimizes KL-divergence)
  - M-step: Same as standard EM (maximize ELBO w.r.t. $\boldsymbol{\theta}$)
- This connection has led to powerful techniques like variational autoencoders (VAEs)

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:
  - Missing data problems

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:
  - Missing data problems
  - Hidden Markov Models

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:
  - Missing data problems
  - Hidden Markov Models
  - Mixture of experts models

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:
  - Missing data problems
  - Hidden Markov Models
  - Mixture of experts models
  - Bayesian networks with hidden variables

Applications of EM algorithm

- The EM algorithm can be applied to many problems beyond GMMs:
  - Missing data problems
  - Hidden Markov Models
  - Mixture of experts models
  - Bayesian networks with hidden variables
- The same principles apply: identify latent variables, compute their expected values, and maximize parameters.

EM for Gaussian Mixture Models: Derivation

- The log-likelihood for GMM with unknown latent variables $\mathbf{z}_n$ is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{43}$$

EM for Gaussian Mixture Models: Derivation

- The log-likelihood for GMM with unknown latent variables $\mathbf{z}_n$ is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{43}$$

- This is difficult to maximize directly due to the log of a sum

EM for Gaussian Mixture Models: Derivation

- The log-likelihood for GMM with unknown latent variables $\mathbf{z}_n$ is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{43}$$

- This is difficult to maximize directly due to the log of a sum
- If we knew the values of $\mathbf{z}_n$ (which component generated each data point), the problem would be simple

EM for Gaussian Mixture Models: Derivation

- The log-likelihood for GMM with unknown latent variables $\mathbf{z}_n$ is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{43}$$

- This is difficult to maximize directly due to the log of a sum
- If we knew the values of $\mathbf{z}_n$ (which component generated each data point), the problem would be simple
- Instead of solving directly, we compute the expected values of $\mathbf{z}_n$ given current parameters, then maximize the expected complete-data log-likelihood

EM for Gaussian Mixture Models: Derivation

- The log-likelihood for GMM with unknown latent variables $\mathbf{z}_n$ is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{43}$$

- This is difficult to maximize directly due to the log of a sum
- If we knew the values of $\mathbf{z}_n$ (which component generated each data point), the problem would be simple
- Instead of solving directly, we compute the expected values of $\mathbf{z}_n$ given current parameters, then maximize the expected complete-data log-likelihood

This approach gives us the two-step iterative EM algorithm

Practical Considerations for GMM Implementation

- **Initialization strategies:**

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**
  - Adding small values to diagonal of covariance matrices

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**
  - Adding small values to diagonal of covariance matrices
  - Performing computations in log space to avoid underflow

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**
  - Adding small values to diagonal of covariance matrices
  - Performing computations in log space to avoid underflow
- **Model selection:**

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**
  - Adding small values to diagonal of covariance matrices
  - Performing computations in log space to avoid underflow
- **Model selection:**
  - Determining the optimal number of components K

Practical Considerations for GMM Implementation

- **Initialization strategies:**
  - Random initialization of means and covariances
  - K-means initialization (use K-means clusters to initialize GMM parameters)
  - Multiple restarts to avoid local optima
- **Numerical stability:**
  - Adding small values to diagonal of covariance matrices
  - Performing computations in log space to avoid underflow
- **Model selection:**
  - Determining the optimal number of components K
  - Using criteria like BIC (Bayesian Information Criterion) or AIC (Akaike Information Criterion)