Important Dates

- **Homework 6**: Due by end of tomorrow 10th April
- **Homework 7**: 4 Jupyter Notebooks will be released today/tomorrow. One of them will be due 19th Apr, and others on 26th April.
  - 18-661: Final score will consider the best 3 out of 4
  - 18-461: Final score will consider the best 2 out of 4
- **No class on April 11** due to Spring Carnival
- **No recitation on April 12** due to Spring Carnival
- **Final exam**
  - May 2nd, 6 pm to 9 pm
  - Please let us know asap if you have conflicting exams or need special accommodations
1. Review: Clustering and $k$-means

2. Gaussian mixture models

3. EM Algorithm
Review: Clustering and $k$-means
Supervised Learning: labeled observations \((x_1, y_1), \ldots, (x_n, y_n)\)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
Supervised Learning: labeled observations \( \{(x_1, y_1), \ldots (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)
**Supervised Learning**: labeled observations \( \{(x_1, y_1), \ldots (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)
Supervised versus Unsupervised Learning

**Supervised Learning:** labeled observations \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

**Unsupervised Learning:** unlabeled observations \( \{x_1, \ldots, x_n\} \)

- Learning algorithm must find latent structure from features alone
Supervised Learning: labeled observations \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations \( \{x_1, \ldots, x_n\} \)

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
**Supervised versus Unsupervised Learning**

**Supervised Learning:** labeled observations \{ (x_1, y_1), \ldots, (x_n, y_n) \}  
- Labels ‘teach’ algorithm to learn mapping from observations to labels  
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

**Unsupervised Learning:** unlabeled observations \{ x_1, \ldots, x_n \}  
- Learning algorithm must find latent structure from features alone  
- Can be goal in itself (discover hidden patterns, exploratory analysis)  
- Can be means to an end (pre-processing for supervised task)
Supervised versus Unsupervised Learning

**Supervised Learning:** labeled observations \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

**Unsupervised Learning:** unlabeled observations \( \{x_1, \ldots, x_n\} \)

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
Supervised versus Unsupervised Learning

**Supervised Learning:** labeled observations \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

**Unsupervised Learning:** unlabeled observations \( \{x_1, \ldots, x_n\} \)

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
  - Clustering (today)
Supervised Learning: labeled observations \{ (x_1, y_1), \ldots, (x_n, y_n) \}

- Labels ‘teach’ algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations \{ x_1, \ldots, x_n \}

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
  - Clustering (today)
  - Dimensionality Reduction: Transform an initial feature representation into a more concise representation
Setup Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example
Cluster data into two clusters.

Example Applications
- Identify communities within social networks
- Find topic groups in news stories
- Group similar sequences into gene families
Clustering

Setup Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters

Toy Example Cluster data into two clusters.

![Toy Example Graph](image)
Clustering

Setup Given $\mathcal{D} = \{x_n\}_{n=1}^{N}$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^{K}$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

![Toy Example Diagram]
Clustering

Setup Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

Example Applications
- Identify communities within social networks
- Find topic groups in news stories
- Group similar sequences into gene families
Clustering

**Setup** Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

**Toy Example** Cluster data into two clusters.

![Toy Example Diagram](image-url)

**Example Applications**

- Identify communities within social networks
Clustering

Setup Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

![Toy Example Cluster data into two clusters.](image)

Example Applications

- Identify communities within social networks
- Find topic groups in news stories
Clustering

Setup Given $D = \{x_n\}_{n=1}^N$ and $K$, we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

Example Applications

- Identify communities within social networks
- Find topic groups in news stories
- Group similar sequences into gene families
**k-means**

*k*-means: an iterative clustering method

High-level idea:

- **Initialize**: Pick \( k \) random points as cluster centers, \( \{\mu_1, \ldots, \mu_k\} \)
- **Alternate**:
  1. Assign data points to closest cluster center in \( \{\mu_1, \ldots, \mu_k\} \)
  2. Change each cluster center to the average of its assigned points
- **Stop**: When the clusters are stable
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
$k$-means example (several iterations)
*k*-means example (several iterations)
**Intuition**: Data points assigned to cluster $k$ should be near prototype $\mu_k$.
**Intuition**: Data points assigned to cluster $k$ should be near prototype $\mu_k$

**Distortion measure**: (clustering objective function, cost function)

\[
J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|^2
\]

where $r_{nk} \in \{0, 1\}$ is an indicator variable

\[
r_{nk} = 1 \quad \text{if and only if} \quad A(x_n) = k
\]
**Intuition:** Data points assigned to cluster $k$ should be near prototype $\mu_k$

**Distortion measure:** (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(x_n) = k$$

**Notes:**

- Distance measure: $||x_n - \mu_k||^2$ calculates how far $x_n$ is from the cluster center $\mu_k$
**Intuition:** Data points assigned to cluster $k$ should be near prototype $\mu_k$.

**Distortion measure:** (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \text{ if and only if } A(x_n) = k$$

**Notes:**

- Distance measure: $\| x_n - \mu_k \|^2$ calculates how far $x_n$ is from the cluster center $\mu_k$
- Canonical example is the 2-norm, i.e., $\| \cdot \|_2^2$, but could be something else!
Algorithm

Minimize distortion Alternative optimization between \( \{r_{nk}\} \) and \( \{\mu_k\} \)

- **Step 0** Initialize \( \{\mu_k\} \) to some values
Minimize distortion Alternative optimization between \( \{r_{nk}\} \) and \( \{\mu_k\} \)

- **Step 0** Initialize \( \{\mu_k\} \) to some values
- **Step 1** Fix \( \{\mu_k\} \) and minimize over \( \{r_{nk}\} \), to get this assignment:
  
  \[
  r_{nk} = \begin{cases} 
  1 & \text{if } k = \text{argmin}_j \|x_n - \mu_j\|^2 \\
  0 & \text{otherwise}
  \end{cases}
  \]
Minimize distortion Alternative optimization between \( \{ r_{nk} \} \) and \( \{ \mu_k \} \)

- **Step 0** Initialize \( \{ \mu_k \} \) to some values
- **Step 1** Fix \( \{ \mu_k \} \) and minimize over \( \{ r_{nk} \} \), to get this assignment:
  \[
  r_{nk} = \begin{cases} 
  1 & \text{if } k = \arg\min_j ||x_n - \mu_j||^2 \\
  0 & \text{otherwise}
  \end{cases}
  \]
- **Step 2** Fix \( \{ r_{nk} \} \) and minimize over \( \{ \mu_k \} \) to get this update:
  \[
  \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}
  \]
- **Step 3** Return to Step 1 unless stopping criterion is met
Properties of $k$-means algorithm

Does it converge?

- Guaranteed to converge in a finite number of iterations.
- Key idea: $k$-means is an alternating optimization approach.
- Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge.
- However, may converge to a local minimum (objective is non-convex).

What's the runtime?

- Running time per iteration:
  - Assign data points to closest cluster: $O(ndk)$
  - Re-compute cluster centers: $O(nd)$
- Thus, total runtime is $O(n dik)$, where $i$ is the number of iterations.
Properties of $k$-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: $k$-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)
Properties of \( k \)-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: \( k \)-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What’s the runtime?

- Running time per iteration:
  - Assume: \( n \) data points, each with \( d \) features, and \( k \) clusters
  - Assign data points to closest cluster: \( O(ndk) \)
  - Re-compute cluster centers: \( O(nd) \)

Thus, total runtime is \( O(ndki) \), where \( i \) is the number of iterations.
Properties of $k$-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: $k$-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What’s the runtime?

- **Running time per iteration:**
  - Assume: $n$ data points, each with $d$ features, and $k$ clusters
  - Assign data points to closest cluster: $O(ndk)$
  - Re-compute cluster centers: $O(nd)$

- **Thus, total runtime is:** $O(ndki)$, where $i$ is the number of iterations
Practical Issues with $k$-means

- How to select $k$?
  - Prior knowledge
  - Heuristics (e.g., elbow method)
- How to select distance measure?
  - Often requires some knowledge of problem
  - Some examples: Euclidean distance (for images), Hamming distance (distance between two strings), shared key words (for websites)
- How to initialize cluster centers?
  - The final clustering can depend significantly on the initial points you pick!
Elbow method

Key idea: select a small value of $k$ that minimizes within-cluster distances
How to get $k$-means to work on this data?

Should look at the distance of the data points from the origin:

$$\sqrt{x^2 + y^2}$$
How to get $k$-means to work on this data?

Should look at the distance of the data points from the origin $\sqrt{x_n^2 + y_n^2}$
Changing features (distance measure) can help

If the cluster $i$ mean is $(\mu_{i,x}, \mu_{i,y})$, the distance of $(x_n, y_n)$ from it can be defined as

$$|\sqrt{\mu_{i,x}^2 + \mu_{i,y}^2} - \sqrt{x_n^2 + y_n^2}|$$
**Key idea**: Run $k$-means, but with a better initialization

- Choose center $\mu_1$ at random
- For $j = 2, \ldots, k$
  - Choose $\mu_j$ among $x_1, \ldots, x_n$ with probability:
  - $P(\mu_j = x_i) \propto \min_{j' < j} ||x_i - \mu_{j'}||^2$

**Initialization helps to get good coverage of the space**

**Theorem**: $k$-means++ always obtains a $O(\log k)$ approximation to the optimal solution in expectation.

Running $k$-means after this initialization can only improve on the result
$k$-means++

N=200, K=5
K-means with random initialization

N=200, K=5
K-means++
Connection to $k$-Nearest Neighbors

- Nearest Neighbors is a **supervised** learning method
  - Each training point $x_n$ has a corresponding given label $y_n$
  - Objective: Assign label to a new $x$ by looking at the labels of its $k$ nearest points

- Clustering is an **unsupervised** learning method
  - We are given training points $x_n$ without labels
  - Objective: Divide them into $k$ groups to understand patterns in the data
Clustering can make Nearest Neighbors more efficient

- A drawback of nearest neighbors is that we have to remember the training data
- Clustering can help compress the training data into a small number of representative points

**Algorithm to Improve Nearest Neighbors**

- For all training data points $x_n$ with label $y_n = c$, for $C$ classes $c = 1, \ldots, C$, cluster the $x_n$ into $R$ groups.
- Store these $R$ cluster means for each of the $C$ classes
- For a test data point $x$, find the $k$ nearest neighbors among the $RC$ cluster means and assign their majority label to $x$
1. Review: Clustering and $k$-means

2. Gaussian mixture models

3. EM Algorithm
Gaussian mixture models
Data points are assigned *deterministically* to one (and only one) cluster.
One more potential issue with \emph{k-means} . . .

Data points are assigned \emph{deterministically} to one (and only one) cluster.

In reality, clusters may overlap, and it may be better to identify the \emph{probability} that a point belongs to each cluster.
Probabilistic interpretation of clustering?

How can we model $p(x)$ to reflect our intuition that points stay close to their cluster centers?
Probabilistic interpretation of clustering?

How can we model $p(x)$ to reflect our intuition that points stay close to their cluster centers?

- Points seem to form 3 clusters

![Diagram showing points forming 3 clusters](image)
Probabilistic interpretation of clustering?

How can we model $p(x)$ to reflect our intuition that points stay close to their cluster centers?

- Points seem to form 3 clusters
- We cannot model $p(x)$ with simple and known distributions
Probabilistic interpretation of clustering?

How can we model $p(x)$ to reflect our intuition that points stay close to their cluster centers?

- Points seem to form 3 clusters
- We cannot model $p(x)$ with simple and known distributions
- E.g., the data is not a Gaussian b/c we have 3 distinct concentrated regions
Gaussian mixture models: intuition

- **Key idea:** Model each region with a distinct distribution

![Graph showing distinct regions with different distributions](image)

- Must learn from unlabeled data $D = \{x_n\}_{n=1}^N$
Gaussian mixture models: intuition

- **Key idea:** Model each region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
Gaussian mixture models: intuition

- **Key idea:** Model each region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
Gaussian mixture models: intuition

- **Key idea:** Model each region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- *However*, we don’t know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!
Key idea: Model each region with a distinct distribution.

Can use Gaussians — Gaussian mixture models (GMMs).

However, we don’t know cluster assignments (label), parameters of Gaussians, or mixture components!

Must learn from unlabeled data $\mathcal{D} = \{x_n\}_{n=1}^N$. 
Recall: Gaussian (Normal) distributions
GMM has the following density function for $x$

$$p(x) = \sum_{k=1}^{K} \omega_k N(x | \mu_k, \Sigma_k)$$

- $K$: number of Gaussians — they are called mixture components
- $\mu_k$ and $\Sigma_k$: mean and covariance matrix of $k$-th component
GMM has the following density function for $x$

$$p(x) = \sum_{k=1}^{K} \omega_k N(x | \mu_k, \Sigma_k)$$

- $K$: number of Gaussians — they are called mixture components
- $\mu_k$ and $\Sigma_k$: mean and covariance matrix of $k$-th component
- $\omega_k$: mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:
GMM has the following density function for $x$

$$p(x) = \sum_{k=1}^{K} \omega_k N(x | \mu_k, \Sigma_k)$$

- $K$: number of Gaussians — they are called mixture components
- $\mu_k$ and $\Sigma_k$: mean and covariance matrix of $k$-th component
- $\omega_k$: mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

$$\forall k, \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

These properties ensure $p(x)$ is in fact a probability density function
GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

\[ p(x, z) = p(z)p(x|z) \]

where \( z \) is a discrete random variable taking values between 1 and \( K \).
Consider the following joint distribution

\[ p(x, z) = p(z)p(x|z) \]

where \( z \) is a discrete random variable taking values between 1 and \( K \).

Denote

\[ \omega_k = p(z = k) \]

Now, assume the conditional distributions are Gaussian distributions

\[ p(x|z = k) = \mathcal{N}(x|\mu_k, \Sigma_k) \]
Consider the following joint distribution

\[ p(x, z) = p(z)p(x|z) \]

where \( z \) is a discrete random variable taking values between 1 and \( K \).

Denote

\[ \omega_k = p(z = k) \]

Now, assume the conditional distributions are Gaussian distributions

\[ p(x|z = k) = \mathcal{N}(x|\mu_k, \Sigma_k) \]

Then, the marginal distribution of \( x \) is

\[
p(x) = \sum_{k=1}^{K} \omega_k \mathcal{N}(x|\mu_k, \Sigma_k)
\]

Namely, the Gaussian mixture model
Gaussian mixtures in 1D

Mixture of 1D Gaussians

- Component 1
- Component 2
- Mixture
Gaussian mixture model for clustering
The conditional distribution between $x$ and $z$ (representing color) are

\[
p(x|z = red) = N(x|\mu_1, \Sigma_1) \\
p(x|z = blue) = N(x|\mu_2, \Sigma_2) \\
p(x|z = green) = N(x|\mu_3, \Sigma_3)
\]
The conditional distribution between $x$ and $z$ (representing color) are

$$p(x|z = red) = \mathcal{N}(x|\mu_1, \Sigma_1)$$

$$p(x|z = blue) = \mathcal{N}(x|\mu_2, \Sigma_2)$$

$$p(x|z = green) = \mathcal{N}(x|\mu_3, \Sigma_3)$$

The marginal distribution is thus

$$p(x) = p(red)\mathcal{N}(x|\mu_1, \Sigma_1) + p(blue)\mathcal{N}(x|\mu_2, \Sigma_2) + p(green)\mathcal{N}(x|\mu_3, \Sigma_3)$$
The parameters in GMMs are
Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$

Let’s first consider the simple/unrealistic case where we have labels $z$

Define $\mathcal{D}' = \{x_n, z_n\}_{n=1}^N$, $\mathcal{D} = \{x_n\}_{n=1}^N$

- $\mathcal{D}'$ is the **complete** data
- $\mathcal{D}$ the **incomplete** data

How can we learn our parameters?
The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$

Let’s first consider the simple/unrealistic case where we have labels $z$

Define $D' = \{x_n, z_n\}_{n=1}^N$, $D = \{x_n\}_{n=1}^N$

- $D'$ is the complete data
- $D$ the incomplete data

How can we learn our parameters?

Given $D'$, the maximum likelihood estimation of the $\theta$ is given by

$$\theta = \arg \max \log D' = \sum_n \log p(x_n, z_n)$$
Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(x_n, z_n) = \sum_n \log p(z_n)p(x_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(x_n|z_n)$$

where we have grouped data by cluster labels $z_n$. 

Note: in the complete setting the $r_{nk}$ are binary, but later we will 'relax' these variables and allow them to take on fractional values.
The complete likelihood is decomposable

\[ \sum_n \log p(x_n, z_n) = \sum_n \log p(z_n) p(x_n | z_n) = \sum_k \sum_{n: z_n = k} \log p(z_n) p(x_n | z_n) \]

where we have grouped data by cluster labels \( z_n \).

Let \( r_{nk} \in \{0, 1\} \) be a binary variable that indicates whether \( z_n = k \):
The complete likelihood is decomposable

\[
\sum_n \log p(x_n, z_n) = \sum_n \log p(z_n) p(x_n | z_n) = \sum_k \sum_{n: z_n = k} \log p(z_n) p(x_n | z_n)
\]

where we have grouped data by cluster labels \( z_n \).

Let \( r_{nk} \in \{0, 1\} \) be a binary variable that indicates whether \( z_n = k \):

\[
\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \log p(z = k) p(x_n | z = k)
\]
Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(x_n, z_n) = \sum_n \log p(z_n) p(x_n | z_n) = \sum_k \sum_{n: z_n = k} \log p(z_n) p(x_n | z_n)$$

where we have grouped data by cluster labels $z_n$.

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \log p(z = k) p(x_n | z = k)$$

$$= \sum_k \sum_n r_{nk} [\log \omega_k + \log N(x_n | \mu_k, \Sigma_k)]$$

Note: in the complete setting the $r_{nk}$ are binary, but later we will ‘relax’ these variables and allow them to take on fractional values.
Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \left[ \log \omega_k + \log N(x_n | \mu_k, \Sigma_k) \right]$$
From our previous discussion, we have

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \left[ \log \omega_k + \log N(x_n | \mu_k, \Sigma_k) \right]$$

Regrouping, we have

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \log \omega_k + \sum_k \left\{ \sum_n r_{nk} \log N(x_n | \mu_k, \Sigma_k) \right\}$$
Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$
\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \left[ \log \omega_k + \log N(x_n | \mu_k, \Sigma_k) \right]
$$

Regrouping, we have

$$
\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \log \omega_k + \sum_k \left\{ \sum_n r_{nk} \log N(x_n | \mu_k, \Sigma_k) \right\}
$$

The term inside the braces depends on $k$-th component’s parameters. It is now easy to show that (left as an exercise) the MLE is:

$$
\omega_k = \frac{\sum_n r_{nk}}{\sum_k \sum_n r_{nk}}, \quad \mu_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} x_n
$$

$$
\Sigma_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (x_n - \mu_k)(x_n - \mu_k)^\top
$$

What’s the intuition?
Intuition

Since $r_{nk}$ is binary, the previous solution is nothing but:

- $\omega_k$: fraction of total data points whose cluster label $z_n$ is $k$
  - note that $\sum_k \sum_n r_{nk} = N$
- $\mu_k$: mean of all data points whose $z_n$ is $k$
- $\Sigma_k$: covariance of all data points whose $z_n$ is $k$
Since $r_{nk}$ is binary, the previous solution is nothing but:

- $\omega_k$: fraction of total data points whose cluster label $z_n$ is $k$
  - note that $\sum_k \sum_n r_{nk} = N$
- $\mu_k$: mean of all data points whose $z_n$ is $k$
- $\Sigma_k$: covariance of all data points whose $z_n$ is $k$

Recall that this depends on us knowing the true cluster labels $z_n$

This intuition will help us develop an algorithm for estimating $\theta$ when we *do not* know $z_n$ (incomplete data)
Parameter estimation for GMMs: Incomplete data

GMM Parameters

\[ \theta = \{ \omega_k, \mu_k, \Sigma_k \}_{k=1}^K \]

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete

- Observed: \( D = \{ x_n \} \)
- Unobserved (hidden): \( \{ z_n \} \)
Parameter estimation for GMMs: Incomplete data

**GMM Parameters**

\[ \theta = \{ \omega_k, \mu_k, \Sigma_k \}_{k=1}^K \]

**Incomplete Data**

Our data contains observed and unobserved data, and hence is incomplete

- Observed: \( D = \{ x_n \} \)
- Unobserved (hidden): \( \{ z_n \} \)

**Goal** Obtain the maximum likelihood estimate of \( \theta \):

\[ \theta = \arg \max \ell(\theta) = \arg \max \log D = \arg \max \sum_n \log p(x_n | \theta) \]
Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete

- Observed: $$D = \{x_n\}$$
- Unobserved (hidden): $$\{z_n\}$$

Goal

Obtain the maximum likelihood estimate of $$\theta$$:

$$\theta = \arg \max \ell(\theta) = \arg \max \log D = \arg \max \sum_n \log p(x_n|\theta)$$

$$= \arg \max \sum_n \log \sum_{z_n} p(x_n, z_n|\theta)$$

The objective function $$\ell(\theta)$$ is called the **incomplete** log-likelihood.
When \( z_n \) is not given, we can guess it via the **posterior probability** (recall: Bayes’ rule!)

\[
p(z_n = k | x_n) = \frac{p(x_n | z_n = k)p(z_n = k)}{p(x_n)} = \frac{p(x_n | z_n = k)p(z_n = k)}{\sum_{k' = 1}^{K} p(x_n | z_n = k')p(z_n = k')}
\]
When $z_n$ is not given, we can guess it via the **posterior probability** (recall: Bayes’ rule!)

$$p(z_n = k|x_n) = \frac{p(x_n|z_n = k)p(z_n = k)}{p(x_n)} = \frac{p(x_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(x_n|z_n = k')p(z_n = k')}$$

To compute the posterior probability, we need to know the parameters $\theta$!

**Idea**: Let’s pretend we know the value of the parameters so we can compute the posterior probability.

How is that going to help us?
We define $r_{nk} = p(z_n = k | x_n)$.
We define $r_{nk} = p(z_n = k | x_n)$

- Recall that $r_{nk}$ was previously binary
- Now it’s a “soft” assignment of $x_n$ to $k$-th component
- Each $x_n$ is assigned to a component fractionally according to $p(z_n = k | x_n)$
We define \( r_{nk} = p(z_n = k|x_n) \)

- Recall that \( r_{nk} \) was previously binary
- Now it’s a “soft” assignment of \( x_n \) to \( k \)-th component
- Each \( x_n \) is assigned to a component fractionally according to 
  \( p(z_n = k|x_n) \)

We now get the same expression for the MLE as before!

\[
\omega_k = \frac{\sum_n r_{nk}}{\sum_k \sum_n r_{nk}}, \quad \mu_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} x_n
\]

\[
\Sigma_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (x_n - \mu_k)(x_n - \mu_k)^\top
\]

But remember, we’re ‘cheating’ by using \( \theta \) to compute \( r_{nk} \)!
Iterative procedure

Alternate between estimating $r_{nk}$ and computing parameters

- Step 0: initialize $\theta$ with some values (random or otherwise)
- Step 1: compute $r_{nk}$ using the current $\theta$
- Step 2: update $\theta$ using the just computed $r_{nk}$
- Step 3: go back to Step 1
Iterative procedure

Alternate between estimating $r_{nk}$ and computing parameters

- Step 0: initialize $\theta$ with some values (random or otherwise)
- Step 1: compute $r_{nk}$ using the current $\theta$
- Step 2: update $\theta$ using the just computed $r_{nk}$
- Step 3: go back to Step 1

This is an example of the EM algorithm — a powerful procedure for model estimation with hidden/latent variables
Iterative procedure

Alternate between estimating \( r_{nk} \) and computing parameters

- Step 0: initialize \( \theta \) with some values (random or otherwise)
- Step 1: compute \( r_{nk} \) using the current \( \theta \)
- Step 2: update \( \theta \) using the just computed \( r_{nk} \)
- Step 3: go back to Step 1

This is an example of the **EM algorithm** — a powerful procedure for model estimation with hidden/latent variables

Connection with \( K \)-means?
Iterative procedure

Alternate between estimating $r_{nk}$ and computing parameters

- Step 0: initialize $\theta$ with some values (random or otherwise)
- Step 1: compute $r_{nk}$ using the current $\theta$
- Step 2: update $\theta$ using the just computed $r_{nk}$
- Step 3: go back to Step 1

This is an example of the **EM algorithm** — a powerful procedure for model estimation with hidden/latent variables

Connection with $K$-means?

- GMMs provide probabilistic interpretation for $K$-means
- $K$-means is “hard” GMM or GMMs is “soft” $K$-means
- Posterior $r_{nk}$ provides a probabilistic assignment for $x_n$ to cluster $k$
GMMs vs. $k$-means

![Gaussian Mixture vs. KMeans](image1.png)

![Gaussian Mixture vs. KMeans](image2.png)
GMMs vs. $k$-means

Pros/Cons

- $k$-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering.
GMMs vs. $k$-means

Pros/Cons

- $k$-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute
GMMs vs. \textit{k}-means

### Pros/Cons

- \textit{k}-means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute
- Both methods have a similar set of practical issues (having to select \( k \), the distance, and the initialization)
1. Review: Clustering and $k$-means

2. Gaussian mixture models

3. EM Algorithm
EM Algorithm
EM algorithm: motivation and setup

• EM is a general procedure to estimate parameters for probabilistic models with hidden/latent variables

• Suppose the model is given by a joint distribution

\[ p(x|\theta) = \sum_z p(x, z|\theta) \]
EM algorithm: motivation and setup

- EM is a general procedure to estimate parameters for probabilistic models with hidden/latent variables.
- Suppose the model is given by a joint distribution

\[ p(x|\theta) = \sum_z p(x, z|\theta) \]

- Given **incomplete data** \( D = \{x_n\} \) our goal is to compute MLE of \( \theta \):

\[ \theta = \arg \max \ell(\theta) = \arg \max \log D = \arg \max \sum_n \log p(x_n|\theta) \]

\[ = \arg \max \sum_n \log \sum_{z_n} p(x_n, z_n|\theta) \]

The objective function \( \ell(\theta) \) is called **incomplete** log-likelihood.
A lower bound

- log-sum form of incomplete log-likelihood is difficult to work with
- EM: construct lower bound on $\ell(\theta)$ (E-step) and optimize it (M-step)
A lower bound

- log-sum form of incomplete log-likelihood is difficult to work with

- EM: construct lower bound on $\ell(\theta)$ (E-step) and optimize it (M-step)

- If we define $q(z)$ as a distribution over $z$, then

$$
\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta)
$$
A lower bound

- Log-sum form of incomplete log-likelihood is difficult to work with
- EM: construct lower bound on $\ell(\theta)$ (E-step) and optimize it (M-step)
- If we define $q(z)$ as a distribution over $z$, then

$$
\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta) \\
= \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n | \theta)}{q(z_n)}
$$
A lower bound

- log-sum form of incomplete log-likelihood is difficult to work with
- EM: construct lower bound on \( \ell(\theta) \) (E-step) and optimize it (M-step)
- If we define \( q(z) \) as a distribution over \( z \), then

\[
\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n|\theta)
= \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n|\theta)}{q(z_n)}
\geq \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n|\theta)}{q(z_n)}
\]

- Last step follows from Jensen's inequality, i.e., \( f(\mathbb{E}X) \geq \mathbb{E}f(X) \) for concave function \( f \)
• Consider the previous model where $x$ could be from 3 regions
• We can choose $q(z)$ as any valid distribution
• e.g., $q(z = k) = 1/3$ for any of 3 colors
• e.g., $q(z = k) = 1/2$ for red and blue, 0 for green

Which $q(z)$ should we choose?
Which \( q(z) \) to choose?

Recall:

\[
\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta) = \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n | \theta)}{q(z_n)}
\]

\[
\geq \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta)}{q(z_n)}
\]

- The lower bound we derived for \( \ell(\theta) \) holds for all choices of \( q(\cdot) \)
- We want a \emph{tight} lower bound
Which $q(z)$ to choose?

Recall:

$$\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta) = \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n | \theta)}{q(z_n)}$$

$$\geq \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta)}{q(z_n)}$$

- The lower bound we derived for $\ell(\theta)$ holds for all choices of $q(\cdot)$
- We want a *tight* lower bound, and given some current estimate $\theta^t$, we will pick $q(\cdot)$ such that our lower bound holds with equality at $\theta^t$
- Choose $q(z_n) \propto p(x_n, z_n | \theta^t)$!
Which $q(z)$ to choose?

Recall:

$$\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta) = \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n | \theta)}{q(z_n)}$$

$$\geq \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta)}{q(z_n)}$$

- The lower bound we derived for $\ell(\theta)$ holds for all choices of $q(\cdot)$
- We want a tight lower bound, and given some current estimate $\theta^t$, we will pick $q(\cdot)$ such that our lower bound holds with equality at $\theta^t$
- Choose $q(z_n) \propto p(x_n, z_n | \theta^t)$! Since $q(\cdot)$ is a distribution, we have

$$q(z_n) = \frac{p(x_n, z_n | \theta^t)}{\sum_k p(x_n, z_n = k | \theta^t) = \frac{p(x_n, z_n | \theta^t)}{p(x_n | \theta^t)} = p(z_n | x_n; \theta^t)$$
Which $q(z)$ to choose?

Recall:

$$
\ell(\theta) = \sum_n \log \sum_{z_n} p(x_n, z_n | \theta) = \sum_n \log \sum_{z_n} q(z_n) \frac{p(x_n, z_n | \theta)}{q(z_n)}
$$

$$
\geq \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta)}{q(z_n)}
$$

- The lower bound we derived for $\ell(\theta)$ holds for all choices of $q(\cdot)$
- We want a *tight* lower bound, and given some current estimate $\theta^t$, we will pick $q(\cdot)$ such that our lower bound holds *with equality* at $\theta^t$
- Choose $q(z_n) \propto p(x_n, z_n | \theta^t)!$ Since $q(\cdot)$ is a distribution, we have

$$
q(z_n) = \frac{p(x_n, z_n | \theta^t)}{\sum_k p(x_n, z_n = k | \theta)} = \frac{p(x_n, z_n | \theta^t)}{p(x_n | \theta^t)} = p(z_n | x_n; \theta^t)
$$

- This is the *posterior distribution* of $z_n$ given $x_n$ and $\theta^t$
E and M Steps

E-Step: For all \(n\), compute \(q(z_n) = p(z_n | x_n; \theta_t)\).

Why is this called the E-Step? Because we can view it as computing the expected (complete) log-likelihood:

\[
Q(\theta | \theta_t) = \sum_n \sum_{z_n} p(z_n | x_n; \theta_t) \log p(x_n, z_n | \theta) = \sum_n E_{q(z_n)} \log p(x_n, z_n | \theta)
\]

M-Step: Maximize \(Q(\theta | \theta_t)\), i.e.,

\[
\theta_{t+1} = \arg \max_\theta Q(\theta | \theta_t)
\]
E and M Steps

**E-Step**: For all \( n \), compute \( q(z_n) = p(z_n|x_n; \theta^t) \)

*Why is this called the E-Step?*
**E and M Steps**

**E-Step:** For all $n$, compute $q(z_n) = p(z_n|x_n; \theta_t)$

*Why is this called the E-Step?* Because we can view it as computing the expected (complete) log-likelihood:

$$Q(\theta|\theta^t) = \sum_n \sum_{z_n} p(z_n|x_n; \theta^t) \log p(x_n, z_n|\theta) = \sum_n \mathbb{E}_{q_{z_n}} \log p(x_n, z_n|\theta)$$
**E and M Steps**

**E-Step**: For all $n$, compute $q(z_n) = p(z_n|x_n; \theta^t)$

*Why is this called the E-Step?* Because we can view it as computing the expected (complete) log-likelihood:

$$Q(\theta|\theta^t) = \sum_n \sum_{z_n} p(z_n|x_n; \theta^t) \log p(x_n, z_n|\theta) = \sum_n \mathbb{E}_{q_{z_n}} \log p(x_n, z_n|\theta)$$

**M-Step**: Maximize $Q(\theta|\theta^t)$, i.e., $\theta^{t+1} = \arg \max_{\theta} Q(\theta|\theta^t)$
EM in Pictures

(Figure from tutorial by Sean Borman)
What is the E-step in GMM?

\[ r_{nk} = p(z = k|x_n; \theta^{(t)}) \]
What is the E-step in GMM?

\[ r_{nk} = p(z = k|x_n; \theta^{(t)}) \]

What is the M-step in GMM? The Q-function is

\[ Q(\theta, \theta^{(t)}) = \sum_n \sum_k p(z = k|x_n; \theta^{(t)}) \log p(x_n, z = k|\theta) \]
What is the E-step in GMM?

\[ r_{nk} = p(z = k|x_n; \theta^{(t)}) \]

What is the M-step in GMM? The Q-function is

\[ Q(\theta, \theta^{(t)}) = \sum_n \sum_k p(z = k|x_n; \theta^{(t)}) \log p(x_n, z = k|\theta) \]

\[ = \sum_n \sum_k r_{nk} \log p(x_n, z = k|\theta) \]
Example: applying EM to GMMs

What is the E-step in GMM?

\[ r_{nk} = p(z = k | x_n; \theta^{(t)}) \]

What is the M-step in GMM? The \( Q \)-function is

\[
Q(\theta, \theta^{(t)}) = \sum_n \sum_k p(z = k | x_n; \theta^{(t)}) \log p(x_n, z = k | \theta) \\
= \sum_n \sum_k r_{nk} \log p(x_n, z = k | \theta) \\
= \sum_k \sum_n r_{nk} \log p(z = k) p(x_n | z = k)
\]
Example: applying EM to GMMs

What is the E-step in GMM?

\[ r_{nk} = p(z = k|x_n; \theta^{(t)}) \]

What is the M-step in GMM? The Q-function is

\[
Q(\theta, \theta^{(t)}) = \sum_n \sum_k p(z = k| x_n; \theta^{(t)}) \log p(x_n, z = k| \theta) = \sum_n \sum_k r_{nk} \log p(x_n, z = k| \theta) = \sum_k \sum_n r_{nk} \log p(z = k) p(x_n| z = k) = \sum_k \sum_n r_{nk} [\log \omega_k + \log \mathcal{N}(x_n| \mu_k, \Sigma_k)]
\]

We have recovered the parameter estimation algorithm for GMMs that we previously discussed
Iterative and monotonic improvement

- We can show that $\ell(\theta^{t+1}) \geq \ell(\theta^t)$
Iterative and monotonic improvement

- We can show that $\ell(\theta^{t+1}) \geq \ell(\theta^t)$
- Recall that we chose $q(\cdot)$ in the E-step such that:

$$
\ell(\theta^t) = \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta^t)}{q(z_n)}
$$

However, in the M-step, $\theta^{t+1}$ is chosen to maximize the right hand side of the equation, thus proving our desired result

- Note: the EM procedure converges but only to a local optimum
• We can show that $\ell(\theta^{t+1}) \geq \ell(\theta^t)$

• Recall that we chose $q(\cdot)$ in the E-step such that:

$$
\ell(\theta^t) = \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta^t)}{q(z_n)}
$$

• However, in the M-step, $\theta^{t+1}$ is chosen to maximize the right hand side of the equation, thus proving our desired result
Iterative and monotonic improvement

- We can show that $\ell(\theta^{t+1}) \geq \ell(\theta^t)$
- Recall that we chose $q(\cdot)$ in the E-step such that:

$$
\ell(\theta^t) = \sum_n \sum_{z_n} q(z_n) \log \frac{p(x_n, z_n | \theta^t)}{q(z_n)}
$$

- However, in the M-step, $\theta^{t+1}$ is chosen to maximize the right hand side of the equation, thus proving our desired result
- Note: the EM procedure converges but only to a local optimum
EM as Coordinate Ascent

- Suppose we want: \( \max_a \max_b F(a, b) \)
- Coordinate ascent:
  - Fix \( a \), minimize with respect to \( b \)
  - Fix \( b \), minimize with respect to \( a \)
- Guaranteed to converge (to local optimum)
You should know . . .

- EM is a general procedure for maximizing a likelihood with latent (unobserved) variables.
You should know . . .

- EM is a general procedure for maximizing a likelihood with *latent (unobserved)* variables
- The two steps of EM:
You should know . . .

- EM is a general procedure for maximizing a likelihood with latent (unobserved) variables
- The two steps of EM:
  - (1) Estimating unobserved data from observed data and current parameters
You should know . . .

• EM is a general procedure for maximizing a likelihood with *latent (unobserved) variables*

• The two steps of EM:
  
  • (1) Estimating unobserved data from observed data and current parameters
  
  • (2) Using this “complete” data to find the maximum likelihood parameter estimates
You should know . . .

• EM is a general procedure for maximizing a likelihood with latent (unobserved) variables

• The two steps of EM:
  • (1) Estimating unobserved data from observed data and current parameters
  • (2) Using this “complete” data to find the maximum likelihood parameter estimates

• Pros: Guaranteed to converge, no parameters to tune (e.g., compared to gradient methods)
You should know . . .

- EM is a general procedure for maximizing a likelihood with *latent (unobserved) variables*
- The two steps of EM:
  - (1) Estimating unobserved data from observed data and current parameters
  - (2) Using this “complete” data to find the maximum likelihood parameter estimates
- Pros: Guaranteed to converge, no parameters to tune (e.g., compared to gradient methods)
- Cons: Can get stuck in local optima, can be expensive
You should know . . .

- EM is a general procedure for maximizing a likelihood with latent (unobserved) variables.
- The two steps of EM:
  - (1) Estimating unobserved data from observed data and current parameters
  - (2) Using this “complete” data to find the maximum likelihood parameter estimates
- Pros: Guaranteed to converge, no parameters to tune (e.g., compared to gradient methods)
- Cons: Can get stuck in local optima, can be expensive
- Why is EM useful for unsupervised learning?
You should know . . .

- EM is a general procedure for maximizing a likelihood with latent (unobserved) variables
- The two steps of EM:
  1. Estimating unobserved data from observed data and current parameters
  2. Using this “complete” data to find the maximum likelihood parameter estimates
- Pros: Guaranteed to converge, no parameters to tune (e.g., compared to gradient methods)
- Cons: Can get stuck in local optima, can be expensive
- Why is EM useful for unsupervised learning?
  - EM is a general method to deal with hidden data; we have studied it in the context of hidden labels (unsupervised learning)