Overview

- In last lecture, we covered PCA which was an unsupervised learning algorithm.
  - Its main purpose was to reduce the dimension of the data.
  - In practice, even though data is very high dimensional, it can be well represented in low dimensions.
- This method relies on an interesting assumption that data depends on some latent variables that are never observed. Such models are called latent variable models.
  - For PCA, these corresponds to the code vectors (representation).
  - Today’s lecture: K-means, a simple algorithm for clustering, i.e. grouping data points into clusters
  - Today’s lecture: reformulate clustering as a latent variable model, apply the EM algorithm
  - Next lecture: recommender systems using matrix factorization.
Clustering

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

- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.
- Grouping data points into clusters, with no observed labels, is called clustering. It is an unsupervised learning technique.
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)
  - But topics are never observed (unsupervised).
Clustering problem

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

Let’s formulate this as an optimization problem

- **K-means Objective:**
  Find cluster centers \( \{m_k\}_{k=1}^K \) and assignments \( \{r^{(n)}\}_{n=1}^N \) to minimize the sum of squared distances of data points \( \{x^{(n)}\} \) to their assigned cluster centers

  - Data sample \( n = 1, \ldots, N \): \( x^{(n)} \in \mathbb{R}^D \) (observed),
  - Cluster center \( k = 1, \ldots, K \): \( m_k \in \mathbb{R}^D \) (not observed),
  - Responsibilities: Cluster assignment for sample \( n \):
    \( r^{(n)} \in \mathbb{R}^K \) 1-of-K encoding (not observed)

- Mathematically:

  \[
  \min_{\{m_k\},\{r^{(n)}\}} J(\{m_k\}, \{r^{(n)}\}) = \min_{\{m_k\},\{r^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \|m_k - x^{(n)}\|^2
  \]

  where \( r_k^{(n)} = \mathbb{I}[x^{(n)} \text{ is assigned to cluster } k] \), i.e., \( r^{(n)} = [0, \ldots, 1, \ldots, 0]^\top \)

- Finding an optimal solution is an NP-hard problem!
K-means Objective

- Optimization problem:

\[
\min_{\{m_k\}, \{r(n)\}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2
\]

distance between \(x^{(n)}\) and its assigned cluster center

- Since \(r_k^{(n)} = \mathbb{I}[x^{(n)} \text{ is assigned to cluster } k]\), i.e., \(r^{(n)} = [0, \ldots, 1, \ldots, 0]^\top\)
- Inner sum is over \(K\) terms but only one of them is non-zero.
- E.g. say sample \(x^{(n)}\) is assigned to cluster \(k = 3\), then

\[
\sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2 = \|m_3 - x^{(n)}\|^2
\]
How to optimize?: Alternating Minimization

Optimization problem:

\[
\min_{\{m_k\}, \{r(n)\}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2
\]

- Problem is hard when minimizing jointly over the parameters \(\{m_k\}, \{r^{(n)}\}\)

- But note that if we fix one and minimize over the other, then it becomes easy.

- Doesn’t guarantee the same solution!
How to optimize?: Alternating Minimization

Optimization problem:

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

Note:

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

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

- Assign each point to the cluster with the nearest center

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

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

$$r^{(n)} = [0, 0, \ldots, 1, \ldots, 0]^\top$$

Only $\hat{k}$-th entry is 1
Alternating Minimization

- Likewise, if we fix the assignments \( \{ r^{(n)} \} \) then can easily find optimal centers \( \{ m_k \} \)
  - Set each cluster’s center to the average of its assigned data points:
    
    For \( l = 1, 2, ..., K \)

    \[
    0 = \frac{\partial}{\partial m_l} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{k}^{(n)} ||m_k - x^{(n)}||^2
    \]

    \[
    = 2 \sum_{n=1}^{N} r_{l}^{(n)} (m_l - x^{(n)}) \quad \implies \quad m_l = \frac{\sum_{n} r_{l}^{(n)} x^{(n)}}{\sum_{n} r_{l}^{(n)}}
    \]

- Let’s alternate between minimizing \( J(\{ m_k \}, \{ r^{(n)} \} ) \) with respect to \( \{ m_k \} \) and \( \{ r^{(n)} \} \)

- This is called alternating minimization
K-means Algorithm

High level overview of algorithm:

- **Initialization**: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - **Assignment step**: Assign each data point to the closest cluster
  - **Refitting step**: Move each cluster center to the mean of the data assigned to it
Figure from Bishop

The K-means Algorithm

- **Initialization**: Set K cluster means $m_1, \ldots, m_K$ to random values

- Repeat until convergence (until assignments do not change):
  
  - **Assignment**: Optimize $J$ w.r.t. $\{r\}$: Each data point $x^{(n)}$ assigned to nearest center
    \[
    \hat{k}^{(n)} = \arg \min_k ||m_k - x^{(n)}||^2
    \]
    
    and Responsibilities (1-hot or 1-of-$K$ encoding)
    
    $r_k^{(n)} = \mathbb{I}[\hat{k}^{(n)} = k]$ for $k = 1, \ldots, K$

  - **Refitting**: Optimize $J$ w.r.t. $\{m\}$: Each center is set to mean of data assigned to it
    
    $m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$. 
K-means for Vector Quantization

- Given image, construct "dataset" of pixels represented by their RGB pixel intensities
- Run k-means, replace each pixel by its cluster center
K-means for Image Segmentation

- Given image, construct “dataset” of pixels, represented by their RGB pixel intensities and grid locations
- Run k-means (with some modifications) to get superpixels
Questions about K-means

- Why does update set $m_k$ to mean of assigned points?
- What if we used a different distance measure?
- How can we choose the best distance?
- How to choose $K$?
- Will it converge?

Hard cases – unequal spreads, non-circular spreads, in-between points
Why K-means Converges

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

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

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

K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.
The objective $J$ is non-convex (so coordinate descent on $J$ is not guaranteed to converge to the global minimum)

There is nothing to prevent k-means getting stuck at local minima.

We could try many random starting points
Soft K-means

- Instead of making hard assignments of data points to clusters, we can make **soft assignments**. One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
  - Allows a cluster to use more information about the data in the refitting step.
  - How do we decide on the soft assignments?
  - We already saw this in multi-class classification:
    - 1-of-$K$ encoding vs softmax assignments
Soft K-means Algorithm

- **Initialization**: Set K means \{m_k\} to random values
- Repeat until convergence (measured by how much \( J \) changes):
  - **Assignment**: Each data point \( n \) given soft ”degree of assignment” to each cluster mean \( k \), based on responsibilities

\[
 r_k^{(n)} = \frac{\exp[-\beta \|m_k - x^{(n)}\|^2]}{\sum_j \exp[-\beta \|m_j - x^{(n)}\|^2]}
\]

\[\Rightarrow r^{(n)} = \text{softmax}(-\beta \{\|m_k - x^{(n)}\|^2\}_{k=1}^{K})\]

- **Refitting**: Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

\[
m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}
\]
Questions about Soft K-means

Some remaining issues

- How to set $\beta$?
- Clusters with unequal weight and width?

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

As $\beta \to \infty$, soft k-Means becomes k-Means! (Exercise)
A Generative View of Clustering

- Next: probabilistic formulation of clustering
- We need a sensible measure of what it means to cluster the data well
  - This makes it possible to judge different methods
  - It may help us decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we adjust the model parameters using maximum likelihood i.e. to maximize the probability that it would produce exactly the data we observed
The Generative Model

- We’ll be working with the following generative model for data \( \mathcal{D} \)
- Assume a datapoint \( x \) is generated as follows:
  - Choose a cluster \( z \) from \( \{1, \ldots, K\} \) such that \( p(z = k) = \pi_k \)
  - Given \( z \), sample \( x \) from a Gaussian distribution \( \mathcal{N}(x|\mu_z, \mathbf{I}) \)
- Can also be written:
  \[
  p(z = k) = \pi_k \\
  p(x|z = k) = \mathcal{N}(x|\mu_k, \mathbf{I})
  \]
Clusters from Generative Model

- This defines joint distribution $p(z, x) = p(z)p(x|z)$ with parameters $\{\pi_k, \mu_k\}_{k=1}^K$

- The marginal of $x$ is given by $p(x) = \sum_z p(z, x)$

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

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

and tells us the probability $x$ came from the $k^{th}$ cluster
The Generative Model

- 500 points drawn from a mixture of 3 Gaussians.

(a) Samples from $p(x \mid z)$  
(b) Samples from the marginal $p(x)$  
(c) Responsibilities $p(z \mid x)$
How should we choose the parameters \( \{\pi_k, \mu_k\}_{k=1}^K \)?

Maximum likelihood principle: choose parameters to maximize likelihood of observed data.

We don’t observe the cluster assignments \( z \), we only see the data \( x \).

Given data \( D = \{x^{(n)}\}_{n=1}^N \), choose parameters to maximize:

\[
\log p(D) = \sum_{n=1}^N \log p(x^{(n)})
\]

We can find \( p(x) \) by marginalizing out \( z \):

\[
p(x) = \sum_{k=1}^K p(z = k, x) = \sum_{k=1}^K p(z = k)p(x|z = k)
\]
Gaussian Mixture Model (GMM)

What is $p(x)$?

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

- This distribution is an example of a Gaussian Mixture Model (GMM), and $\pi_k$ are known as the mixing coefficients.

- In general, we would have different covariance for each cluster, i.e., $p(x | z = k) = \mathcal{N}(x | \mu_k, \Sigma_k)$. For this lecture, we assume $\Sigma_k = I$ for simplicity.

- If we allow arbitrary covariance matrices, GMMs are universal approximators of densities (if you have enough Gaussians). Even diagonal GMMs are universal approximators.
Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

  ![Gaussian Fit](image)

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

    ![GMM Fit](image)

[Slide credit: K. Kutulakos]
Visualizing a Mixture of Gaussians – 2D Gaussians
Fitting GMMs: Maximum Likelihood

Maximum likelihood objective:

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

- How would you optimize this w.r.t. parameters $\{\pi_k, \mu_k\}$?
  - No closed form solution when we set derivatives to 0
  - Difficult because sum inside the log
- One option: gradient ascent. Can we do better?
- Can we have a closed form update?
**Observation:** if we knew $z^{(n)}$ for every $x^{(n)}$, (i.e. our dataset was $D_{\text{complete}} = \{(z^{(n)}, x^{(n)})\}_{n=1}^N$) the maximum likelihood problem is easy:

$$
\log p(D_{\text{complete}}) = \sum_{n=1}^N \log p(z^{(n)}, x^{(n)})
$$

$$
= \sum_{n=1}^N \log p(x^{(n)}|z^{(n)}) + \log p(z^{(n)})
$$

$$
= \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}[z^{(n)} = k] \left( \log \mathcal{N}(x^{(n)}|\mu_k, \mathbf{I}) + \log \pi_k \right)
$$
Maximum Likelihood

$$\log p(D_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} I[z^{(n)} = k] \left( \log \mathcal{N}(x^{(n)} | \mu_k, I) + \log \pi_k \right)$$

- We have been optimizing something similar for Naive bayes classifiers
- By maximizing \( \log p(D_{\text{complete}}) \), we would get this:

\[
\hat{\mu}_k = \frac{\sum_{n=1}^{N} I[z^{(n)} = k] x^{(n)}}{\sum_{n=1}^{N} I[z^{(n)} = k]} = \text{class means}
\]

\[
\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^{N} I[z^{(n)} = k] = \text{class proportions}
\]
Maximum Likelihood

- We haven’t observed the cluster assignments $z^{(n)}$, but we can compute $p(z^{(n)}|x^{(n)})$ using Bayes rule.

- Conditional probability (using Bayes rule) of $z$ given $x$

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

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

$$= \frac{\pi_k \mathcal{N}(x|\mu_k, I)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, I)}$$
Maximum Likelihood

\[
\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k] (\log \mathcal{N}(x^{(n)}|\mu_k, \mathbf{I}) + \log \pi_k)
\]

- We don’t know the cluster assignments \(\mathbb{I}[z^{(n)} = k]\), but we know their expectation \(\mathbb{E}[\mathbb{I}[z^{(n)} = k] | x^{(n)}] = p(z^{(n)} = k | x^{(n)})\).
- If we plug in \(r_k^{(n)} = p(z^{(n)} = k | x^{(n)})\) for \(\mathbb{I}[z^{(n)} = k]\), we get:

\[
\sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} (\log \mathcal{N}(x^{(n)}|\mu_k, \mathbf{I}) + \log \pi_k)
\]

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

\[
\hat{\mu}_k = \frac{\sum_{n=1}^{N} r_k^{(n)} x^{(n)}}{\sum_{n=1}^{N} r_k^{(n)}} \quad \hat{\pi}_k = \frac{\sum_{n=1}^{N} r_k^{(n)}}{N}
\]

- Note: this only works if we treat \(r_k^{(n)} = \frac{\pi_k \mathcal{N}(x^{(n)}|\mu_k, \mathbf{I})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x^{(n)}|\mu_j, \mathbf{I})}\) as fixed.
How Can We Fit a Mixture of Gaussians?

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

  1. **E-step**: Compute the posterior probabilities \( r_k^{(n)} = p(z^{(n)} = k | x^{(n)}) \) given our current model - i.e. how much do we think a cluster is responsible for generating a datapoint.

  2. **M-step**: Use the equations on the last slide to update the parameters, assuming \( r_k^{(n)} \) are held fixed- change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.
EM Algorithm for GMM

- Initialize the means $\hat{\mu}_k$ and mixing coefficients $\hat{\pi}_k$
- Iterate until convergence:
  
  ▶ **E-step:** Evaluate the responsibilities $r_k^{(n)}$ given current parameters
  
  $$
  r_k^{(n)} = p(z^{(n)} = k|x^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(x^{(n)} | \hat{\mu}_k, I)}{\sum_{j=1}^{K} \hat{\pi}_j \mathcal{N}(x^{(n)} | \hat{\mu}_j, I)} = \frac{\hat{\pi}_k \exp\left\{-\frac{1}{2} \|x^{(n)} - \hat{\mu}_k\|^2\right\}}{\sum_{j=1}^{K} \hat{\pi}_j \exp\left\{-\frac{1}{2} \|x^{(n)} - \hat{\mu}_j\|^2\right\}}
  $$

  ▶ **M-step:** Re-estimate the parameters given current responsibilities
    
    $$
    \hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_k^{(n)} x^{(n)}
    $$
    
    $$
    \hat{\pi}_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} r_k^{(n)}
    $$

  ▶ Evaluate log likelihood and check for convergence
    
    $$
    \log p(D) = \sum_{n=1}^{N} \log \left( \sum_{k=1}^{K} \hat{\pi}_k \mathcal{N}(x^{(n)} | \hat{\mu}_k, I) \right)
    $$
The maximum likelihood objective \( \sum_{n=1}^{N} \log p(x^{(n)}) \) was hard to optimize.

The complete data likelihood objective was easy to optimize:

\[
\sum_{n=1}^{N} \log p(z^{(n)}, x^{(n)}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k] \left( \log \mathcal{N}(x^{(n)} | \mu_k, I) + \log \pi_k \right)
\]

We don’t know \( z^{(n)} \)'s (they are latent), so we replaced \( \mathbb{I}[z^{(n)} = k] \) with responsibilities \( r_{k}^{(n)} = p(z^{(n)} = k|x^{(n)}) \).

That is: we replaced \( \mathbb{I}[z^{(n)} = k] \) with its expectation under \( p(z^{(n)}|x^{(n)}) \) (E-step).
We ended up with the expected complete data log-likelihood:

\[
\sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)}|x^{(n)})}[\log p(z^{(n)}, x^{(n)})] = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{k}^{(n)} \left( \log \mathcal{N}(x^{(n)}|\mu_{k}, I) + \log \pi_{k} \right)
\]

which we maximized over parameters \(\{\pi_{k}, \mu_{k}\}_{k}\) (M-step)

The EM algorithm alternates between:

- The E-step: computing the \(r_{k}^{(n)} = p(z^{(n)} = k|x^{(n)})\) (i.e. expectations \(\mathbb{E}[\mathbb{I}[z^{(n)} = k]|x^{(n)}]\)) given the current model parameters \(\pi_{k}, \mu_{k}\)
- The M-step: update the model parameters \(\pi_{k}, \mu_{k}\) to optimize the expected complete data log-likelihood
Relation to k-Means

- The K-Means Algorithm:
  1. Assignment step: Assign each data point to the closest cluster
  2. Refitting step: Move each cluster center to the average of the data assigned to it

- The EM Algorithm:
  1. E-step: Compute the posterior probability over $z$ given our current model
  2. M-step: Maximize the probability that it would generate the data it is currently responsible for.

- Can you find the similarities between the soft k-Means algorithm and EM algorithm with shared covariance $\frac{1}{\beta}I$?

- Both rely on alternating optimization methods and can suffer from bad local optima.
Further Discussion

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

- Possible problems with maximum likelihood objective:
  - **Singularities**: Arbitrarily large likelihood when a Gaussian explains a single point with variance shrinking to zero
  - Non-convex

- EM is more general than what was covered in this lecture. Here, EM algorithm is used to find the optimal parameters under the GMMs.
GMM Recap

- A probabilistic view of clustering - Each cluster corresponds to a different Gaussian.

- Model using latent variables.

- General approach, can replace Gaussian with other distributions (continuous or discrete)

- More generally, mixture models are very powerful models, i.e. universal distribution approximators

- Optimization is done using the EM algorithm.