CS 181 LECTURE 3/19/24

Scribe Notes

Contents

1 Review **3**

2 Mixture Models **3**

2.1 Why Gaussian? 4

2.2 Training a GMM 4

2.3 Expectation Maximization 5

2.4 Optimizing ELBO 6

2.5 Summary 6

2.6 K-means vs. GMM 7

2.6.1 Monotonic Improvement 7

2.6.2 Flexible and Soft Assignments 7

2.6.3 GMM and K means Similarity 7

3 Concept Check **7**

4 Concept Check Solution **8**

5 Pictures of the Board **9**

1 Review

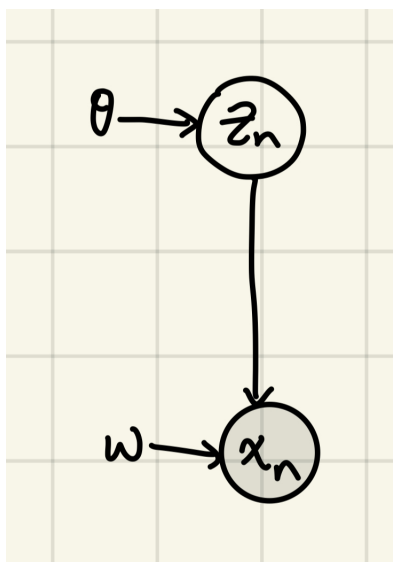
Last time, we talked about k-means. There are two main issues to consider: 1) how to measure a "good" cluster and 2) how restrictive k-means is. One case that is problematic is when points lie between clusters. Just choosing to assign the point to one of the clusters does not feel very principled.

Today, we will look at clustering from a probablistic perspective. We are still doing unsupervised learning but this time probablistic for discrete data.

Gaussian mixture models is the most natural version of clustering.

2 Mixture Models

Mixture models are a generative view. The data generation process is one where we have latent variables z_n, one latent variable for each data point.



We shade the variable that we observe.

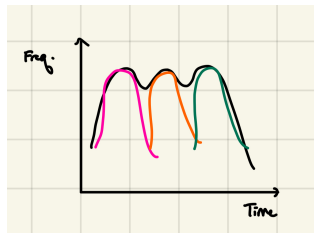
An example: say we ask, what is your average commute from your dorm to the SEC? The latent variable, z_n, is the dorm/house you live in and x_n is your commute time. Your commute time depends on where you live.

$$z_n \sim \pi(\text{cluster probs})$$

$$x_n \sim p(x|z_n, w)$$

Imagine the black line is the distribution. As you can observe, the model is composed of three guassian models.

This happens because where you live on campus influences your distribution of commute times.



2.1 Why Gaussian?

In the real-world, often times, data looks messy when in aggregate. But there may exist a simple group structure. In many cases, the structure is so simple we can estimate it with Gaussian. For example, we can examine the frequency of heights in society. When we split the group into males and females, then we see we can model per group distribution in a simple way.

GMM's are universal approximators, which means that for an arbitruary distribution, if we have enough mixture components, we can create a mixture model that represents the distribution.

Theorem 2.1. GMM as a universal approximator: For arbitrary P(x); there exists a GMM $g(x; \mu, \Sigma, \pi)$ such that $D(p(x), g) \to 0$ as $k \to \infty$. Here, D is a distance function and k represents the number of components.

2.2 Training a GMM

We are training the complete (x, z) likelihood.

$$p(x, z|w, \theta)$$

where θ is the parameters of $p(z)$ and w is μ, Σ.

$$
\begin{aligned}
\mathcal{L}(w,\theta) = \sum_{n=1}^{N} \log p(x_n, z_n|w,\theta) &= \sum_{n=1}^{N} [\log p(x_n|z_n) + \log p(z_n|\theta)] \\
&= \sum_{n=1}^{N} \log N(x_k|\mu_{z_n}, \Sigma_{z_n}) + \log \pi_{z_n} \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} I(z_n = C_k)[\log(x_k, \mu_k, \Sigma_k) + \log \pi_k]
\end{aligned}
$$

First, we applied Bayes rule to expand the expression. Then, we wrote the sum as a double sum at the last step by using our indicator variables, where $I(z_n = C_k) = 1$ when z_n is in class k.

Let us redefine $\theta = (\theta, w)$. We know the assignment of all the points.

$$\pi_k = \frac{n_k}{N}$$

$$\mu_k = \text{mean of all the points labeled } k$$

$$\sigma_k = \text{standard deviation of all the points labeled } k$$

This is similar to what we did previously. But the problem is that the z_n's are hidden from us. We are trying to estimate something when we don't have all the data. A probablistic take on this would say we should marginalize z. Using the total law of probability,

$$p(x|\theta) = \sum_k p(x|z_n = C_k, \theta) p(z_n = C_k)$$

$$
\begin{aligned}
\sum_n \log p(x_n|\theta) &= \sum_n \log \sum_k p(x|z_n = C_k, \theta) p(z_n = C_k) \\
&= \log \mathbf{E}_{z_n \sim p(z,\theta) p(x_n|Z,\theta)}
\end{aligned}
$$

Now, we have this expression for the expectation. We want to optimize this expectation. But computing log of an expectation is mathematically difficult. Secondly, we have a problem that θ is in the expectation. So, computing a gradient of the expression would be hard. We cannot solve this objective.

2.3 Expectation Maximization

That is where the EM (Expectation Maximization) process comes in. What we will do is separate the problem into two steps.

Let us introduce q_n where q_n is a probability vector that is a proxy for $p(z)$.

$$z_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow q_n = \begin{bmatrix} 0.1 \\ 0.7 \\ \vdots \\ 0.4 \end{bmatrix}$$

$$\begin{aligned}
\log p(x|\theta) &= \log \sum_k p(x|z_n = C_k, \theta) p(z_n = C_k) \\
&= \log q_n(z = C_k)[\frac{\sum_k p(x|z_n = C_k, \theta) p(z_n = C_k)}{q_n(z = C_k)}] \\
&= \log \mathbf{E}_{z \sim q(z)}[\frac{p(x|z_n = C_k, \theta) p(z_n = C_k)}{q(z)}]
\end{aligned}$$

Now, we use Jensen's Inequality to flip the log and the expectation.

$$\geq \mathbf{E}_{z \sim q(z)} \log[\frac{p(x|z_n = C_k, \theta) p(z_n = C_k)}{q(z)}]$$

We call this new expression the Evidence Lower Bound, known as the ELBO. Our goal is now to optimize the ELBO.

2.4 Optimizing ELBO

We do so in two steps that we run alternatingly.

Step 1. Write ELBO as joint distribution.

$$ELBO = \sum_n \mathbf{E}[\frac{\log p(x_n, z_n|\theta)}{q(z_n)}] = \sum_n \mathbf{E}[\log p(x_n, z_n|\theta) - \log q(z_n)]$$

This is now easy to solve!

Once we have the z's, our optimization becomes easy.

$\hat{\theta}$ is the $\text{argmax}_\theta \sum \mathbf{E} \log p(x_n, z_n|\theta)$

This is the M step.

Step 2. E-Step.

$$\begin{aligned}
ELBO &= \sum_n \mathbf{E} \log p(x_n|\theta) p(z_n|x_n, \theta) \\
&= \sum_n \mathbf{E} \log p(x_n|\theta) - \sum_n \mathbf{E} \log \frac{Q(z_n)}{p(z_n|x_n, \theta)}
\end{aligned}$$

2.5 Summary

E Step (Expectation Step)

The E step can be thought of as a "local" operation because it operates on individual data points, determining the likelihood of each point belonging to each cluster. This step is analogous to the assignment step in K-means clustering, where each data point is assigned to

the nearest centroid.

M Step (Maximization Step)
The M step can be considered a "global" operation because it involves updating the parameters of the entire model based on the collective information gathered from all data points. This step is akin to the update step in K-means clustering, where the centroids are recalculated based on the assignments of data points to clusters.

2.6 K-means vs. GMM

2.6.1 Monotonic Improvement

GMM exhibits a monotonic improvement in likelihood with each iteration during training. This means that as the model iteratively optimizes its parameters, the likelihood of the data given the model consistently increases or remains stable. In contrast, K-means may not exhibit this property due to its reliance on centroid assignment and update steps, which might lead to non-monotonic behavior in the optimization process.

2.6.2 Flexible and Soft Assignments

GMM allows for more flexible and probabilistic assignments of data points to clusters. Unlike K-means, which assigns each data point to the nearest centroid, GMM assigns probabilities to each data point belonging to each cluster. This soft assignment allows GMM to capture complex data distributions more effectively, as data points lying near cluster boundaries are not forced into rigid assignments.

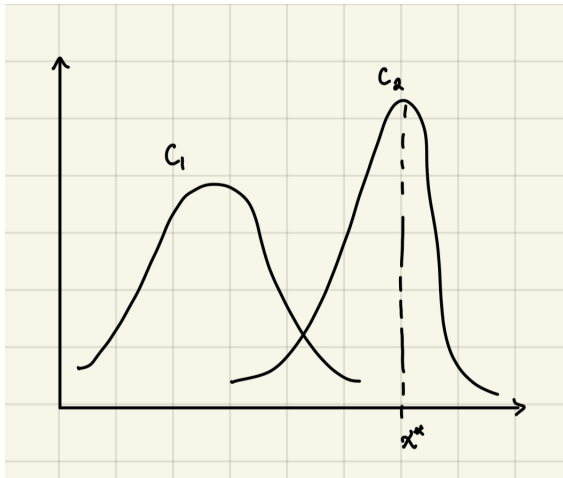
2.6.3 GMM and K means Similarity

When the variance of Gaussian components in a GMM becomes small, the model behaves similarly to K-means.

3 Concept Check

Consider the following two distributions.

What happens to μ_2 as $\Sigma_2 \to 0$?



4 Concept Check Solution

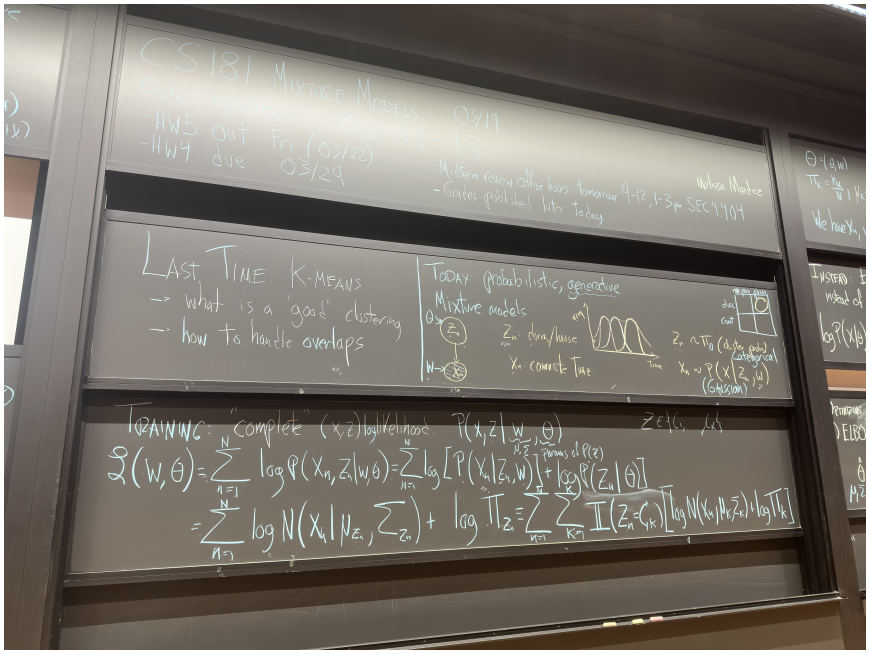
$$\mu_2 \to \infty \text{ as } \Sigma_2 \to 0$$

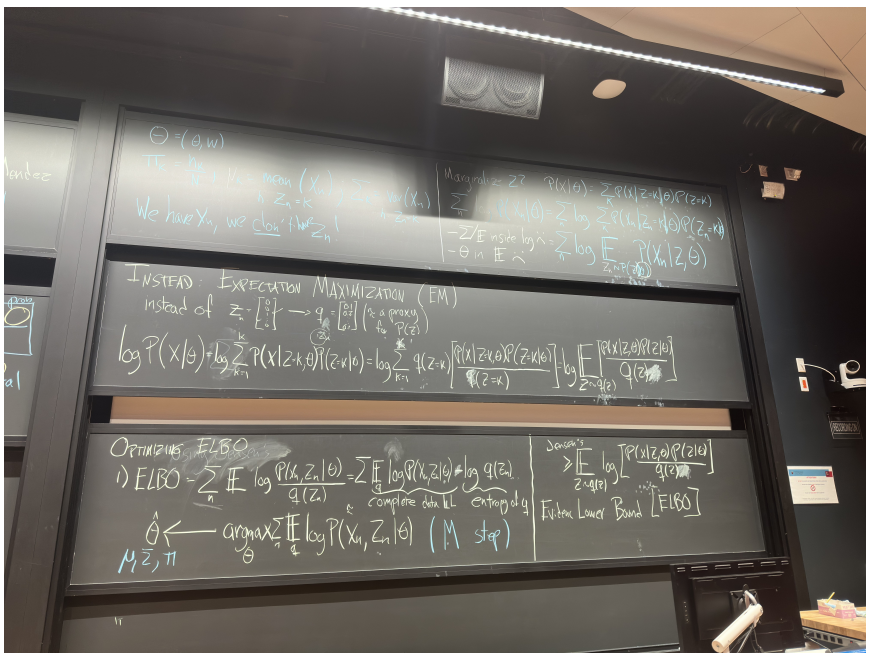
In the context of GMMs, decreasing the variance excessively can lead to a situation where the model assigns almost all probability mass to a single point, effectively reducing the model's ability to capture the variability in the data. This scenario is often referred to as overfitting or collapsing clusters.

To mitigate this issue, it's common practice to add entropy to the model. Entropy in this context refers to a measure of uncertainty or randomness. By adding entropy, we introduce a regularization term that encourages the model to maintain some level of uncertainty and prevents the variance from becoming too small.

One way to add entropy is by initializing the means and variances of the Gaussian components to be different. This initialization strategy ensures that the model starts with a certain level of variability across the clusters, which helps prevent them from collapsing too early in the optimization process.

5 Pictures of the Board




We have X_n, we don't have Z_n!
INSTEAD: EXPECTATION MAXIMIZATION (EM)
instead of
OPTIMIZING ELBO
1) ELBO
complete data LL
entropy of q
(M step)
Jensen's
Evidence Lower Bound [ELBO]