LECTURE 13
Mixture models and latent space models

We now consider models that have extra unobserved variables. The variables are called latent variables or state variables and the general name for these models are state space models.A classic example from genetics/evolution going back to 1894 is whether the carapace of crabs come from one normal or from a mixture of two normal distributions.

We will start with a common example of a latent space model, mixture models.

13.1. Mixture models

13.1.1. Gaussian Mixture Models (GMM)

Mixture models make use of latent variables to model different parameters for different groups (or **clusters**) of data points. For a point x_i, let the cluster to which that point belongs be labeled z_i; where z_i is latent, or unobserved. In this example (though it can be extended to other likelihoods) we will assume our observable features $\mathbf{x}_i$ to be distributed as a Gaussian, so the mean and variance will be cluster-specific, chosen based on the cluster that point x_i is associated with. However, in practice, almost any distribution can be chosen for the observable features. With d-dimensional Gaussian data x_i, our model is:

$$\begin{aligned} z_i \mid \boldsymbol{\pi} &\sim \text{Mult}(\boldsymbol{\pi}) \\ \mathbf{x}_i \mid z_i = k &\sim \mathcal{N}_D(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \end{aligned}$$

where $\boldsymbol{\pi}$ is a point on a K-dimensional simplex, so $\boldsymbol{\pi} \in \mathbb{R}^K$ obeys the following properties: $\sum_{k=1}^K \pi_k = 1$ and $\forall k \in \{1, 2, .., K\} : \pi_k \in [0, 1]$. The variables π are known as the *mixture proportions* and the cluster-specific distributions are called the *mixture components*. Variables $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the k^{th} cluster specific mean and covariance parameters, respectively ($k \in \{1, 2, .., K\}$). Figure 1 is an example of a GMM where $d = 1$ and $K = 3$.

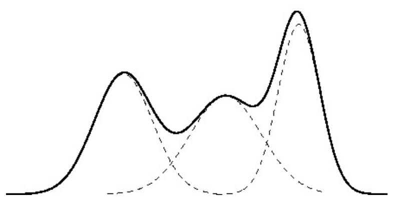

Figure 1. The density of a univariate Gaussian Mixture Model with three Gaussian mixture components, each with their own mean and variance terms ($K = 3$, $d = 1$). [Source: http://prateekvjoshi.com]

The likelihood for $\mathbf{x_i}$ conditioned on parameters is then:

$$\begin{aligned} p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\pi}) &= \sum_{k=1}^{K} p(x_i, z_i = k \mid \pi, \theta) \\ &= \sum_{k=1}^{K} p(z_i = k | \pi_k) p(x_i \mid z_i = k, \theta) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}_d(\mathbf{x_i}; \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}), \end{aligned}$$

which is a weighted, linear sum of Gaussians. This gives a nice interpretation of π_k as the probabilistic 'weight' we place on each cluster k in our model.

13.1.2. Mixture Models for Clustering

Mixture models are often used for clustering; this is a *generative model* because we specifically model $p(z)$ and $p(x \mid z)$. For general parameters $\theta = \{\pi, \mu, \Sigma\}$, the posterior probability of assigning point x_i to cluster k is given by (using Bayes rule):

$$r_{ik} \triangleq p(z_i = k \mid \mathbf{x}_i, \theta) \propto p(\mathbf{x}_i \mid z_i = k, \theta) \; p(z_i = k \mid \pi).$$

Calculating the posterior probability of each cluster for a data point x_i is known as *soft clustering. Hard clustering* is assigning the best cluster z_i^* to data point x_i^* such that

$$z_i^* = \arg\max_k (r_{ik}) = \arg\max_k \; \log\left(p\left(\mathbf{x}_i \mid z_i = k, \theta\right)\right) + \log\left(p\left(z_i = k \mid \pi\right)\right).$$

Hard clustering induces a linear boundary between clusters that assigns points to a single cluster, whereas *soft clustering* computes the probability for each point that it was generated from each of the clusters.

13.1.3. Estimation Attempt for GMM

One possible approach to estimate all of our parameters and clusters is through the MLE process given we have observed $\mathbf{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$]. Our parameters we wish to estimate are $\theta = \{\pi, \mu, \Sigma\}$. For notational simplicity we will write out $z_i = k$ as the vector $\mathbf{z}_i$, where $z_{ij} = \mathbf{1}(j = k)$, meaning that there is a single 1 in a vector of length K of all zeros that indicates the value that the multinomial z_i takes on. This is called the multinomial vector representation. So $\mathbf{z}_i^k$ is defined as the boolean value at the k^{th} position of the $\mathbf{z}_i$ vector. Using this:

$$p(\mathbf{z}_i \mid \pi) = \prod_{k=1}^{K} \pi_k^{\mathbf{z}_i^k}$$

$$p(\mathbf{x}_i \mid \mathbf{z}_i, \theta) = \prod_{k=1}^{K} \mathcal{N}(\mu_k, \Sigma_k)^{\mathbf{z}_i^k}$$

Thus, the log likelihood of the GMM is written as

$$\begin{aligned} \ell(\theta; \mathbf{D}) &= \sum_{i=1}^{N} \log p(\mathbf{x}_i \mid \theta) = \sum_{i=1}^{N} \log \left[\sum_{\mathbf{z}_i=1}^{K} p(\mathbf{x}_i, \mathbf{z}_i \mid \theta) \right] \\ &= \sum_{i=1}^{N} \log \left[\sum_{\mathbf{z}_i \in Z} \prod_{k=1}^{K} \pi_k^{\mathbf{z_i^k}} \mathcal{N}(\mu_k, \Sigma_k)^{\mathbf{z}_i^k} \right] \end{aligned}$$

note that $\sum_{\mathbf{z}_i \in Z}$ indicates the sum through all possible categorical values of $\mathbf{z}_i$

This does not decouple the likelihood because the log cannot be 'pushed' inside the summation; however, if we had observed each $\mathbf{z}_i$, then would this problem decouple? Let's say **we observe** each $\mathbf{z}_i$, so now $\mathbf{D} = \{(\mathbf{x}_1\mathbf{z}_1), \ldots, (\mathbf{x}_N\mathbf{z}_N)]$. The log likelihood now becomes:

$$\begin{aligned} \ell(\theta; \mathbf{D}) &= \log \prod_{i=1}^{N} p(\mathbf{z}_i \mid \pi) p(\mathbf{x}_i \mid \mathbf{z}_i, \theta) \\ &= \sum_{i=1}^{N} \log p(\mathbf{z}_i \mid \pi) + \log p(\mathbf{x}_i \mid \mathbf{z}_i, \theta) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} \left[\mathbf{z}_i^k \log \pi_k + \mathbf{z}_i^k \log \mathcal{N}(\mu_k, \Sigma_k) \right] \end{aligned}$$

Our parameters estimations are now decoupled since we can estimate π_k and μ_k, Σ_k separately. In fact, we have a unimodal posterior distribution $p(\theta \mid \mathbf{D})$ with respect to each of the parameters, so we say that we have **identifiable** parameters.

The issue of **unidentifiable** parameters comes up when we do not have a unique MLE or MAP estimates of θ, meaning our posterior has multiple modes. In other words, it is possible that multiple values of θ produce the same likelihood. In the case of unobserved $\mathbf{z}_i$'s for our GMM, we could not compute a unique MLE estimate of our parameters since the posterior depended on the unobserved $\mathbf{z}_i$'s.

Another problem to consider with mixture models is *label switching*: If we order the clusters A, B, C, and then run it again, it is possible we may get the clusters C, B, A, which would have the same likelihood as clusters A, B, C. This just needs to be considered when we compare different models or different runs from a given model.

Without observations of z, there is no ground truth, so we only have the feature distribution to guess the hidden causes from.

13.2. Expectation Maximization (EM)

The **EM Algorithm** is a general method for parameter estimation when the model depends on unobserved or latent variables, published in 1977 by Demster, Laird, and Rubin. The EM algorithm alternates estimating model parameters, starting from some initial guess, with estimating the values of the latent variables. Each iteration consists of an E step (Expectation step) and an M step (Maximization step). Let $X = \{x_1, \ldots, x_n\}$ be a set of observed variables and $Z = \{z_1, \ldots, z_n\}$ be a set of latent variables. Recall the marginal log likelihood:

$$\begin{aligned}\ell(\theta; Z, X) &= \sum_{i=1}^{N} \log p(x_i \mid \theta) \\ &= \sum_{i=1}^{N} \log \left[\sum_{z_i} p(x_i, z_i \mid \theta)\right].\end{aligned}$$

As discussed in Section 2.3, this is hard to optimize since we have the summation inside the log, which does not allow our latent variables and parameters to separate in this equation, causing multiple possible modes. Instead let's define the **complete data log likelihood**, or the likelihood of the data if we assume that we have complete observations (i.e., latent variables and observed variables both):

$$\ell_C(\theta; Z, X) = \sum_{i=1}^{N} \log p(x_i, z_i \mid \theta)$$

This is simple to work with since we do not have any summations inside the log, and it decouples nicely. But it still depends on the latent states which are unknown. To this end, we will see that the E step in EM to estimates realizations of the latent states. From the complete log likelihood, we can take the expectation of the latent variables with respect to the current values of our parameters:

$$\begin{aligned}Q\left(\theta^{(t)}\right) &= \mathbb{E}_{\theta^{(t)}}\left[\ell_C(\theta; Z, X) \mid \mathbf{D}, \theta^{(t-1)}\right] \\ &= E\left[\sum_{i=1}^{n}\sum_{k=1}^{K} z_i^k \log \pi_k + z_i^k \log \mathcal{N}(\mathbf{x_i}; \boldsymbol{\mu}_\mathbf{k}, \boldsymbol{\Sigma}_\mathbf{k})\right] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} E[z_i^k] \log \pi_k + E[z_i^k] \log \mathcal{N}(\mathbf{x_i}; \boldsymbol{\mu}_\mathbf{k}, \boldsymbol{\Sigma}_\mathbf{k}).\end{aligned}$$

This equation Q as known as the *expected complete log likelihood* or the *auxiliary function.* This function denotes the *expected sufficient statistics* for our model. To solve $Q\left(\theta^{(t)}\right)$, the EM algorithm is composed of two main steps:

- E step: compute the expected sufficient statistics of $\mathbf{z}_i$:

$$r_{ik}^t = \mathbb{E}_{\theta^t}\left[\mathbf{z}_i^k\right]$$

  In this Gaussian mixture model, we will compute $p(\mathbf{z}_i = k \mid \theta, \mathbf{x}_i)$
- M step: update θ^t to θ^{t+1} where

$$\theta^{t+1} = \underset{\theta}{\operatorname{argmax}}\left(Q(\theta^t)\right)$$

Instead of computing the ML, we could also compute the MAP. Then,

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}} \Big(Q(\theta^{(t)}) + \log(p(\theta^{(t)}) \Big)$$

Because these variables are separated, we know that, for a given set of expected values for the z parameters, the ECLL is concave with respect to each of our parameters. Thus, the M step involves maximizing the ECLL with respect to each of our variables by taking the derivative with respect to each of them, setting to zero, and solving, as in our standard MLE process.

13.2.1. EM for GMM

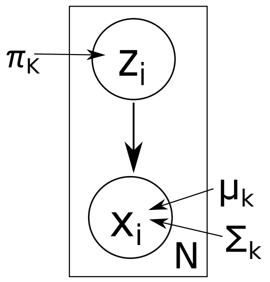

Figure 2. Graphical model of GMM

Here we will show an example of using EM for the GMM we have be discussing (Figure 5 and Section 2.3). Where the parameters we wish to estimate are $\theta = \{\pi_k, \mu_k, \Sigma_k\}$. Our expected complete data log likelihood decouples as:

$$\begin{aligned}
Q(\theta^t) &= \mathbb{E}_{\theta^t}(\ell_C(\theta; Z, X)) \\
&= \mathbb{E}(\sum_{i=1}^{n} \log p(\mathbf{x}_i, \mathbf{z}_i | \theta)) \\
&= \sum_{i=1}^{N} \mathbb{E}_{\theta^t} \left[\log p(z_i|\pi) p(\mathbf{x}_i \mid \mathbf{z}_i, \theta)\right] \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\theta^t} \left[\mathbf{z}_i^k \log(\pi_k p_k(\mathbf{x}_i \mid \mu_k^t, \Sigma_k^t))\right] \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\theta^t} \left[\mathbf{z}_i^k\right] \log(\pi_k p_k(\mathbf{x}_i \mid \mu_k^t, \Sigma_k^t)) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\theta^t} \left[\mathbf{z}_i^k\right] \log(\pi_k) + \mathbb{E}_{\theta^t} \left[\mathbf{z}_i^k\right] \log(p_k(\mathbf{x}_i \mid \mu_k^t, \Sigma_k^t)) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik}^t \log(\pi_k) + r_{ik}^t \log(p_k(\mathbf{x}_i \mid \mu_k^t, \Sigma_k^t))
\end{aligned}$$

Now for each iteration, t, we compute the E step and M step as follows.

- E step:

$$r_{ik}^{t+1} = p\left(\mathbf{z}_i = k \mid \theta, \mathbf{x}_i\right) = \frac{\pi_k^t \, \mathcal{N}(\mathbf{x}_i \mid \mu_k^t, \Sigma_k^t)}{\sum_{j=1}^K \pi_j^t \, \mathcal{N}(\mathbf{x}_i \mid \mu_j^t, \Sigma_j^t)}$$

  r_{ik}^{t+1} is the posterior probability of each cluster assignment to k on a specific data point.
- M step: After we have computed r_i^t, the ML updates are the same as in the Naive Bayes, where we update each parameter by the respective partial derivatives of $Q(\theta^t)$.

$$\begin{aligned}
\pi_k^{t+1} &= \frac{\sum_{i=1}^N r_{ik}^t}{N} \\
\mu_k^{t+1} &= \frac{\sum_{i=1}^N r_{ik}^t \mathbf{x}_i}{\sum_{i=1}^N r_{ik}^t} \\
\Sigma_k^{t+1} &= \frac{\sum_{i=1}^N r_{ik}^t \left(\mathbf{x}_i - \mu_k^{t+1}\right)\left(\mathbf{x}_i - \mu_k^{t+1}\right)^T}{\sum_{i=1}^N r_{ik}^t}
\end{aligned}$$

13.3. K-Means

The **K-means algorithm** is a simple clustering method that aims to partition n observations into K clusters using hard clustering, where each observation is assigned to the cluster with the nearest mean/centroid. As we will see, EM assigns probabilities (or weights) of observations belonging to each cluster. K-means, on the other hand, has no underlying probability model, and instead it assigns each observation to a specific cluster in a *hard clustering* manner. This is why we call the cluster means *centroids*: to emphasize that there is no underlying (Gaussian) probability model. The following steps implement the K-means algorithm.

(1) Set K and choose the initial centroids, η_k (often one can choose these from K data points).
(2) Assign each data point to its closest centroid:

$$z_{ik} = \begin{cases} 1, & \text{if } k = \arg\min_k \|x_i - \eta_k\|^2 \\ 0, & \text{otherwise} \end{cases} \tag{13.1}$$

(3) Update each cluster center by computing the mean of all points assigned to it

$$\eta_k = \frac{\sum_{i=1}^N z_{ik} x_i}{\sum_{i=1}^N z_{ik}}$$

(4) Repeat the second and third steps until cluster assignments do not change between iterations.

An inappropriate choice of K may result in poor results, so it is important to vary K and run diagnostic checks. The Euclidian distance (the term in step 2 defining the distance between point x_i and centroid η_k also need not be the metric minimized; other metrics such as the Mahalanobis distance may also be used. The distance metric may be customized for the specific space and feature set you are working with. Figure 4 illustrates provides a visualization of the K-means algorithm. In essence, K-means pretends we observe the latent states z and updates the cluster-specific centroids based on this (pretend) observation. The next question we should ask is how can we adapt K-means in a probabilistic framework? *We will see that the EM algorithm does this.*

As with GLMs, we can use a gradient-based approach to find the local minima in the likelihood (marginalizing out the latent variables z. Instead, we will follow the ideas in K-means in a probabilistic way.

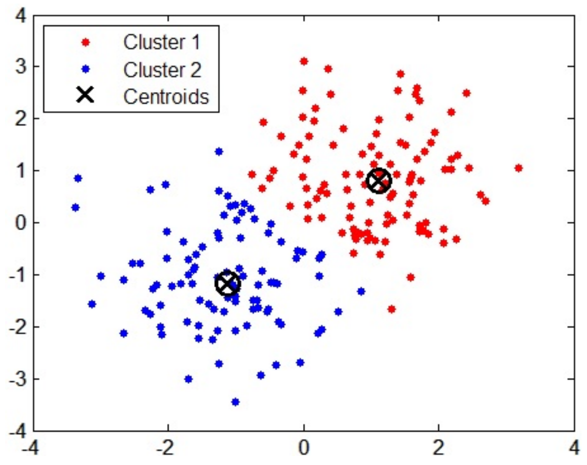

Figure 3. Example of K-means algorithm results. Source: www.mathworks.com.