DS 102 Data, Inference, and Decisions **Fall 2019**

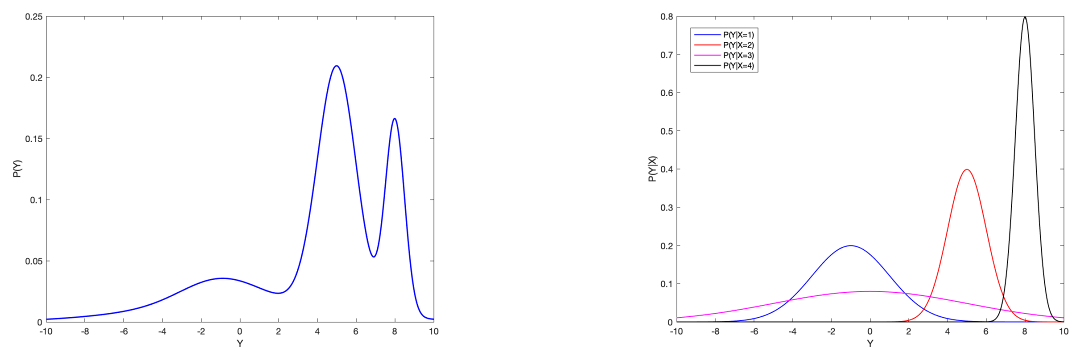
Lecture 9: Bayesian Hierarchical Models 2

Lecturer: Ramesh Sridharan

In the last lecture, we introduced the idea of Gaussian mixture models (GMMs). In this lecture, we will introduce the *Expectation-Maximization* Algorithm as a way of performing unsupervised learning to learn GMMs from data.

1 Gaussian Mixture Models

Suppose we have a random variable Y with an unknown distribution $\mathbb{P}(Y)$ that may not fit into any known class of distributions that we are familiar with. However, we would like to come up with a model of the distribution that we can interpret and sample from easily, that closely matches the distribution of the random variable Y. One common model that is simple to use yet powerful enough to represent complex distributions are mixtures of Gaussians as illustrated in Figure 9.1.



(a) (b)

Figure 9.1: The probability density function of Y and the GMM that describes it.

To model $\mathbb{P}(Y)$ as a mixture of Gaussians we assume that there is more structure underlying the random variable. In particular, we assume that there is a hidden random variable X taking values in $i = 1, 2, ..., d$ such that:

$$\mathbb{P}(Y|X = i) = \mathcal{N}(\mu_i, \sigma_i^2)$$

Thus, $\mathbb{P}(Y) = \sum_{i=1}^{d} \pi_i \mathcal{N}(Y; \mu_i, \sigma_i^2)$, where $\pi_i = \mathbb{P}(X = i)$, and we model $\mathbb{P}(Y)$ as a mixture of d Gaussians.

Example 9.1. For the distribution showed in Figure 9.1, $\pi_1 = 0.1, \pi_2 = 0.5, \pi_3 = 0.2, \pi_4 = 0.2$.

Looking at GMMs from a Bayesian perspective, one can think of π_i as the prevalence that Y came from the normal with mean μ_i and variance σ_i^2.

2 Expectation-Maximization

In the previous section we discussed Gaussian Mixture models and their ability to model complex distributions. In this section we will present a method of learning the parameters of a GMM μ_i, σ_i^2, and π_i from data. One of the biggest problems with learning Gaussian mixture models is that simply maximizing the likelihood of the data does not work since we typically only observe $y_1, ..., y_n$, and not the hidden variables $x_1, ..., x_n$.

If we knew the value of x_j for each y_j we could write the likelihood of the data as:

$$\mathbb{P}(y_j, x_j | \theta_1, ..., \theta_d, \pi_1, ..., \pi_d) = \mathbb{P}(x_j | \pi_1, ..., \pi_d)\mathbb{P}(y_j | x_j, \theta_1, ..., \theta_d) = \prod_{i=1}^{d} (\pi_i \mathcal{N}(y_j; \theta_i))^{\mathbb{I}(x_j = i)},$$

where $\theta_i = (\mu_i, \sigma_i^2)$. The log likelihood of all of the data would therefore be given by:

$$\ell(y, x_j; \theta_1, ..., \theta_d, \pi_1, ..., \pi_d) = \sum_{j=1}^{n} \sum_{i=1}^{d} \mathbb{I}(x_j = i)(\log(\pi_i) + \log(\mathcal{N}(y_j; \mu_i, \sigma_i^2)).$$

which we could maximize over all $\theta_1, ..., \theta_d$ and $\pi_1, ..., \pi_d$.

On the other hand, if we knew all the values of $\theta_1, ..., \theta_d$ and $\pi_1, ..., \pi_d$ we could find the posterior distribution over X_j for each data point y_j. This posterior is given by:

$$\mathbb{P}(X_j = i | y_j) = \frac{\pi_i \mathcal{N}(y_j; \mu_i, \sigma_i^2)}{\sum_{k=1}^{d} \pi_k \mathcal{N}(y_j; \mu_k, \sigma_k^2)}$$

Therefore, we have a chicken and egg scenario. Given data points $y_1, ..., y_n$, if we knew $x_1, ..., x_n$ we could find the values of the parameters of our GMM, $(\pi_i, \mu_i, \sigma_i^2)$ for $i = 1, ..., d$, and if we knew the values of the parameters of our GMM, $(\pi_i, \mu_i, \sigma_i^2)$ for $i = 1, ..., d$, we could predict $x_1, ..., x_n$.

To solve this problem, we introduce the *Expectation-Maximization Algorithm* or EM algorithm for short. There are 3 main ideas behind the EM Algorithm:

1. Randomly initialize θ_i and π_i.
2. Given fixed θ_i and π_i, for each data point y_j approximate the probability that y_j comes from Gaussian i, denoted $Z_j(i) = \mathbb{P}(X_j = i | y_j)$.

3. Given fixed distributions Z_j find the values of θ_i and π_i that maximize the expected likelihood of the data (over the distributions $Z_j(i)$):
$$(\pi^*, \theta^*) = \underset{\pi_1,...,\pi_d,\theta_1,...,\theta_d}{\text{argmax}} \ \mathbb{E}_Z[\ell(y; \theta_1, ..., \theta_n)]$$
Since $\mathbb{E}[\mathbb{I}(x_j = i)] = Pr(X_j = i|Y_i) = Z_j(i)$, this simplifies to:
$$(\pi^*, \theta^*) = \underset{\pi_1,...,\pi_d,\theta_1,...,\theta_d}{\text{argmax}} \sum_{j=1}^{n} \sum_{i=1}^{d} Z_j(i)(\log(\pi_i) + \log(\mathcal{N}(y_j; \mu_i, \sigma_i^2))$$
4. Iterate between the two sub-problems until convergence.

Remark 9.2. The EM algorithm can be shown to maximize the lower bound on the log-likelihood of the data at each iteration, meaning that as the algorithm runs we have more and more confidence that the log-likelihood of the data is improving.

We now outline the EM algorithm with unknown μ_i, σ_i^2, π_i for a mixture of d Gaussians given $y_1, ...y_n$.

Algorithm 1 Expectation-Maximization Algorithm for Gaussian Mixture Models

Input: Data: $y_1, ..., y_n$, Number of Gaussians in the mixture d, number of iterations r
Output: $(\pi_i, \mu_i, \sigma_i^2)$ for $i = 1, ..., d$.
Randomly Initialize $(\pi_{i,0}, \mu_{i,0}, \sigma_{i,0}^2)$ **for** $t = 1$ ***to*** r **do**
    Expectation Step: **for** $j = 1$ ***to*** n **do**
        for $i = 1$ ***to*** d **do**
            $Z_j(i) \leftarrow \frac{\pi_{i,t-1}\mathcal{N}(y_j;\mu_{i,t-1},\sigma_{i,t-1}^2)}{\sum_{k=1}^{d} \pi_{k,t-1}\mathcal{N}(y_j;\mu_{k,t-1},\sigma_{k,t-1}^2)}$
        end
    end
    Maximization Step: **for** $i = 1$ ***to*** d **do**
        $N_{i,t} \leftarrow \sum_{j=1}^{n} Z_j(i)$.
        $\mu_{i,t} \leftarrow \frac{1}{N_{i,t}} \sum_{j=1}^{n} Z_j(i)y_j$.
        $\sigma_{i,t} \leftarrow \frac{1}{N_{i,t}} \sum_{j=1}^{n} Z_j(i)(y_j - \mu_{i,t})^2$.
        $\pi_{i,t} \leftarrow \frac{N_{i,t}}{n}$.
    end
end

Note that the update for μ_i and σ_i^2 are both the maximizers of the expected likelihood using the straightforward derivation we have seen in previous lectures and discussions. The update for π_i, however, requires maximizing the expected likelihood while constraining $\sum_{i=1}^{d} \pi_i = 1$. This derivation requires using solving a constrained optimization problem which is outside the scope of this class.

Remark 9.3. Note that the EM algorithm can be very sensitive to the initialization, and is not guaranteed to converge to the same solution from any initialization.