Gaussian Mixture Models

Author:
Esther Ling

December 10, 2017

1 Overview of Mixture Models

A mixture model is one where we "mix" a sum of different component pdfs, each weighted by a scalar. The weights are to designate how much of each pdf is present in the mixture. This is equivalent to taking convex combinations of component pdfs.

$$f_x(x) = \sum_{q=1}^{Q} \beta_q \phi_q(x) \tag{1}$$

where $\beta_q >= 0$ is a scalar, and $\int_{x \in R^D} \phi_q(x) dx = 1$

If we are familiar with basis expansions, where we approximate a function using a linear combination of basis functions, then this formulation should not surprise us.

There is however another way to look at mixture models. We can think of it as random variables being controlled by latent "states". Suppose there are Q different "modes" in which the random variable might operate in, each with a different probability β_q, with ϕ_q specifying the behaviour in each mode. Then, we can interpret the mixture model as a combination of behaviours, each weighted to reflect each mode's contribution.

In general, a realization of a mixture model can be generated as follows:

1. Generate a state q $\in$ {1,...,Q},
2. Draw X using the pdf $\phi_q(X)$ of that state.

Mixture models are generally used for clustering our data-points $\{x_1, ..., x_N\}$ together. In order to estimate the paramters, we employ Maximum Likelihood Estimation (MLE), which maximizes the likelihood that a particular mixture model occurs, given a set of parameters $\{\beta_1, ..., \beta_Q, \theta_1, ..., \theta_Q\}$, where θ_q is the set of parameters for the pdf of a particular state. We will see MLE in action in the following section.

2 Gaussian Mixture Models

Suppose the pdf for our model is Gaussian:

$$\phi_q(x) = \phi(x; \mu_q, \Sigma_q)$$

where $\mu_q \in R^D$ is the mean, and $\Sigma_q \in S^D_{++}$ is the covariance matrix.

In this case, each state is governed by its mean and covariance matrix.

2.1 MLE Formulation

Let our parameters set be $\theta = \{\beta_1, ..., \beta_Q, \mu_1, ..., \mu_Q, \Sigma_1, ..., \Sigma_Q\}$. Notice the numbers of parameters to be estimated is Q + DQ + D^2Q.

Assuming that our data-points are drawn from an iid process, we can write the likelihood function as follows:

$$L(\theta; x_1, ..., x_N) = \prod_{n=1}^{N} (\sum_{q=1}^{Q} \beta_q \phi(x_n; \mu_q, \Sigma_q)) \tag{2}$$

Taking the log-likehood:

$$\begin{aligned} l(\theta; x_1, ..., x_N) &= \sum_{n=1}^{N} \log(\sum_{q=1}^{Q} \beta_q \phi(x_n; \mu_q, \Sigma_q)) \\ &= \sum_{n=1}^{N} (\log \beta_q + \log \phi(x_n; \mu_q, \Sigma_q)) \end{aligned} \tag{3}$$

2.2 Problem

The above formulation in (3) sets up MLE nicely, however, there are a few problems. First, there is a sheer number of parameters to estimate, second, there is no closed form solution, and third, this is non-concave.

2.3 An Idea

There is an idea, nevertheless. What if we know what the state variables are, i.e., that our particular sample x_n has state S_n, and $S_n \in \{S_1, ..., S_Q\}$. To illustrate, consider the MNIST classification dataset. Suppose we knew that a particular state belongs to number "0', while another belongs to '1', and so on for all Q=9 states.

In this case, we can re-formulate the log-likelihood function. Suppose state q = S_n, then, (3) can be re-cast as follows:

$$l(\theta; x_1, ..., x_N; S_1, ..., S_N) = \sum_{n=1}^{N} \log(\beta_{S_n}) + \log(\phi(x_n; \mu_{S_n}, \Sigma_{S_n}) \qquad (4)$$

Notice how the sum over Q has disappeared.
The solution to this problem is intuitive, where:

$$\hat{\beta}_q = \frac{|I_q|}{N}$$
$$\hat{\mu_q} = samplemean for state_q$$
$$\hat{\sigma_q} = samplecovariance for state_q$$

Unfortunately, we do not get to observe the state, so this does not work. Nevertheless, it gives us an insight into how we might solve the problem, which leads us to the Expectation-Maximization Algorithm.

2.4 Expectation-Maximization Algorithm

To make up for our lack of knowledge of the state, we can instead start with an initial guess for a set of parameters, and then compute a likelihood function that is averaged, by taking the Expectation across all possible state realizations. In doing so, we will introduce a variable $\gamma_{n,q}$ that is the probability that we are in state q, given that we observed sample x_n.

The act of repeatedly taking the Expectation, and then Maximizing the log-likelihood function is where we get the name Expectation-Maximization (EM) algorithm. This algorithm has no guarantee of finding the global maximum, but tends to work well in practice, even when finding local minima.

3 Demonstration

Homework 10

4 Conclusion