Clustering and Mixture Models

Read Chapter 9 in the text by Bishop

Unsupervised Learning, Clustering, and Mixture Distributions

Recall that *unsupervised learning* does not focus on predicting anything in particular, but rather tries to find "interesting" aspects of the data.

One possible informal objective is to find *clusters* of similar items. One possible formal objective is modeling the *probability distribution* of all observed variables.

It can be useful to model a probability distribution in which variables are dependent using *latent* (also called *hidden*) variables to express some or all of the dependencies.

When there is one discrete latent variable, the model will express the distribution as a *mixture* of distributions. The latent variable can also be seen as identifing the cluster an item belongs to.

Example: We have data on symptoms of patients (body temperature, blood pressure, etc.). We could cluster the patients, hoping the clusters will correspond to "diseases". We could model the distribution of symptoms using a discrete latent variable, hoping it will represent the disease a patient has. The distribution of symptoms will be a mixture of distributions for each disease.

K-Means Clustering

Suppose we have data $x_1, \ldots, x_N$, with each x_i being a vector of D variables.

We aim to divide the data into K clusters of similar items, measuring similarity by Euclidean distance (perhaps rescaling variables to all have standard deviation one).

The *K-means algorithm* iteratively finds K centres for the clusters, and assigns each item to the cluster whose centre it is nearest to:

1) Initialize $\mu_1, \ldots, \mu_K$ somehow (eg, set them to randomly selected data items).

2) Repeat the following:

   a) For $i = 1, \ldots, N$, assign data item i to the cluster, k, for which $||x_i - \mu_k||$ is smallest. (Prefer the current assignment if it is tied for the best.)

   b) For $k = 1, \ldots, K$, set μ_k to the mean of data items assigned to cluster k.

   until there is no change in the cluster assignments from the previous iteration.

It's easy to see that this process converges, since each step reduces the value of

$$J \quad = \quad \sum_{i=1}^{N} ||x_i - \mu_{c_i}||^2$$

where c_i is the cluster assigned to data item i. The algorithm may not find the global minimum of J, however.

From Clustering to Mixture Models

K-means clustering assigns a definite cluster to each data item. But often this is unrealistic — eg, some patients have combinations of the symptoms we have measured that are consistent with more than one disease.

The clusters found by the K-means algorithm are described only by the mean of the data items in the cluster. We might like a more complete description of what items in a cluster are like.

Both of these issues can be addressed by modeling the data as coming from a *mixture* distribution, with mixture components corresponding to clusters.

Mixture Distributions

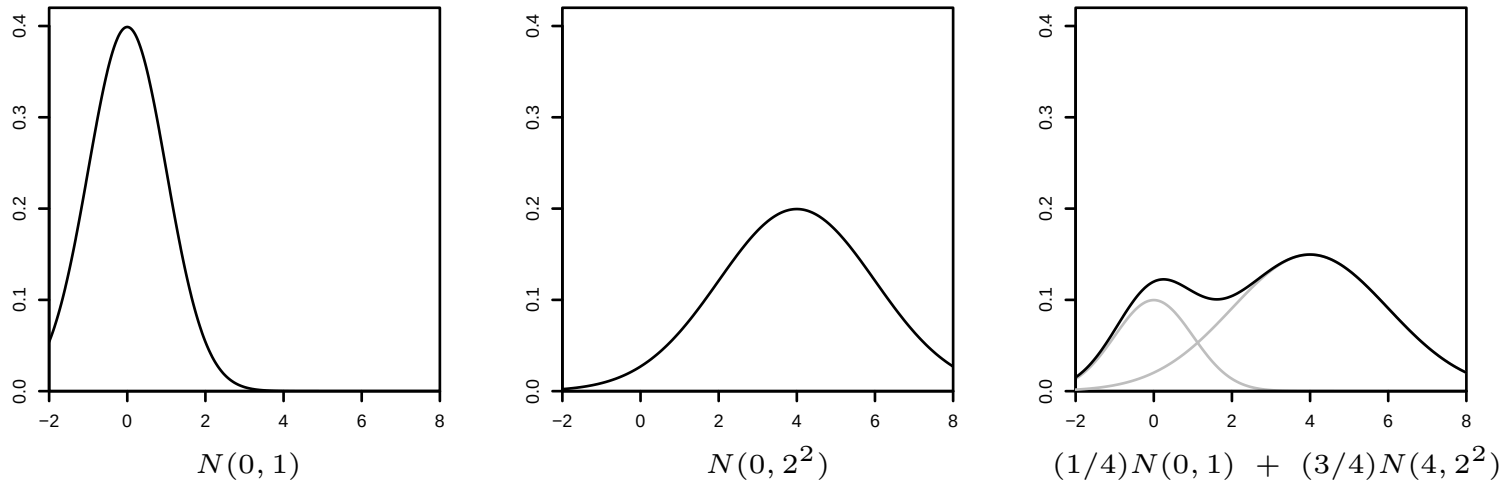
K distributions with density functions $f_1(x), \ldots, f_K(x)$ can be mixed in proportions $\pi_1, \ldots, \pi_K$ to give a mixture distribution with density function

$$f(x) \quad = \quad \sum_{k=1}^{K} \pi_k \, f_k(x)$$

For example, when x is one dimensional, we can mix $N(0, 1^2)$ and $N(4, 2^2)$ with proportions $1/4$ and $3/4$, giving the density function

$$f(x) \quad = \quad \frac{1}{4} \frac{1}{\sqrt{2\pi}} \exp(-(1/2)x^2) \; + \; \frac{3}{4} \frac{1}{2\sqrt{2\pi}} \exp(-(1/2)(x-4)^2/2^2)$$

as pictured below:



Gaussian Mixture Models

When data items are real vectors, it may be reasonable to model the data as a mixture of Gaussian distributions, using the data to estimate both the mixing proportion and the mean vector and covariance matrix of each component distribution.

This model, with K components, can be written as

$$P(x) \quad = \quad \sum_{k=1}^{K} \pi_k \, N(x|\mu_k, \Sigma_k)$$

We might allow Σ_k to be any valid covariance matrix, or we might constrain Σ_k to be a diagonal matrix. If the Σ_k are diagonal, *all* the dependence between the variables in x is a consequence of the distribution being a mixture.

A natural idea is to estimate π_k, μ_k, and Σ_k for $k = 1, \ldots, K$ by maximizing the likelihood. Assuming that data items are independent, the log likelihood is

$$\sum_{i=1}^{N} \log P(x_i)$$

where x_i is the data vector for item i, and $P(\,\cdot\,)$ is as defined above.

Issues with Maximum Likelihood for Gaussian Mixture Models

Non-identifiability: The global maximum of the likelihood for a mixture model is not unique, since permuting the labels of the mixture components will produce a different set of parameter values that fits the data just as well.

Other local maxima: Even aside from re-labellings, there is often more than one local maximum of the likelihood. Finding one of the global maxima (or at least a good local maximum) may require searching for the maximum from many different starting points.

We don't want the global maximum anyway: When $K > 1$, the actual global maximum will be at a point with infinite likelihood, in which for some component, k, and some data item, i, we have $0 < \pi_k < 1$, $\mu_k = x_i$, and $\Sigma_k = 0$. This gives an infinite spike of probability density at one point, while other points have non-zero probability density from other components.

Because of this problem, we need to try as many starting points as needed to find a good local maximum that *isn't* one where some $\Sigma_k \to 0$.

The EM Algorithm for Gaussian Mixture Models

We could use some general purpose optimization method (eg, gradient descent) to find the parmeters of a mixture model that maximize the likelihood (avoiding the singular solutions with $\Sigma_k \to 0$). But a method known as the *EM algorithm* is commonly used, because it is simple to implement, and very stable. (Though it can unfortunately also be rather slow.)

The idea: If we knew which mixture component each data item came from, estimating the mixing proportions and the parameters of each component distribution would be easy. We don't know this, but given an initial guess at the parameters, we can probabilistically assign a component to each data item, and then get a better estimate of the parameters based on these assignments.

This is sort of like the K-means algorithm, but in a probabilistic setting, with a proof that the algorithm will reach a (local) maximum of the likelihood.

Details of the EM Algorthim for Gaussian Mixture Models

Here are the details of the EM algorithm for a Gaussian mixture model with Σ_k being diagonal, with diagonal elements of σ_{kj}^2.

The algorithm alternates between "E" steps and "M" steps:

E Step: Using the current values of the parameters, compute the "responsibilities" of components for data items, by applying Bayes' Rule:

$$r_{ik} \;=\; P(\text{data item } i \text{ came from component } k \,|\, x_i) \;=\; \frac{\pi_k \, N(x_i|\mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \, N(x_i|\mu_{k'}, \Sigma_{k'})}$$

M Step: Using the current responsibilities, re-estimate the parameters, using weighted averages, with weights given by the responsibilities:

$$\pi_k \;=\; \frac{1}{N} \sum_i r_{ik}, \quad \mu_k \;=\; \sum_i r_{ik} x_i \Big/ \sum_i r_{ik}, \quad \sigma_k^2 \;=\; \sum_i r_{ik} (x_i - \mu_k)^2 \Big/ \sum_i r_{ik}$$

We start with some initial guess at the parameter values (perhaps random), or perhaps with some initial guess at the responsibilities (in which case we start with an M step). We continue alternating E and M steps until there is little change.