Singapore University of Technology and Design
Information Systems Technology and Design
50.007 Machine Learning

Mixture Models

Probability models and mixtures

The idea of generative models is that we specify a mechanism by which we can generate (sample) points such as those given in the training data. This is a powerful idea and allows us to automatically discover many hidden mechanism that underly the data. We will start with simple *mixture models* which assume a two-stage generative process: first we select which type of point to generate (e.g., which cluster the point belongs to), and then we generate a point from the corresponding model (cluster). You can think of mixture models as probabilistic extensions of k-means clustering. However, the idea is more general and flexible than K-means.

Spherical Gaussian

Let's start with a simple model for one cluster such as a Gaussian. In other words, we assume that points $x \in \mathcal{R}^d$ are generated as samples from the following *spherical* Gaussian distribution:

$$P(x|\mu, \sigma^2) = N(x; \mu, \sigma^2 I) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp(-\frac{1}{2\sigma^2}\|x-\mu\|^2) \tag{1}$$

where d is the dimension. This is a simple spherically symmetric distribution around the mean (centroid) μ. The probability of generating points away from the mean μ decreases the same way (based on the squared distance) regardless of the direction. A typical representation of this distribution is by a circle centered at the mean μ with radius σ (called the standard deviation). See Figure 1 below. σ^2 is the average squared variation of coordinates of x from the coordinates of the mean μ (see below). The two parameters, μ and σ, summarize how the data points in the cluster are expected to vary.

Given a training set of points $S_n = \{x^{(t)}, t = 1, \ldots, n\}$, we can estimate the parameters of the Gaussian distribution to best match the data. Note that we can do this regardless of what the points look like (there may be several clusters or just one). We are simply asking what is the best Gaussian that fits the data. The criterion we use is *maximum likelihood* or ML for short. We evaluate the probability of generating all the data points, each one independently. This means that the likelihood of the training data is a product

$$L(S_n; \mu, \sigma^2) = \prod_{t=1}^{n} P(x^{(t)}|\mu, \sigma^2) \tag{2}$$

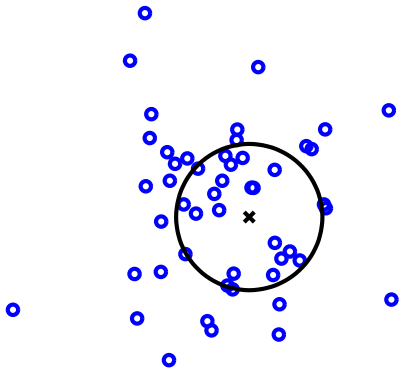

Figure 1: Samples from a spherical Gaussian distribution and the corresponding representation in terms of the mean (center) and the standard deviation (radius).

Since the training data S_n is fixed (given), we view this as a function of the parameters μ and σ^2. The higher the value of $L(S_n; \mu, \sigma^2)$, the better the Gaussian with those parameters fits the data.

To maximize the likelihood, it is convenient to maximize the log-likelihood instead. In other words, we maximize

$$
\begin{aligned}
l(S_n; \mu, \sigma^2) &= \sum_{t=1}^{n} \log P(x^{(t)} | \mu, \sigma^2) && (3) \\
&= \sum_{t=1}^{n} \left[-\frac{d}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|x^{(t)} - \mu\|^2 \right] && (4) \\
&= -\frac{nd}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{t=1}^{n} \|x^{(t)} - \mu\|^2 && (5)
\end{aligned}
$$

By setting $\partial/\partial\mu \; l(S_n; \mu, \sigma^2) = 0$, $\partial/\partial\sigma \; l(S_n; \mu, \sigma^2) = 0$, and solving for the parameters, we obtain the ML parameter estimates

$$
\hat{\mu} = \frac{1}{n} \sum_{t=1}^{n} x^{(t)}, \quad \hat{\sigma}^2 = \frac{1}{dn} \sum_{t=1}^{n} \|x^{(t)} - \hat{\mu}\|^2 \qquad (6)
$$

In other words, the mean μ is simply the sample mean (cf. the choice of centroid for k-means), and σ^2 is the average squared deviation from the mean, averaged across the points and across the d-dimensions (because we use the same σ^2 for each dimension).

A mixture of spherical Gaussians

When the data are best described by multiple clusters, we need to specify multiple Gaussians, one for each cluster. Assuming there are exactly k clusters (this is our

hypothesis, not necessarily true) we would define

$$P(x|\mu^{(i)}, \sigma_i^2), \quad i = 1, \ldots, k \tag{7}$$

and have to somehow estimate $\mu^{(1)}, \ldots, \mu^{(k)}$, $\sigma_1^2, \ldots, \sigma_k^2$ without knowing a priori where the clusters are, i.e., which points belong to which cluster. Since the clusters may vary by size as well, we also include parameters $p_1, \ldots, p_k$ that specify the frequency of points we would expect to see in each cluster. Note that k-means clustering did not include either different spreads (different σ_i^2's) nor different a priori sizes (different p_i's).

So how do we generate data points from the mixture? We first sample index i to see which cluster we should use. In other words, we sample i from a multinomial distribution governed by $p_1, \ldots, p_k$, where $\sum_{i=1}^k p_i = 1$. Think of throwing a biased k-faced die. Larger p_i means that we generate more points from that clusters. Once we know the cluster, we can sample x from the corresponding Gaussian. More precisely,

$$i \sim \text{Multinomial}(p_1, \ldots, p_k) \tag{8}$$

$$x \sim P(x|\mu^{(i)}, \sigma_i^2) \tag{9}$$

Figure 2 below shows data generated from the mixture model with colors identifying the clusters. We have also drawn the corresponding Gaussians, one for each cluster, as well as the prior frequencies[1] (a.k.a. mixing proportions) p_i

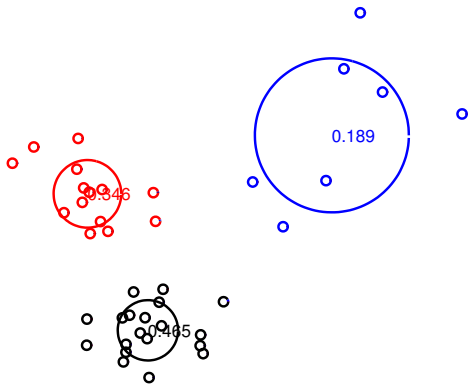

Figure 2: Samples from a mixture of Gaussian distributions with colors indicating the sampled cluster labels. The Figure also shows the corresponding Gaussian cluster models, including the prior cluster frequencies as numbers.

The data we have do not typically come with labels identifying the clusters. Indeed, the main use of mixture models (as in clustering) is to try to uncover these hidden labels, i.e., find the underlying clusters. To this end, we must evaluate the probability that each

[1]For this figure, the prior frequencies appear to match exactly the cluster sizes. This is not true in general, only on average, as the labels are sampled.

data point x could come as a sample from our mixture model, and adjust the model parameters so as to increase this probability. Each x could have been generated from any cluster, just with different probabilities. So, to evaluate $P(x|\theta)$, where θ specifies all the parameters in our mixture model

$$\theta = \{\mu^{(1)}, \ldots, \mu^{(k)}, \sigma_1^2, \ldots, \sigma_k^2, p_1, \ldots, p_k\}, \tag{10}$$

we must sum over all the alternative ways we could have generated x (all the ways that Eq.(9) could have resulted in x). In other words,

$$P(x|\theta) = \sum_{i=1}^{k} p_i P(x|\mu^{(i)}, \sigma_i^2) \tag{11}$$

This is the mixture model we must estimate from data $S_n = \{x^{(t)}, t = 1, \ldots, n\}$. It is not easy to resolve where to place the clusters, and how they should be shaped. We'll start with a simpler problem of estimating the mixture from labeled points. Then generalize the solution to estimated mixtures from S_n alone.

Estimating mixtures: labeled case

If our data points came labeled, i.e., each point would be assigned to a single cluster, we could estimate our Gaussian models as before. In addition, we could evaluate the cluster sizes just based on the actual numbers of points. For later utility, let's expand on this a bit. Let $\delta(i|t)$ be an indicator that tells us whether $x^{(t)}$ should be assigned to cluster i. In other words,

$$\delta(i|t) = \begin{cases} 1, \text{ if } x^{(t)} \text{ is assigned to } i \\ 0, \text{ otherwise} \end{cases} \tag{12}$$

Using this notation, our maximum likelihood objective is

$$\sum_{t=1}^{n} \left[\sum_{i=1}^{k} \delta(i|t) \log(P(x^{(t)}|\mu^{(i)}, \sigma_i^2) p_i) \right] = \sum_{i=1}^{k} \left[\sum_{t=1}^{n} \delta(i|t) \log(P(x^{(t)}|\mu^{(i)}, \sigma_i^2) p_i) \right] \tag{13}$$

where, in the first expression, the inner summation over clusters simply selects the Gaussian that we should use to generate the corresponding data point, consistent with the assignments. In the second expression, we exchanged the summations to demonstrate that the Gaussians can be solved separately from each other, as in the single Gaussian case. Note that we also include p_i in generating each point, i.e., the probability that we would select this cluster from the mixture model for this point. The ML solution based

on labeled points is given by

$$\hat{n}_i = \sum_{t=1}^{n} \delta(i|t) \text{ (number of points assigned to cluster } i) \tag{14}$$

$$\hat{p}_i = \frac{\hat{n}_i}{n} \text{ (fraction of points in cluster } i) \tag{15}$$

$$\hat{\mu}^{(i)} = \frac{1}{\hat{n}_i} \sum_{t=1}^{n} \delta(i|t) x^{(t)} \text{ (mean of points in cluster } i) \tag{16}$$

$$\hat{\sigma}_i^2 = \frac{1}{d\hat{n}_i} \sum_{t=1}^{n} \delta(i|t) \|x^{(t)} - \hat{\mu}^{(i)}\|^2 \text{ (mean squared spread in cluster } i) \tag{17}$$

Estimating mixtures: the EM-algorithm

Our goal is to maximize the likelihood that the data was generated by a mixture model. In other words, on a log-scale, we try to maximize

$$l(S_n; \theta) = \sum_{t=1}^{n} \log P(x^{(t)}|\theta) = \sum_{t=1}^{n} \log \left(\sum_{i=1}^{k} p_i P(x^{(t)}|\mu^{(i)}, \sigma_i^2) \right) \tag{18}$$

with respect to the parameters θ. Unfortunately, the summation inside the logarithm makes this a bit nasty to optimize. More intuitively, it is clearly hard to entertain different arrangements of k Gaussians that best explain the data.

Our solution is an iterative algorithm known as the *Expectation-Maximization algorithm* or the EM-algorithm for short. The trick we use is to return the problem back to the simple labeled case. In other words, we can use the current mixture model to assign examples to clusters (see below), then re-estimate each cluster model separately based on the points assigned to it, just as in the labeled case. Since the assignments were based on the current model, and the model was just improved by re-estimating the Gaussians, the assignments would potentially change as well. The algorithm is therefore necessarily iterative. The setup is very analogous to k-means. However, here we cannot fully assign each example to a single cluster. We have to entertain the possibility that the points were generated by different cluster models. We will make soft assignments, based on the relative probabilities that each cluster explains (can generate) the point.

We need to first initialize the mixture parameters. For example, we could initialize the means $\mu^{(1)}, \ldots, \mu^{(k)}$ as in the k-means algorithm, and set the variances σ_i^2 all equal to the overall data variances:

$$\hat{\sigma}^2 = \frac{1}{dn} \sum_{t=1}^{n} \|x^{(t)} - \hat{\mu}\|^2 \tag{19}$$

where $\hat{\mu}$ is the mean of all the data points. This ensures that the Gaussians can all "see" all the data points (spread is large enough) that we do not a priori assign points

to specific clusters too strongly. Since we have no information about the cluster sizes, we will set $p_i = 1/k$, $i = 1, \ldots, k$.

The EM-algorithm is then defined by the following two steps.

- **E-step:** softly assign points to clusters according to the posterior probabilities

$$p(i|t) = \frac{p_i P(x|\mu^{(i)}, \sigma_i^2)}{P(x|\theta)} = \frac{p_i P(x|\mu^{(i)}, \sigma_i^2)}{\sum_{j=1}^k p_j P(x|\mu^{(j)}, \sigma_j^2)} \tag{20}$$

  Here $\sum_{i=1}^k p(i|t) = 1$. These are exactly analogous to (but soft versions of) $\delta(i|t)$ in the labeled case. Each point $x^{(t)}$ is assigned to cluster i with weight $p(i|t)$. The larger this weight, the more strongly we require cluster i to generate this point in the M-step below.

- **M-step:** Once we have $p(i|t)$, we pretend that we were given these assignments (as softly labeled examples) and can use them to estimate the Gaussians separately, just as in the labeled case.

$$\hat{n}_i = \sum_{t=1}^n p(i|t) \quad \text{(effective number of points assigned to cluster } i\text{)} \tag{21}$$

$$\hat{p}_i = \frac{\hat{n}_i}{n} \quad \text{(fraction of points in cluster } i\text{)} \tag{22}$$

$$\hat{\mu}^{(i)} = \frac{1}{\hat{n}_i} \sum_{t=1}^n p(i|t) x^{(t)} \quad \text{(weighted mean of points in cluster } i\text{)} \tag{23}$$

$$\hat{\sigma}_i^2 = \frac{1}{d\hat{n}_i} \sum_{t=1}^n p(i|t) \|x^{(t)} - \hat{\mu}^{(i)}\|^2 \quad \text{(weighted mean squared spread)} \tag{24}$$

  We will then use these parameters in the E-step, and iterate.

This simple algorithm is guaranteed to monotonically increase the log-likelihood of the data under the mixture model (cf. k-means). Just as in k-means, however, it may only find a locally optimal solution. But it is less "brittle" than k-means due to the soft assignments. Figure 3 below shows an example of running a few steps of the EM-algorithm. The points are colored based on the soft assignments in the E-step. Initially, many of the points are assigned to multiple clusters. The assignments are clarified (mostly in one cluster) as the algorithm finds where the clusters are.

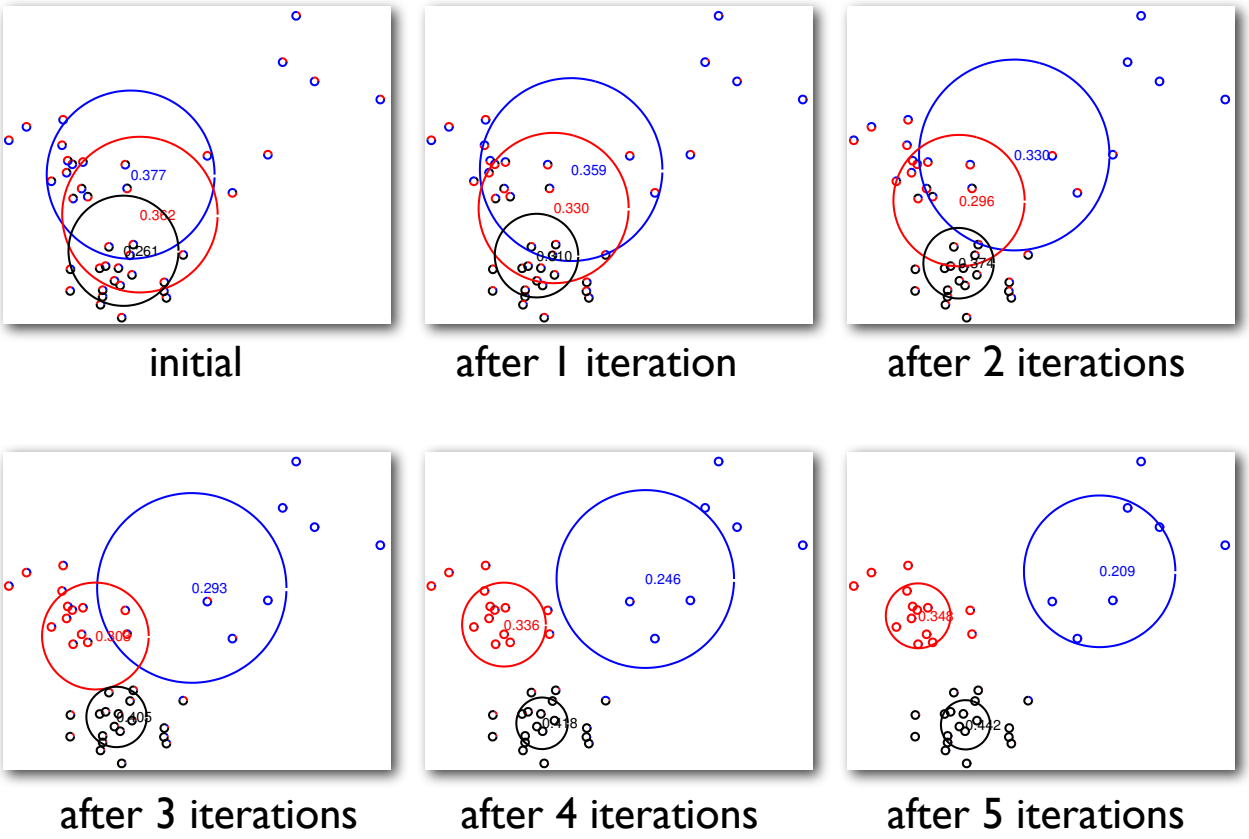

Figure 3: An example of running the EM-algorithm for five iterations