A Probabilistic View of Unsupervised Learning

One interpretation of the goal of learning "interesting" properties of a dataset is that we should learn those things that help us model the probability (density) of the data items.

K-means flat clustering and average-linkage hierarchical clustering are non-probabilistic algorithms — based just on the intuitive notion of clustering together "similar" cases.

Instead, we can use a probabilistic model based on *latent variables* that describe the clustering.

For *mixture models*, the latent variables are cluster indicators.

Mixture Models for Clustering

Suppose we have univariate or multivariate training cases $y^{(1)}, \ldots, y^{(n)}$.

We may think that this data form several clusters, perhaps because it comes from a "mixture" of several sources.

Example: We capture n beetles. The weight of the i'th beetle is $y^{(i)}$. If there are K species of beetles, we may expect K clusters.

If we assume that the data form K clusters (ie, come from K sources) the probability density for a data point, y, can be written as

$$P(y) \;=\; \sum_{k=1}^{K} \rho_k \, g_k(y)$$

where ρ_k is the probability of a data point coming from cluster k, and $g_k(y)$ is the probability (density) for data from cluster k.

This makes sense only if the component distributions, given by g_k, are relatively simple — we build a more complex distribution by combining several of them.

Latent Variable Models

This mixture distribution can also be expressed using latent variables, $z^{(1)}, \ldots, z^{(n)}$, one for each data point, $y^{(1)}, \ldots, y^{(n)}$.

The value of $z^{(i)}$ is an integer from 1 to K identifying which cluster $y^{(i)}$ is in. So

$$P(z^{(i)} = k) \;=\; \rho_k$$

If we know which cluster $y^{(i)}$ is in, it's distribution is just the distribution for that cluster. So

$$p(y^{(i)} \,|\, z^{(i)} = k) \;=\; g_k(y^{(i)})$$

This produces the mixture we want as the marginal density for $y^{(i)}$:

$$p(y^{(i)}) \;=\; \sum_{k=1}^{K} P(z^{(i)} = k)\, p(y^{(i)} \,|\, z^{(i)} = k) \;=\; \sum_{k=1}^{K} \rho_k \, g_k(y^{(i)})$$

Mixture Models for Binary Data

If $y^{(i)} = y^{(i)}_1, \ldots, y^{(i)}_p$, with each $y^{(i)}_j$ being binary, we might decide that these p variables are independent *given* the mixture component, z_i, that they come from. Of course, they *aren't* independent overall — the whole point of the model is to express the dependencies by means of the latent variables, $z^{(i)}$.

In other words, we can write the component distributions as

$$P(y^{(i)} \,|\, z^{(i)} = k) \;=\; g_k(y^{(i)}) \;=\; \prod_{j=1}^{p} \theta_{k,j}^{y^{(i)}_j} \, (1 - \theta_{k,j})^{1 - y^{(i)}_j}$$

where $\theta_{k,j}$ is the probability that $y^{(i)}_j$ will be 1 if training case i has $z^{(i)} = k$.

Examples of such data:

> Students' answers to test questions, marked as correct or incorrect. Here, $z^{(i)}$ might indicate what sort of background the student has, together with other characteristics.
>
> Descriptions of documents, as to whether or not they contain certain words. Here, $z^{(i)}$ might indicate what the topic of the document is — eg, sports, politics, science, etc.

Gaussian Mixture Models

If the data is real-valued, we might use multivariate Gaussian distributions as mixture components.

We might allow any sort of Gaussian distribution — still much simpler than an arbitrary distribution. However, we might instead require variables be independent within each component — ie, that the Gaussian component distributions have diagonal covariance matrices.

If variables are independent within a component, we can write

$$g_k(y|\mu_k, \sigma_k) \;=\; \prod_{j=1}^{p} (2\pi\sigma_{k,j}^2)^{-1/2} \exp\Big(-(y_j-\mu_{k,j})^2/2\sigma_{k,j}^2\Big)$$

Here, the model parameters are the mixing proportions, $\rho_1, \ldots, \rho_K$, the mean vectors for the mixture components, $\mu_1, \ldots, \mu_k$, and the vectors of standard deviations for the mixture components, $\sigma_1, \ldots, \sigma_k$.

The EM Algorithm for Mixture Models

We can estimate the parameters of a mixture model by maximum likelihood. This can be done using the "Expectation-Maximization" (EM) algorithm, which for Gaussian mixture models resembles a "soft" form of K-means clustering.

Starting with some initial guesses, $\hat\rho_1, \ldots, \hat\rho_K$, $\hat\mu_1, \ldots, \hat\mu_K$, and $\hat\sigma_1, \ldots, \hat\sigma_K$, repeatedly do the following, until the estimates have converged:

E Step: Compute the "responsibilities" of each cluster for each training case, using the current parameters:

$$r_k^{(i)} \;=\; \hat\rho_k\, g_k(y^{(i)}|\hat\mu_k, \hat\sigma_k) \Big/ \sum_{k'=1}^{K} \hat\rho_{k'}\, g_{k'}(y^{(i)}|\hat\mu_{k'}, \hat\sigma_{k'})$$

M Step: Re-estimate the parameters based on the fractions of each data point in each cluster:

$$\hat\rho_k \;=\; (1/n) \sum_{i=1}^{n} r_k^{(i)}$$

$$\hat\mu_{k,j} \;=\; \sum_{i=1}^{n} r_k^{(i)}\, y_j^{(i)} \Big/ \sum_{i=1}^{n} r_k^{(i)}$$

$$\hat\sigma_{k,j}^2 \;=\; \sum_{i=1}^{n} r_k^{(i)}\, (y_j^{(i)} - \hat\mu_{k,j})^2 \Big/ \sum_{i=1}^{n} r_k^{(i)}$$

Advantages and Disadvantages of Maximum Likelihood via EM for Mixture Models

Advantages:

- Easy to implement.
- No "learning rate" or "stepsize" parameters to fiddle with.
- Guaranteed to increase the likelihood (or leave it unchanged) every iteration.

Disadvantages:

- EM can take many iterations to get close a local maximum.
- EM can get stuck in a local maximum that isn't good.
- The global maximum of the likelihood will have infinite likelihood, with one mixture component modeling a single case "perfectly" — ie, with $\mu_{k,j} = x_j^{(i)}$ and $\sigma_{k,j} = 0$. This is of course drastic overfitting.

  Fortunately, EM usually doesn't find this global maximum, but instead finds a local maximum that is better!
- Overfitting is possible even if the infinite global maximum is avoided, however.