The EM Algorithm for Gaussian Mixtures

Probabilistic Learning: Theory and Algorithms, CS 274A

Finite Mixture Models

We are given a data set $D = \{\underline{x}_1, \ldots, \underline{x}_N\}$ where $\underline{x}_i$ is a d-dimensional vector measurement. Assume that the points are generated in an IID fashion from an underlying density $p(\underline{x})$. We further assume that $p(\underline{x})$ is defined as a finite mixture model with K components:

$$p(\underline{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(\underline{x}|z_k, \theta_k)$$

where:

- The $p_k(\underline{x}|z_k, \theta_k)$ are *mixture components*, $1 \leq k \leq K$. Each is a density or distribution defined over $p(\underline{x})$, with parameters θ_k.
- $z = (z_1, \ldots, z_K)$ is a vector of K binary indicator variables that are mutually exclusive and exhaustive (i.e., one and only one of the z_k's is equal to 1, and the others are 0). z is a K-ary random variable representing the identity of the mixture component that generated $\underline{x}$. It is convenient for mixture models to represent z as a vector of K indicator variables.
- The $\alpha_k = p(z_k)$ are the mixture weights, representing the probability that a randomly selected $\underline{x}$ was generated by component k, where $\sum_{k=1}^{K} \alpha_k = 1$.

The complete set of parameters for a mixture model with K components is

$$\Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}$$

Membership Weights

We can compute the "membership weight" of data point $\underline{x}_i$ in cluster k, given parameters Θ as

$$w_{ik} \;=\; p(z_{ik} = 1|\underline{x}_i, \Theta) \;=\; \frac{p_k(\underline{x}_i|z_k, \theta_k) \cdot \alpha_k}{\sum_{m=1}^{K} p_m(\underline{x}_i|z_m, \theta_m) \cdot \alpha_m}, \quad 1 \leq k \leq K, \quad 1 \leq i \leq N.$$

This follows from a direct application of Bayes rule.

The membership weights above reflect our uncertainty, given $\underline{x}_i$ and Θ, about which of the K components generated vector $\underline{x}_i$. Note that we are assuming in our generative mixture model that each $\underline{x}_i$ was generated by a single component—so these probabilities reflect our uncertainty about which component $\underline{x}_i$ came from, not any "mixing" in the generative process.

Gaussian Mixture Models

For $\underline{x} \in \mathcal{R}^d$ we can define a Gaussian mixture model by making each of the K components a Gaussian density with parameters $\underline{\mu}_k$ and Σ_k. Each component is a multivariate Gaussian density

$$p_k(\underline{x}|\theta_k) \;=\; \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^t \Sigma_k^{-1}(\underline{x}-\underline{\mu}_k)}$$

with its own parameters $\theta_k = \{\underline{\mu}_k, \Sigma_k\}$.

The EM Algorithm for Gaussian Mixture Models

We define the EM (Expectation-Maximization) algorithm for Gaussian mixtures as follows. The algorithm is an iterative algorithm that starts from some initial estimate of Θ (e.g., random), and then proceeds to iteratively update Θ until convergence is detected. Each iteration consists of an E-step and an M-step.

E-Step: Denote the current parameter values as Θ. Compute w_{ik} (using the equation above for membership weights) for all data points $\underline{x}_i, 1 \leq i \leq N$ and all mixture components $1 \leq k \leq K$. Note that for each data point $\underline{x}_i$ the membership weights are defined such that $\sum_{k=1}^{K} w_{ik} = 1$. This yields an $N \times K$ matrix of membership weights, where each of the rows sum to 1.

M-Step: Now use the membership weights and the data to calculate new parameter values. Let $N_k = \sum_{i=1}^{N} w_{ik}$, i.e., the sum of the membership weights for the kth component—this is the effective number of data points assigned to component k.

Specifically,

$$\alpha_k^{new} = \frac{N_k}{N}, \quad 1 \leq k \leq K.$$

These are the new mixture weights.

$$\underline{\mu}_k^{new} = \left(\frac{1}{N_k}\right) \sum_{i=1}^{N} w_{ik} \cdot \underline{x}_i \quad 1 \leq k \leq K.$$

The updated mean is calculated in a manner similar to how we could compute a standard empirical average, except that the ith data vector $\underline{x}_i$ has a fractional weight w_{ik}. Note that this is a vector equation since $\underline{\mu}_k^{new}$ and $\underline{x}_i$ are both d-dimensional vectors.

$$\Sigma_k^{new} = \left(\frac{1}{N_k}\right) \sum_{i=1}^{N} w_{ik} \cdot (\underline{x}_i - \underline{\mu}_k^{new})(\underline{x}_i - \underline{\mu}_k^{new})^t \quad 1 \leq k \leq K.$$

Again we get an equation that is similar in form to how we would normally compute an empirical covariance matrix, except that the contribution of each data point is weighted by w_{ik}. Note that this is a matrix equation of dimensionality $d \times d$ on each side.

The equations in the M-step need to be computed in this order, i.e., first compute the K new α's, then the K new $\underline{\mu}_k$'s, and finally the K new Σ_k's.

After we have computed all of the new parameters, the M-step is complete and we can now go back and recompute the membership weights in the E-step, then recompute the parameters again in the E-step, and continue updating the parameters in this manner. Each pair of E and M steps is considered to be one iteration.

Initialization and Convergence Issues for EM

The EM algorithm can be started by either initializing the algorithm with a set of initial parameters and then conducting an E-step, or by starting with a set of initial weights and then doing a first M-step. The initial parameters or weights can be chosen randomly (e.g. select K random data points as initial means and select the covariance matrix of the whole data set for each of the initial K covariance matrices) or could be chosen via some heuristic method (such as by using the k-means algorithm to cluster the data first and then defining weights based on k-means memberships).

Convergence is generally detected by computing the value of the log-likelihood after each iteration and halting when it appears not to be changing in a significant manner from one iteration to the next. Note that the log-likelihood (under the IID assumption) is defined as follows:

$$\log l(\Theta) \;=\; \sum_{i=1}^{N} \log p(\underline{x}_i|\Theta) \;=\; \sum_{i=1}^{N} \left(\log \sum_{k=1}^{K} \alpha_k p_k(\underline{x}_i|z_k, \theta_k) \right)$$

where $p_k(\underline{x}_i|z_k, \theta_k)$ is the Gaussian density for the kth mixture component.

The K-means Algorithm

The K-means algorithm is another algorithm for clustering real-valued data. It is based on minimizing the sum of Euclidean distances between each point and its assigned cluster, rather than on a probabilistic model. The algorithm takes as input an $n \times d$ data matrix (with real-valued entries), a value for K, and operates as follows:

1. Initialize by randomly selecting K mean vectors, e.g., pick K data vectors (rows) randomly from the input data matrix
2. Assign each of the n data vectors to the cluster corresponding to which of the K clusters means it is closest to, where distance is measured as Euclidean distance in the d-dimensional input space.
3. For each cluster k, compute its new mean as the mean (average) of all the data vectors that were assigned to this cluster in Step 2.
4. Check for convergence. An easy way to determine convergence is to execute Step 2 and check if any of the data points change cluster assignments relative to their assignment on the previous iteration. If not, exit; if 1 or more points change cluster assignment, continue to Step 3.

The K-means algorithm can be viewed as a heuristic search algorithm for finding the cluster assignments that minimize the total sum of squares, namely the sum of the squared Euclidean distances from each of the n data points to a cluster center. Finding the optimal solution is NP-hard, so K-means may converge to local minima. For this reason it can be useful to start the algorithm multiple random starting conditions, and select the solution with the minimum sum of squares score over different runs.

The K-means algorithm can also be thought of as a simpler non-probabilistic alternative to Gaussian mixtures. K-means has no explicit notion of cluster covariances. One can "reduce" Gaussian mixture clustering to K-means if one were to (a) fix a priori all the covariances for the K components to be the identity matrix (and not update them during the M-step), and (b) during the E-step, for each data vector, assign a membership probability of 1 for the component it is most likely to belong to, and 0 for all the other memberships (in effect make a "hard decision" on component membership at each iteration).