CS5487 - EM-GMM implementation tips

Antoni Chan
Department of Computer Science
City University of Hong Kong

Oct 11, 2012

The following are some tips on implementing the EM algorithm for Gaussian mixture models (GMMs). The first section discusses initialization strategies, while the second section presents other implementation tips. In the remainder, we will assume the GMM has K components, where the j-th component has parameters $\{\pi_j, \mu_j, \Sigma_j\}$. The set of samples is $X = \{x_1, \cdots, x_n\}$, with $x_i \in \mathbb{R}^d$. Some of these tips can also apply to EM for other models.

1 Initialization Strategies

The initialization of the EM algorithm is very important to avoid bad local minima, and to obtain a good clustering or estimate. One solution is to use the K-means algorithm to form the initial clusters for EM, but obviously K-means also requires some initialization. The following are some typical strategies of initializing EM or Kmeans.

1.1 Random

Select a random data point for each component mean μ_j (or cluster center). For EM, the covariance of each component Σ_j can be initialized proportional to the overall sample covariance of the data. Another way to initialize the covariance is to assign data points to clusters according to K-means, and then use the sample covariance from each resulting cluster.

In general, random initialization is not very robust because it could start near a bad local minimum (e.g., if an outlier point is selected). One way around this is to run several trials of EM with different random initializations, and then select the trial that results in the largest data log-likelihood, $\log p(X)$. For K-means, you would select the trial with smallest total distance to the centers.

1.2 Furthest first

The "furthest first" method selects a deterministic set of points as the component means. The points are selected to be furthest apart from each other, so that the clusters will fully cover the region of the data. Define the distance between two points as $d(x_i, x_j) = \|x_i - x_j\|$. The procedure is as follows:

1. Select the two points that are furthest apart as the first two component means,

$$\{\mu_1, \mu_2\} = \operatorname*{argmax}_{i,j} d(x_i, x_j). \tag{1}$$

2. For $k \in \{3, \cdots, K\}$, select a point that is furthest from the previous component means,

$$\mu_k = \operatorname*{argmax}_{i} \sum_{j=1}^{k-1} d(x_i, \mu_j). \tag{2}$$

1.3 Component splitting

This initialization strategy works by running EM with an increasing number of components. After each run of EM (after convergene), one mixture component is split into two components and perturbed slightly. EM is then run again until convergence, and the process is repeated until the desired number of components is reached. This component splitting method is related to deterministic annealing.

Formally, let K_f be the final number of components desired. The procedure of the component splitting strategy is as follows:

1. Run EM with $K = 1$ to obtain parameters $\Theta = \{\pi_1, \mu_1, \Sigma_1\}$.
2. For $k \in \{2, 3, \cdots, K_f\}$,
   (a) Select a component $j \in \{1, \cdots, k-1\}$ according to some criteria (see below).
   (b) Create a new component k from j, and adjust the priors:

$$\mu_k \leftarrow \mu_j, \quad \Sigma_k \leftarrow \Sigma_j, \quad \pi_k \leftarrow \frac{1}{2}\pi_j, \quad \pi_j \leftarrow \frac{1}{2}\pi_j. \tag{3}$$

   (c) Perturb the parameters of component j and k (see below).
   (d) Run EM with $K = k$ to obtain a new set of parameters Θ.

With this strategy there is some freedom to select the criteria for selecting the splitting component, and perturbing the parameters. These criteria can affect the final clustering and the parameter estimate.

- *Component selection*: There are different methods for selecting the component for splitting, which can influence the final clustering or parameter estimate.
  - Selecting the component with the largest prior tends to form evenly-sized clusters (or at least clusters that are not too large),

$$j = \operatorname{argmax} \pi_j. \tag{4}$$

  - In contrast, selecting the component with the largest variance, tends to form compact clusters that evenly cover the space,

$$j = \operatorname{argmax} |\Sigma_j|. \tag{5}$$

    This could also be modified to only look at the directions of maximum variance for each Σ_j, or the ratio between the largest and smallest eigenvalues (to find skinny Gaussians).

- *Perturbing components*: There are also several methods for perturbing the Gaussian mixture components.
  - The means of the two components can be translated in opposite directions,

$$\mu_j \leftarrow \mu_j + \alpha v, \qquad \mu_k \leftarrow \mu_k - \alpha v, \tag{6}$$

    where v is the direction vector and α a constant. One candidate direction is the direction of greatest variance of the Gaussian component, i.e., the eigenvector with the largest eigenvalue. This will encourage the two components to try to form separate clusters.
  - On the other hand, the covariance of the two components could be scaled in opposite directions, while keeping the means fixed,

$$\Sigma_j \leftarrow \alpha \Sigma_j, \qquad \Sigma_k \leftarrow \frac{1}{\alpha}\Sigma_k, \tag{7}$$

    where α is the scaling constant (usually close to 1). This would encourage EM to better model the density around the mean, e.g., if it is not exactly shaped like a Gaussian.

- *Splitting schedule*: In the above algorithm, one component is added at a time until K_f is reached. However, this "splitting schedule" can be changed so that more than one component is selected after each run of EM. The splitting schedule also influences the final clustering.
  - For example, the most aggressive strategy would be $k \in \{1, 2, 4, 8, \ldots\}$, which splits every component each time. Splitting every component will obtain a good estimate of the distribution of the data, since the components will evenly cover the data.
  - On the other hand, splitting one component at a time, $K \in \{1, 2, 3, 4, \ldots\}$, can obtain a good clustering of the data (e.g., in image segmentation), since the "best" cluster can be selected each time (e.g., the cluster that corresponds to two objects).
- *Other details*: For the intermediate values of K, it is not necessary to run EM to full convergence. Some time can be saved by loosening the convergence criteria.

2 Implementation Tips

The following are some other implementation tips.

- *Covariance matrices*: Sometimes it is helpful to restrict the covariance of the multivariate Gaussian component to prevent overfitting. Instead of a full covariance matrix, an iid or a diagonal covariance matrix can be used.
  - An iid covariance matrix is a scaled identity matrix,

$$\Sigma_j = \sigma_j^2 I, \tag{8}$$

    where $\sigma_j^2 > 0$ is a scalar value. In the M-step, the estimate is $\sigma_j^2 = \frac{1}{d\hat{N}_j} \sum_{i=1}^n \hat{z}_{ij}(x_i - \hat{\mu}_j)^T(x_i - \hat{\mu}_j)$.
  - A diagonal covariance matrix is non-zero on the diagonal, and zero everywhere else.

$$\Sigma_j = \begin{bmatrix} s_{j,1} & 0 & 0 & \cdots & 0 \\ 0 & s_{j,2} & 0 & \ddots & \vdots \\ 0 & 0 & s_{j,3} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & s_{j,d} \end{bmatrix} = \text{diag}(s_j), \tag{9}$$

    where $s_j = [s_{j,1}, \cdots, s_{j,d}]^T$ is a vector of the individual variances $s_j > 0$. In the M-step, the estimate of the variances is $s_j = \frac{1}{\hat{N}_j} \sum_{i=1}^n \hat{z}_{ij}(x_i - \hat{\mu}_j) \circ (x_i - \hat{\mu}_j)$, where $\circ$ is the Hadamard product (the element-wise product, or "`.*`" in MATLAB).

  Both of these types of covariances ignore the correlation between the features, and hence each Gaussian component models the dimensions independently. Nonetheless, the overall GMM can still represent correlations between features.
- *Regularization*: Regularizing the covariance matrices can help to prevent overfitting and improve the estimates. This can also help to prevent singularities, i.e., some covariances going to zero. There are two ways to regularize the covariance:
  - add a constant to the diagonal,

$$\Sigma_j \leftarrow \Sigma_j + \alpha I, \tag{10}$$

    where α is the regularization parameter.

– enforce a minimum variance by bounding the eigenvalues of the covariance matrix. Let $\{v_i, \lambda_i\}$ be the eigenvectors and eigenvalues of Σ_j, and $\{V, \lambda\}$ the corresponding matrix of eigenvectors and vector of eigenvalues. The eigenvalues are bounded to form a new $\hat{\lambda}$ and the covariance matrix is reconstructed:

$$\hat{\lambda}_i = \begin{cases} \lambda_i, & \lambda_i \geq \alpha \\ \alpha, & \lambda_i < \alpha \end{cases}, \tag{11}$$

$$\Sigma_j \leftarrow V \text{diag}(\hat{\lambda}) V^T. \tag{12}$$

- *Empty clusters*: During the EM iterations, if an empty cluster is found (all $\hat{z}_{ij}$ are small for a given j, or $\hat{N}_j$ is small), then the cluster can be reinitialized, e.g. using the same initialization methods above. This could also be performed for clusters with a single data point to prevent singularities.

- *Convergence*: There are several ways to test for convergence of EM: 1) check that the percent change in data log-likelihood, $\log p(Y)$, is below a threshold; 2) check that the change in soft assignments is below a threshold; 3) check that the change in parameter values is below a threshold; 4) just run a set number of iterations. Personally, I prefer the percent change in log-likelihood as the test for convergence.

  EM guarantees that the data log-likelihood, $\log p(Y)$, will always increase. Hence, if you find that this is not the case, then there is probably something wrong with your implementation!

- *Numerical stability*: For very high-dimensional spaces, the log-likelihoods under each component might be very small. This may cause problems when calculating $\hat{z}_{ij}$, since a naive approach would directly use $p(x|j)$, which might be 0 if the log-likelihood is small enough. A better approach is to calculate everything in the log-domain. See Problems 4.13 and 4.14 for more details.