UCSB Spring 2018

ECE 283: Soft K-means, K-means and deterministic annealing

1 Recap of the EM algorithm for Gaussian mixtures

(This overlaps with the previous handout, providing a little more detail on the derivation.)

Recall that our observed data $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. The $\mathbf{x}_i$ is modeled as being i.i.d. draws from a mixture of K Gaussians. For $1 \leq k \leq K$, the kth component of the mixture is a $N(\mathbf{m}_k, \mathbf{C}_k)$ distribution, and π_k denotes the probability of drawing $\mathbf{X}_i$ from the kth component. The parameters characterizing the mixture are therefore given by

$$\theta = \{\mathbf{m}_k, \mathbf{C}_k, \pi_k, 1 \leq k \leq K\}$$

with the constraint that

$$\pi_k \geq 0, \quad \sum_{k=1}^{K} \pi_k = 1$$

The mixture density is given by

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mathbf{m}_k, \mathbf{C}_k)$$

Introduce hidden data $Z = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$ such that $\mathbf{z}_i$ tells us which component $\mathbf{x}_i$ is drawn from. We use one-hot encoding for the $\mathbf{z}_i = (z_i[1], ..., z_i[K])^T$., with $\mathbf{z}_i = \mathbf{e}_k$ if $\mathbf{x}_i$ is drawn from component k.

Since $\log p(X, Z|\theta) = \sum_{i=1}^{N} \log p(\mathbf{x}_i, \mathbf{z}_i|\theta)$ and expectation is a linear operator, we can separately take the expectations of the terms in the summation. For the ith term, we have

$$p(\mathbf{x}_i, \mathbf{z}_i|\theta) = p(\mathbf{x}_i|\mathbf{z}_i, \theta)p(\mathbf{z}_i|\theta)$$

so that

$$\log p(\mathbf{x}_i, \mathbf{z}_i|\theta) = \log p(\mathbf{x}_i|\mathbf{z}_i, \theta) + \log p(\mathbf{z}_i|\theta) \tag{1}$$

Since

$$P(\mathbf{z}_i = \mathbf{e}_k|\theta) = \pi_k$$

we can take advantage of the one-hot encoding to write

$$p(\mathbf{z}_i|\theta) = \prod_{k=1}^{K} \pi_k^{z_i[k]}$$

so that

$$\log p(\mathbf{z}_i|\theta) = \sum_{k=1}^{K} z_i[k] \log \pi_k \tag{2}$$

Conditioned on $\mathbf{z}_i$, the density of $\mathbf{x}_i$ is a Gaussian:

$$\log p(\mathbf{x}_i|\mathbf{z}_i = \mathbf{e}_k, \theta) = \log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right)$$

Again, the one-hot encoding enables us to write this compactly as

$$\log p(\mathbf{x}_i|\mathbf{z}_i, \theta) = \sum_{k=1}^{K} z_i[k] \log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right) \tag{3}$$

(Only one of the K terms in the summation is nonzero at a time. The kth term being nonzero corresponds to $\mathbf{z}_i = \mathbf{e}_k$.)

Plugging (2) and (3) into (1), we obtain

$$\log p(\mathbf{x}_i, \mathbf{z}_i|\theta) = \sum_{k=1}^{K} z_i[k] \left(\log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right) + \log \pi_k\right) \tag{4}$$

The E-step requires evaluating the expectation of (4) for the current estimate of the parameter. As we discussed in class, since the observed data $\{\mathbf{x}_i\}$ is fixed, what we are really doing is finding the conditional distribution of the hidden data $\{\mathbf{z}_i\}$, conditioned on $\{\mathbf{x}_i\}$ and the current estimate $\theta = \theta^\ell$. Since the observed and hidden data points are i.i.d., we can do this separately for each data point. We use Bayes' rule for computing these *assignment probabilities:*

$$p(k|\mathbf{x}_i) = P[\mathbf{z}_i = \mathbf{e}_k|\mathbf{x}_i, \theta = \theta^\ell] = \frac{P[\mathbf{x}_i|\mathbf{z}_i = \mathbf{e}_k, \theta = \theta^\ell]P[\mathbf{z}_i = \mathbf{e}_k|\theta = \theta^\ell]}{\eta_i} = \frac{N(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k)\pi_k}{\eta_i}\ ,\ k = 1, ..., K$$

where η_i is a normalization constant that ensures that the preceding probabilities sum to one:

$$\eta_i = \sum_{j=1}^{K} N(\mathbf{x}_i|\mathbf{m}_j, \mathbf{C}_j)\pi_j$$

where $\{\pi_j, \mathbf{m}_j, \mathbf{C}_j\}_{j=1}^{K}$ correspond to the current estimate θ^ℓ.

Now, applying the E-step to a given data point, we have, from (4), that

$$E[\log p(\mathbf{x}_i, \mathbf{z}_i|\theta)|\theta^\ell] = \sum_{k=1}^{K} p(k|\mathbf{x}_i) \left(\log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right) + \log \pi_k\right)$$

We note that $\theta = \{\pi_k, \mathbf{m}_k, \mathbf{C}_k\}_{k=1}^{K}$ in the above is to be optimized over to get the next estimate $\theta^{\ell+1}$ in the M-step, and $p(k|\mathbf{x}_i)$ are numerical values that we computed using the current estimate θ^ℓ in the E-step. Summing over the data points, the function to be maximized in the M-step is obtained as

$$Q(\theta|\theta^\ell) = \sum_{i=1}^{N}\sum_{k=1}^{K} P[k|\mathbf{x}_i] \left(\log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right) + \log \pi_k\right) \tag{5}$$

where we must also impose the constraint

$$\sum_{k=1}^{K} \pi_k = 1 \tag{6}$$

Using a Lagrange multiplier λ corresponding to the constraint (6), the function to be optimized is

$$J(\theta) = \sum_{i=1}^{N}\sum_{k=1}^{K} p(k|\mathbf{x}_i) \left(\log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right) + \log \pi_k\right) - \lambda \sum_{k=1}^{K} \pi_k$$

The terms to be optimized separate out nicely. The mean and covariance for the kth component must be chosen to optimize

$$\sum_{i=1}^{N} p(k|\mathbf{x}_i) \log N\left(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k\right)$$

which is a weighted version of ML estimation for a single Gaussian distribution. It is quite easy to see that the standard ML estimates easily extend to this situation in a natural way.

The component probabilities $\{\pi_k\}$ must maximize

$$J(\pi) = \sum_{i=1}^{N}\sum_{k=1}^{K} p(k|\mathbf{x}_i)\log\pi_k - \lambda\sum_{k=1}^{K}\pi_k = \sum_{k=1}^{K}\sum_{i=1}^{N} p(k|\mathbf{x}_i)\log\pi_k - \lambda\sum_{k=1}^{K}\pi_k = \sum_{k=1}^{K}(w_k\log\pi_k - \lambda\pi_k) \tag{7}$$

where we have switched the i and k summations to get the second equality, and where weights

$$w_k = \sum_{i=1}^{N} p(k|\mathbf{x}_i) \tag{8}$$

We now have

$$\frac{\partial}{\partial\pi_k} = \frac{w_k}{\pi_k} - \lambda = 0$$

which means that

$$\pi_k = \frac{w_k}{\lambda}$$

where the Lagrange multiplier λ is chosen so as to satisfy the constraint (6). This yields

$$\lambda = \sum_{j=1}^{K}\sum_{i=1}^{N} p(j|\mathbf{x}_i) = \sum_{i=1}^{N}\sum_{j=1}^{K} p(j|\mathbf{x}_i) = \sum_{i=1}^{N} 1 = N$$

since the assignment probabilities for each data point sum to one. Thus,

$$\pi_k = \frac{w_k}{N} = \frac{1}{N}\sum_{i=1}^{N} p(k|\mathbf{x}_i) \tag{9}$$

We can now consolidate these results as follows.

1.1 Summary of EM algorithm for Gaussian mixtures

Assignment step (E step): Assume that we have estimates of the mixture parameters. For each data point $\mathbf{x}_i$ ($1 \le i \le N$) and each cluster $1 \le k \le K$, estimate the probabilities that $\mathbf{x}_i$ belongs to mixture component k:

$$p(k|\mathbf{x}_i) = \frac{\pi_k p(\mathbf{x}_i|k]}{p(\mathbf{x}_i)} = \frac{\pi_k N(\mathbf{x}_i|\mathbf{m}_k, \mathbf{C}_k)}{\sum_{j=1}^{K}\pi_j N(\mathbf{x}_i|\mathbf{m}_j, \mathbf{C}_j)} \tag{10}$$

Update step (M step): , We now use these assignment probabilities as weights as to how much each data point counts towards estimating the parameters for a given cluster, and update the parameters for each component k as follows:

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} p(k|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{N} p(k|\mathbf{x}_i)} \tag{11}$$

$$\mathbf{C}_k = \frac{\sum_{i=1}^{N} p(k|\mathbf{x}_i)(\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T}{\sum_{i=1}^{N} p(k|\mathbf{x}_i)} \tag{12}$$

$$\pi_k = \frac{\sum_{i=1}^{N} p(k|\mathbf{x}_i)}{N} \tag{13}$$

2 Soft K-Means

Estimation of the covariance matrices in (12) is computationally complex in high dimensions. One alternative is to fix the covariance matrices to some nominal value, say $\mathbf{C}_k = \frac{1}{\beta}\mathbf{I}$, where β is the nominal precision. While the number of components K is fixed for soft K-means, as we discussed in class, it is possible to vary β in a structured way to adapt the number of components, using a procedure called *deterministic annealing.* We often think of $T = \frac{2}{\beta}$ as the temperature of this process: higher temperature allows higher variation of data points around the component means, which means we can use fewer components.

In any case, let us consider a fixed K and fixed $\beta = \frac{2}{T}$ at this point. The EM algorithm in Section 1.1 specializes to the following:

Assignment step: Assume that we have estimates of the mixture parameters. For each data point $\mathbf{x}_i$ $(1 \leq i \leq N)$ and each cluster $1 \leq k \leq K$, estimate the probabilities that $\mathbf{x}_i$ belongs to mixture component k:

$$p(k|\mathbf{x}_i) = \frac{\pi_k p(\mathbf{x}_i|k]}{p(\mathbf{x}_i)} = \frac{\pi_k \exp\left(-||\mathbf{x}_i - \mathbf{m}_k||^2/T\right)}{\sum_{j=1}^K \pi_j \exp\left(-||\mathbf{x}_i - \mathbf{m}_j||^2/T\right)} \tag{14}$$

Update step (M step): , We now use these assignment probabilities as weights as to how much each data point counts towards updating the component probabilities and means.

$$\mathbf{m}_k = \frac{\sum_{i=1}^N p(k|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^N p(k|\mathbf{x}_i)} \tag{15}$$

$$\pi_k = \frac{\sum_{i=1}^N p(k|\mathbf{x}_i)}{N} \tag{16}$$

3 K-Means

The term K-means is used for a hard version of the preceding algorithm, which we can get by letting the temperature T get small. As $T \to 0$ in (14), each data point simply maps to the component whose mean it is closest to. To see this, set

$$k^*(i) = \arg\min_{1\leq k\leq K} ||\mathbf{x}_i - \mathbf{m}_k|| \tag{17}$$

Assuming that the minimum is unique, we have $||\mathbf{x}_i - \mathbf{m}_k||^2 > ||\mathbf{x}_i - \mathbf{m}_{k^*(i)}||^2$ for $k \neq k^*(i)$. This implies that, as $T \to 0$,

$$\frac{\exp\left(-||\mathbf{x}_i - \mathbf{m}_k||^2/T\right)}{\exp\left(-||\mathbf{x}_i - \mathbf{m}_{k^*(i)}||^2/T\right)} \to 0, \ k \neq k^*(i)$$

Plugging into (14), we obtain the hard assignment:

$$p(k|\mathbf{x}_i) = \begin{cases} 1, & k = k^*(i) \\ 0, & else \end{cases} \tag{18}$$

The update step also simplifies. Let $C_k = \{i : k^*(i) = k\}$ denote the set of data points that have been assigned to component k. Then we have $p(k|\mathbf{x}_i) = 1$ for $i \in C_k$, and $p(k|\mathbf{x}_i) = 0$ else, which implies $\sum_{i=1}^N p(k|i) = |C_k|$ (the cardinality of C_k). Plugging into (15), we obtain

$$\mathbf{m}_k = \frac{1}{|C_k|}\sum_{i\in C_k} \mathbf{x}_i$$

That is, the mean of the kth component is updated to the centroid of the data points that have been assigned to the kth component.

We can summarize as follows.

Assignment step in K-means Map each data point to the nearest *cluster center,* which is what we can now start calling the $\{\mathbf{m}_k\}$.

$$C_k = \{i : k = \arg\min_{1 \leq j \leq K} ||\mathbf{x}_i - \mathbf{m}_j||\} \tag{19}$$

Update step in K-means: Update each cluster center to the centroid of points that have been mapped to it.

$$\mathbf{m}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{x}_i \tag{20}$$

We have arrived at the K-means algorithm via a circuitous route, first deriving the EM algorithm, then specializing to soft K-means, and then to K-means. Since K-means is the simplest of these algorithms, we usually describe it first, as the local minimum of a distortion measure found by an alternating minimization, as follows.

3.1 K-means as local minimizer of distortion

Suppose that we want to represent the data $\{\mathbf{x}_i, i = 1, ..., N\}$ by a set of K codewords $\{\mathbf{m}_k, k = 1, ..., K\}$ so as to to minimize the distortion, measured as squared error:

$$D = \sum_{i=1}^{N} \sum_{k=1}^{K} p(k|\mathbf{x}_i) ||\mathbf{x}_i - \mathbf{m}_k||^2 \tag{21}$$

where we allow soft assignments $p(k|\mathbf{x}_i)$ to start with.

However, if we now fix the codewords $\{\mathbf{m}_k\}$ and optimize over the assignments $\{p(k|\mathbf{x}_i)\}$, we see immediately that a hard (greedy) assignment to the closest codeword is optimal:

$$p(k|\mathbf{x}_i) = \begin{cases} 1 & k = \arg\min_j ||\mathbf{x}_i - \mathbf{m}_j|| \\ 0 & \textit{else} \end{cases}$$

The data points thus get assigned into K clusters as in (19), and the distortion is given by

$$D = \sum_{k=1}^{K} \sum_{i \in C_k} ||\mathbf{x}_i - \mathbf{m}_k||^2 = \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)^T (\mathbf{x}_i - \mathbf{m}_k) \tag{22}$$

Taking the gradient with respect to $\mathbf{m}_k$, we obtain

$$\nabla_{\mathbf{m}_k} D = \sum_{i \in C_k} -2(\mathbf{x}_i - \mathbf{m}_k) = 0$$

which yields the update step (20).

Thus, if we do greedy iterative optimization of the squared error distortion, we get the K-means algorithm. The distortion D is monotonically non-increasing under each iteration, hence the procedure is guaranteed to converge. Unfortunately, there is no guarantee that what it converges to is the global minimum, and indeed, poor initializations can lead to convergence to a poor local minima;

4 Deterministic Annealing

We restrict attention to a practical description of deterministic annealing. We have seen that iterative minimization of the distortion as in Section 3.1 leads to K-means, which can get locked into bad local minima. We also showed in Section 3 that this hard K-means solution follows from specializing soft K-means as in Section 2 as $T \to 0$.

Now, consider what happens to soft K-means in the other extreme, as $T \to \infty$. Regardless of our current state, we have

$$\exp\left(-||\mathbf{x}_i - \mathbf{m}_k||^2/T\right) \to 1$$

if $\mathbf{x}_i \neq \mathbf{m}_k$ (which is satisfied with probability one under any reasonable statistical model for the data points). The assignment step (14) now degenerates to

$$p(k|\mathbf{x}_i) = \pi_k$$

for all i.

The update step (15) specializes to

$$\mathbf{m}_k = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

for all k, with π_k remaining unchanged under the computation in (16).

That is, regardless of how large a value of K we choose, or our initialization of $\{\mathbf{m}_k\}$, all of the means collapse to a single point, the centroid of the data points. Which means that, for $T \to \infty$, there is only one "right answer," and the effective $K = 1$.

The key idea of deterministic annealing is to start with large T, where, regardless of the initial number of codewords, we collapse to $K = 1$, and then start reducing T gradually. One possible implementation as follows.
1) *Initialization:* Start with high enough temperature T such that we have one cluster, $K = 1$, corresponding to codeword

$$\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

2) *Cooling:* replace T by αT, where $\alpha < 1$.

- From the current set of K codewords, create $2K$ codewords by replacing $\mathbf{m}_k$ by $\mathbf{m}_k$ and $\mathbf{m}_k + \epsilon_k$, where ϵ_k is a random perturbation. Split the prior probabilities between these new codewords, assigning them priors of $\pi_k/2$ each.
- Run soft K-means until convergence.
- Merge codewords that are close enough, returning a new value of K.

3) *Check for termination:* If number of codewords is smaller than the maximum budgeted for, or the temperature is bigger than the lowest temperature targeted (this could be set based on the desired fidelity with which the codewords represent the points they are close to), go back to step 2. Otherwise terminate the algorithm.
4) *Quenching:* An optional step after termination is to run hard K-means iterations on the current set of codewords. This procedure corresponds to letting $T \to 0$, and can therefore be termed *quenching.*

For a detailed exposition of deterministic annealing (with a notation slightly different from ours), including information-theoretic interpretations and a sampling of applications, see the following paper:
K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proc. IEEE,* November 1998.