CS498: Applied Machine Learning **Spring 2021**

EM on Gaussian Mixture Model in Matrix Form

The purpose of this document is to present the EM update formulas in the matrix form to help you in their Python implementation. Before reading this document, you should study the material about Expectation Maximization (EM) and EM for Mixture of Normal Distributions (Chapter 9 of the textbook [For19]) where you will find the concepts you need to understand. This document does not provide any new concepts since it was designed to help you implement EM on Mixture of Normals for your assignment based on what you have already studied.

Review of EM update formulas for the mixture of normals model

As described in the course, the update formulas in EM for the mixture of normals model (Procedures 9.2 and 9.3 from the textbook [For19]) are as follows:

1. The E-step

$$W_{i,j}^{(n)} = \frac{\left[e^{-\frac{1}{2}\left[(\mathbf{x}_i-\boldsymbol{\mu}_j)^T(\mathbf{x}_i-\boldsymbol{\mu}_j)\right]}\right]\pi_j^{(n)}}{\sum_{l=1}^{K}\left[\left[e^{-\frac{1}{2}\left[(\mathbf{x}_i-\boldsymbol{\mu}_l)^T(\mathbf{x}_i-\boldsymbol{\mu}_l)\right]}\right]\pi_l^{(n)}\right]} \tag{1}$$

2. The M-step

$$\mu_j^{(n+1)} = \frac{\sum_{i=1}^{N}\mathbf{x}_i W_{i,j}^{(n)}}{\sum_{i=1}^{N} W_{i,j}^{(n)}}, \tag{2a}$$

$$\pi_j^{(n+1)} = \frac{\sum_{i=1}^{N} W_{i,j}^{(n)}}{N}, \tag{2b}$$

where

N Number of data points,

K Number of clusters,

d Dimension of the data space, here it is the number of fundamental pixel colors,

$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$ is the vector of pixels in the dataset $1 \leq i \leq N$,

$\boldsymbol{\mu}_j$ The vector of centroid j, with a dimension of d rows and one column,

π_j The probability of cluster j,

${}^{(n)}$ denotes the time step.

Matrix Representation

We want to implement the above update formulas in Python. Hence, it would be efficient to store the variables as matrices. Moreover, to simplify the notation, we may drop the iteration superscript $^{(n)}$ and write, for instance, W instead of $W^{(n)}$.

Let

$$X = \left[\begin{array}{c} \mathbf{x}_1^T \\ \hline \mathbf{x}_2^T \\ \hline \cdots \\ \hline \mathbf{x}_N^T \end{array}\right]$$

be the $N \times d$ matrix containing the input data (pixels). We assume that $\mathbf{x}_i$ is a $d \times 1$ column vector, that's why we write $\mathbf{x}_i^T$ as the ith row of X.

Next, let

$$\mu = \left[\begin{array}{c} \boldsymbol{\mu}_1^T \\ \hline \boldsymbol{\mu}_2^T \\ \hline \cdots \\ \hline \boldsymbol{\mu}_K^T \end{array}\right]$$

be the $K \times d$ matrix containing the centroids of the multinomial distribution for each cluster. Similar to the above, we assume that μ_j is a $d \times 1$ column vector, and that is why we write μ_j^T for the jth row of μ.

Also, let

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \cdots \\ \pi_K \end{bmatrix} \tag{3}$$

be the $K \times 1$ vector of prior probabilities of π_j as described iin the textbook.

Matrix Operations

These are the operations we use in this review:

$\mathbf{u}^T$ The transpose of $\mathbf{u}$

$A \circ B$ The Hadamard product of matrices A and B (i.e. the element-wise multiplication of A and B matrices).

$A \oslash B$ The Hadamard division of matrices A and B (i.e. the element-wise division of A over B matrices).

$A \cdot B$ The matrix multiplication of A and B matrices.

$A_{m\times n}$ Some matrix A having m rows and n columns.

$\mathbf{1}_{m\times n}$ matrix of all one elements, with m rows and n columns.

W The posterior probability matrix with elements of $w_i j$ as described in the textbook.

$\exp(A)$ The element-wise exponentiation of matrix A.

$\log(A)$ The element-wise logarithm of matrix A.

E-Step in Matrix Form

In order to simplify 1, let's define a matrix H with N rows and d columns whose elements are:

$$H[i,j] = -\frac{1}{2}\Big[(\mathbf{x}_i - \boldsymbol{\mu}_j)^T(\mathbf{x}_i - \boldsymbol{\mu}_j)\Big] = -\frac{1}{2}\Big[\mathbf{x}_i^T\mathbf{x}_i + \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j - 2\mathbf{x}_i^T\boldsymbol{\mu}_j\Big]$$

We start by computing $H[i,j]$ in matrix form. First, we define

$$H = -\frac{1}{2}(H_1 + H_2 - 2H_3)$$

Computing H_1 matrix: $H_1[i,j] = \mathbf{x}_i^T\mathbf{x}_i$

1. We start with the $N \times d$ matrix with the squared values of X:

$$U = \Big[X \circ X\Big]_{N\times d}$$

2. Now we sum over the rows in order to compute the $\mathbf{x}_i^T\mathbf{x}_i$ values in the column vector

$$V_{N\times 1} = U_{N\times d} \cdot \mathbf{1}_{d\times 1}$$

3. Next, we repeat the V column K times along the column axis by right-multiplying with a $\mathbf{1}_{1\times K}$ matrix

$$H_1 = V_{N\times 1} \cdot \mathbf{1}_{1\times K}$$

The resulting H_1 matrix is:

$$H_1 = \Big[X \circ X\Big] \cdot \mathbf{1}_{d\times 1} \cdot \mathbf{1}_{1\times K} = \Big[X \circ X\Big] \cdot \mathbf{1}_{d\times K}$$

Computing H_2 matrix: $H_2[i,j] = \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j$

1. We start with the $K \times d$ squared values of μ:

$$A = \Big[\mu \circ \mu\Big]_{K\times d}$$

2. Now we sum over the rows in order to compute the $\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j$ values in the column vector

$$B_{K\times 1} = A_{K\times d} \cdot \mathbf{1}_{d\times 1}$$

3. Next, we transpose the B column to get a row matrix, and repeat the B column N times along the row axis by left-multiplying with a $\mathbf{1}_{N\times 1}$ matrix

$$H_2 = \mathbf{1}_{N\times 1} \cdot [B^T]_{1\times K}$$

The resulting H_2 matrix is:

$$H_2 = \mathbf{1}_{N\times 1} \cdot [[\mu \circ \mu] \cdot \mathbf{1}_{d\times 1}]^T = \mathbf{1}_{N\times 1} \cdot \mathbf{1}_{d\times 1}^T \cdot \Big[\mu \circ \mu\Big]^T$$

$$= \mathbf{1}_{N\times 1} \cdot \mathbf{1}_{1\times d} \cdot \Big[\mu \circ \mu\Big]^T = \mathbf{1}_{N\times d} \cdot \Big[\mu \circ \mu\Big]^T$$

Computing H_3 matrix: $H_3[i,j] = \mathbf{x}_i^T \boldsymbol{\mu}_j$

H_3 is straightforward:

$$H_3 = X \cdot \mu^T$$

Putting matrices together: $H = -\frac{1}{2}(H_1 + H_2 - 2H_3)$

Therefore,

$$H = -\frac{1}{2}\left(\left[X \circ X\right] \cdot \mathbf{1}_{d \times K} + \mathbf{1}_{N \times d} \cdot \left[\mu \circ \mu\right]^T - 2X \cdot \mu^T\right)$$

Note 1 The repetition steps could be done more efficiently if you think about them...

Note 2 Also,... the summation steps can be done without matrix multiplication. You'll have to figure out how.

Note 3 Some languages let you add a column matrix (i.e. with N rows and one column) to a row matrix (i.e. with one row and K columns), and get a populated matrix (i.e. with N rows and K columns). If you could do this, you may be even able to skip the repetition parts of H_1 and H_2, and just add a row matrix to a column matrix, and hopefully save some time! Try to do this wisely :)

W matrix

Now that we have H, we replace its elements $H[i,j]$ in 1 to get:

$$W[i,j] = \frac{e^{H[i,j]}\pi_j}{\sum_{l=1}^{K}\left[e^{H[i,j]}\pi_l\right]} = \frac{E[i,j]}{F[i,j]}$$

Numerator of $W[i,j]$ elements: $E[i,j]$

Let's define a P matrix whose elements are $P[i,j] = \pi_j$. We can compute it through matrix multiplication as follows:

$$P_{N \times K} = \mathbf{1}_{N \times 1} \cdot [\pi^T]_{1 \times K} = \mathbf{1}_{N \times 1} \cdot \pi^T$$

So, the numerators $E[i,j]$ can be computed as:

$$E = \exp(H) \circ P$$

Denominator of $W[i,j]$ elements: $F[i,j]$

For computing F, we sum over the columns of E, and repeat the summation over the column axis:

$$F_{N \times K} = E_{N \times K} \cdot \mathbf{1}_{K \times 1} \cdot \mathbf{1}_{1 \times K} = E_{N \times K} \cdot \mathbf{1}_{K \times K}$$

Putting matrices together: $W = E \oslash F$

$$W = \left[\exp(H) \circ P\right] \oslash \left[\mathbf{1}_{K \times N} \cdot (\exp(H) \circ P)\right]$$

Throughout the computations, repetitions may be doable without matrix multiplication, and probably you could speed this up by doing summations and repetitions separately using predefined functions in your programming language.

Robustifying the computation of $W[i,j]$ elements

Question: Can we do these computations in log space?, I'm having numerical stability issues...

Answer: Sure, let's do that. In log space, 1 becomes:

$$\log(W[i,j]) = \log(e^{H[i,j]}) + \log(\pi_j) - \log\left(\sum_{l=1}^{K}\left[e^{H[i,j]}\pi_l\right]\right).$$

So, we get:

$$\log(W) = \log(E) - \log(F)$$

$$\log(E) = H + \log(P)$$

- $\log(P)$ can be easily obtained from $\log(\pi)$ values. You'll have to take care of the repetition step, or you might be able to use certain tools in your programming language to bypass that ...
- After computing $\log(E)$, you can easily obtain $\log(F)$ using some sort of log(sum(exp)) function. You just have to apply the log(sum(exp)) function over each row of $\log(E)$ matrix, and then do the repetition step.

M-step in Matrix Form

In the E-step, we obtain a matrix of weights W as the following:

$$W = \left[\begin{array}{c|c|c|c} & & & \\ \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_K \\ & & & \end{array}\right]$$

Update for μ_j

According to 2a, in the M step, the update for μ_j is:

$$\mu[j] = \frac{\sum_{i=1}^{N} \mathbf{x}_i W[i,j]}{\sum_{i=1}^{N} W[i,j]}$$

Numerator of μ: $\sum_i \mathbf{x}_i W[i,j]$

The multiplication $[\mathbf{w}_j^T]_{1\times N} \cdot X_{N\times K}$ gives us a row matrix $(1 \times K)$. For all clusters:

$$D_{K\times d} = [W^T]_{K\times N} \cdot X_{N\times d} = W^T \cdot X$$

Denominator of μ: $\sum_i W[i,j]$

Following a similar process:

$$G_{K\times d} = [W^T]_{K\times N} \cdot \mathbf{1}_{N\times 1} \cdot \mathbf{1}_{1\times d} = W^T \cdot \mathbf{1}_{N\times d}$$

Putting matrices together $\mu = D \oslash G$

$$\mu = (W^T \cdot X) \oslash (W^T \cdot \mathbf{1}_{N\times d})$$

As before, faster computations can be achieved through proper summation and repetition functions.

Update for π_j

According to 2b, in the M-Step, the update for π_j is:

$$\pi_j = \frac{\sum_{i=1}^{N} W_{i,j}}{N}$$

So, π can be rewritten as:

$$\pi_{K\times 1} = \frac{[W^T]_{K\times N} \cdot \mathbf{1}_{N\times 1}}{N} = \frac{W^T \cdot \mathbf{1}_{N\times 1}}{N}$$

Robustifying the computation of μ_j and π

Question: Can we perform these computations in the log space to prevent numerical stability issues?

for π

$$\log(\pi) = \log(W^T \cdot \mathbf{1}_{N\times 1}) - \log(N)$$

Note that $\log(N)$ is just a scalar and we can subtract a scalar from a vector.

In order to compute $\log(W^T \cdot \mathbf{1}_{N\times 1})$, we can apply some log(sum(exp)) function over each row of $\log(W)^T$ matrix. (Alternatively, we can apply the log(sum(exp)) function over each column of $\log(W)$ matrix and then transpose the result.)

for $\boldsymbol{\mu}$

$$\mu = (W^T \cdot X) \oslash (W^T \cdot \mathbf{1}_{N\times d})$$
$$R = W^T = \exp\left[\log(W^T)\right]$$

Now we can evaluate:

$$\mu = (R \cdot X) \oslash (R \cdot \mathbf{1}_{N\times d})$$

Note that for $R \cdot \mathbf{1}_{N\times d}$ you can also use some log(sum(exp)) function.

References

[For19] David Forsyth. *Applied Machine Learning.* Springer, 2019.