CS498: Applied Machine Learning **Spring 2021**

EM Algorithm on the Topic Model in Matrix Form

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

Review of EM update formulas for the topic model

As you have seen in the course, the update formulas in EM for the topic model (Procedures 9.4 and 9.5 from the textbook [For19]) are as follows:

1. The E-step

$$W_{i,j}^{(n)} = \frac{\left[\prod_{k=1}^{d} (P_{j,k}^{(n)})^{x_{i,k}}\right] \pi_j^{(n)}}{\sum_{l=1}^{t} \left[\prod_{k=1}^{d} (P_{l,k}^{(n)})^{x_{i,k}}\right] \pi_l^{(n)}}. \tag{1}$$

2. The M-step

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

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

where

N is the number of documents,

d is the number of possible words that show up in the documents,

$x_{i,j}$ denotes the number of times word j occurs in document i,

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

t is the number of topics,

π_j is the probability of topic $1 \leq j \leq d$,

$\mathbf{p}_j = (p_{j,1}, \ldots, p_{j,d})$ is the multinomial distribution associated to topic $1 \leq j \leq t$, and

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

Therefore, given the learned parameters $\theta^{(n)} = (\pi_1^{(n)}, \ldots, \pi_t^{(n)}, \mathbf{p}_1^{(n)}, \ldots, \mathbf{p}_t^{(n)})$, we should compute the weights $W_{i,j}^{(n)}$ during the E-step, and use them to obtain the updated parameters $\theta^{(n+1)}$.

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 word count. Therefore, entries in X are nonnegative integers, and $\mathbf{x}_i^T \mathbf{1}_{d\times 1}$ is the number of words in document i. Here, we assume that $\mathbf{x}_i$ is a $d \times 1$ column vector, that is why we write $\mathbf{x}_i^T$ as the ith row of X.

Next, let

$$P = \left[\begin{array}{c} \mathbf{p}_1^T \\ \hline \mathbf{p}_2^T \\ \hline \cdots \\ \hline \mathbf{p}_t^T \end{array}\right]$$

be the $t \times d$ matrix containing the multinomial distribution for each class. Therefore, the entries in each row of P are nonnegative, and sum up to one. Similar to the above, we assume that $\mathbf{p}_j$ is a $d \times 1$ column vector, and that is why we write $\mathbf{p}_j^T$ for the jth row of P.

Also, let

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \cdots \\ \pi_t \end{bmatrix},$$

be the $t \times 1$ matrix containing the class distributions. Thereby, the entries of π are nonnegative and sum up to one.

Furthermore, let W be the $N \times t$ matrix containing the weight variables defined in (1). Recall that we can think of $W_{i,j}$ as the chance of document i being a member of class (topic) j. Therefore, W contains nonnegative values and each row sums up to one.

Throughout this document, we may choose to carry the dimensions of a matrix as subscript to its symbol to remember its dimension. For instance, by writing $X_{N\times d}$, we refer to the matrix X which has N rows and d columns.

E-step in Matrix Form

Recalling (1), we define the $N \times t$ matrix R such that

$$R_{i,j} = \pi_j \prod_{k=1}^{d} P_{j,k}^{x_{i,k}}. \tag{3}$$

In order to calculate W based on R, we need to normalize its rows. So we define S to be the $N \times t$ matrix such that

$$S_{i,j} = \sum_{l=1}^{t} R_{i,l}. \tag{4}$$

Note that all the entries in each row of S have the same value. With these, we have

$$W_{i,j} = \frac{R_{i,j}}{S_{i,j}}. \tag{5}$$

In order to write (3) in matrix form, if we apply the logarithm to both sides, we get

$$\log(R_{i,j}) = \log(\pi_j) + \sum_{k=1}^{d} x_{i,k} \log(P_{j,k}). \tag{6}$$

Motivated by this, let us define $\widetilde{R}$ be the $N \times t$ matrix where $\widetilde{R}_{i,j} = \log R_{i,j}$. All the logarithms in this document are in the natural basis. From this point forward, given an arbitrary matrix A with positive entries, we define $\log A$ to be the matrix with the same dimension which is the element-wise logarithm of the entries in A, so for instance $\widetilde{R} = \log R$. Similarly define $\widetilde{\pi}_{t\times 1} = \log \pi$ and $\widetilde{P}_{t\times d} = \log P$.

Rewriting (6), we get

$$\widetilde{R}_{i,j} = \widetilde{\pi}_j + \sum_{k=1}^{d} x_{i,k} \widetilde{P}_{i,j},$$

or in matrix form

$$\widetilde{R}_{N\times t} = \mathbf{1}_{N\times 1} \cdot (\widetilde{\pi}^T)_{1\times t} + X_{N\times d} \cdot (\widetilde{P}^T)_{d\times t}. \tag{7}$$

Here, $\cdot$ indicates the usual matrix multiplication, and $\mathbf{1}_{a\times b}$ is an $a \times b$ matrix with all values being equal to one.

In addition to this, if we define $\widetilde{S} = \log S$, from (4), we have

$$\widetilde{S}_{i,j} = \log S_{i,j} = \log\left(\sum_{l=1}^{t} R_{i,l}\right) = \log\left(\sum_{l=1}^{t} \exp(\widetilde{R}_{i,l})\right). \tag{8}$$

Finally, based on (5), if we define $\widetilde{W} = \log W$, we arrive at

$$\widetilde{W}_{N\times t} = \widetilde{R}_{N\times t} - \widetilde{S}_{N\times t}. \tag{9}$$

To sum up, the E-step consists of the following smaller steps:

1. Find $\widetilde{R}$ following (7).
2. Find $\widetilde{S}$ following (8).
3. Compute $\widetilde{W}$ from $\widetilde{R}$ and $\widetilde{S}$ using (9).

Remark 1. *In Python, in order to obtain $\widetilde{S}$ from $\widetilde{R}$, motivated by (8), you might want to use the function* `logsumexp` *from* `scipy.special`.

Remark 2. *You might be able to use the broadcasting concept in* `numpy` *to do part of the above calculations more easily and efficiently. Therefore, you should not implement the above formulas exactly as they are, and you should adapt them to have an efficient implementation.*

M-step in Matrix Form

Updating P

We want to write the update equation (2a) in matrix form. Note that the numerator in (2a) can be easily represented via matrix multiplication. More precisely, let $E = E_{t\times d}$ be the matrix defined as

$$E_{t\times d} = (W^T)_{t\times N} \cdot X_{N\times d}. \tag{10}$$

Then, it is easy to see that the jth row of E is precisely $\sum_{i=1}^{N} \mathbf{x}_i W_{i,j}$, or the numerator in (2a) (you should verify this yourself).

Remember that each row in the matrix P is a probability distribution and hence should sum up to one. This means that in order to obtain the matrix P from the matrix E, we should normalize the rows (this is essentially what the denominator in (2a) is doing, and you should be able to verify this yourself). More precisely, let $F = F_{N\times t}$ be the matrix such that

$$F_{i,j} = \sum_{l=1}^{d} E_{i,l}, \tag{11}$$

which contains the row sums of the matrix E. With these, motivated by (2a), the matrix P is obtained by element-wise division of the matrix E by the matrix F, i.e.

$$P_{i,j} = \frac{E_{i,j}}{F_{i,j}}. \tag{12}$$

Updating P in the logarithmic domain

Remember from the E-step that we want to do our calculations in the logarithm domain. Remember from the E-step that we calculate the matrix $\widetilde{W} = \log W$. Once we have $\widetilde{W}$, we can use the element-wise exponential function to find W and use (10) to find E. Then, we can apply the element-wise logarithm function to find the matrix $\widetilde{E}_{t\times d} = \log E$.[1] Note that if we define $\widetilde{F} = \log F$, using (11), we have

$$\widetilde{F}_{i,j} = \log F_{i,j} = \log\left(\sum_{l=1}^{d} E_{i,l}\right) = \log\left(\sum_{l=1}^{d} \exp(\widetilde{E}_{i,l})\right). \tag{13}$$

Finally, from (12), we can find $\widetilde{P} = \log P$ as follows

$$\widetilde{P} = \widetilde{E} - \widetilde{F}. \tag{14}$$

Remark 3. *In Python, in order to obtain $\widetilde{F}$ from $\widetilde{E}$, motivated by (13), you might want to use the function* `logsumexp` *from* `scipy.special`*.*

Remark 4. *You might be able to use the broadcasting concept in* `numpy` *to do part of the above calculations more easily and efficiently. Therefore, you should not implement the above formulas exactly as they are, and you should adapt them to have an efficient implementation.*

Updating π

By taking logarithms from both sides in the update equation (2b), and recalling that we have computed the matrix $\widetilde{W} = \log W$ in the E-step, for $1 \le j \le t$ we get

$$\widetilde{\pi}_j = \log \pi_j = \log\left(\sum_{i=1}^{N} W_{i,j}\right) - \log N = \log\left(\sum_{i=1}^{N} \exp(\widetilde{W}_{i,j})\right) - \log N. \tag{15}$$

Remark 5. *In Python, in order to obtain $\widetilde{\pi}$ from $\widetilde{W}$, motivated by (15), you might want to use the function* `logsumexp` *from* `scipy.special`*.*

References

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

[1] in the programming assignment, we will add a small ϵ at this step to ensure numerical stability