

In Lecture 3 we saw that the Maximum Likelihood (ML) principle over i.i.d. data is achieved by minimizing the relative entropy between a model \mathcal{Q} and the occurrence-frequency of the training data. Specifically, let $\mathbf{x}_1, \dots, \mathbf{x}_m$ be i.i.d. where each $\mathbf{x}_i \in \mathcal{X}^d$ is a d -tuple of symbols taken from an alphabet \mathcal{X} having n different letters $\{a_1, \dots, a_n\}$. Let \hat{P} be the empirical joint distribution, i.e., an array with d dimensions where each axis has n entries, i.e., each entry $\hat{P}_{i_1, \dots, i_d}$, where $i_j = 1, \dots, n$, represents the (normalized) co-occurrence of the d -tuple a_{i_1}, \dots, a_{i_d} in the training set $\mathbf{x}_1, \dots, \mathbf{x}_m$. We wish to find a joint distribution P^* (also a d -array) which belongs to some model family of distributions \mathcal{Q} closest as possible to \hat{P} in relative-entropy:

$$P^* = \operatorname{argmin}_{P \in \mathcal{Q}} D(\hat{P} \| P).$$

In this lecture we will focus on a model of distributions \mathcal{Q} which represents *mixtures* of simple distributions \mathcal{H} —known as *latent class models*. A latent class model arises when the joint probability $P(X_1, \dots, X_d)$ we observe (i.e., from which \hat{P} is generated by observing samples $\mathbf{x}_1, \dots, \mathbf{x}_m$) is in fact a marginal of $P(X_1, \dots, X_d, Y)$ where Y is a "hidden" (or "latent") random variable which has k different discrete values $\alpha_1, \dots, \alpha_k$. Then,

$$P(X_1, \dots, X_d) = \sum_{j=1}^k P(X_1, \dots, X_d | Y = \alpha_j) P(Y = \alpha_j).$$

The idea is that given the value of the hidden variable H the problem of recovering the model $P(X_1, \dots, X_d | Y = \alpha_j)$, which belongs to some family of joint distributions \mathcal{H} , is a relatively simple problem. To make this idea clearer we consider the following example: Assume we have two coins. The first coin has a probability of heads ("0") equal to p and the second coin has a probability of heads equal to q . At each trial we choose to toss coin 1 with probability λ and coin 2 with probability $1 - \lambda$. Once a coin has been chosen it is tossed 3 times, producing an observation $\mathbf{x} \in \{0, 1\}^3$. We are given a set of such observations $D = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ where each observation \mathbf{x}_i is a triplet of coin tosses (the same coin). Given D , we can construct the empirical distribution \hat{P} which is a $2 \times 2 \times 2$ array defined as:

$$\hat{P}_{i_1, i_2, i_3} = \frac{1}{m} |\{\mathbf{x}_i = \{i_1, i_2, i_3\}, i = 1, \dots, m\}|.$$

Let $y_i \in \{1, 2\}$ be a random variable associated with the observation \mathbf{x}_i such that $y_i = 1$ if \mathbf{x}_i was generated by coin 1 and $y_i = 2$ if \mathbf{x}_i was generated by coin 2. If we knew the values of y_i then our task would be simply to estimate two separate Bernoulli distributions by separating the triplets generated from coin 1 from those generated by coin 2. Since y_i is not known, we have the marginal:

$$\begin{aligned} P(\mathbf{x} = (x_1, x_2, x_3)) &= P(\mathbf{x} = (x_1, x_2, x_3) | y = 1)P(y = 1) + P(\mathbf{x} = (x_1, x_2, x_3) | y = 2)P(y = 2) \\ &= \lambda p^{n_i} (1 - p)^{(3 - n_i)} + (1 - \lambda) q^{n_i} (1 - q)^{(3 - n_i)}, \end{aligned} \quad (4.1)$$

where $(x_1, x_2, x_3) \in \{0, 1\}^3$ is a triplet coin toss and $0 \leq n_i \leq 3$ is the number of heads ("0") in the triplet of tosses. In other words, the likelihood $P(\mathbf{x})$ of triplet of tosses $\mathbf{x} = (x_1, x_2, x_3)$ is a linear combination ("mixture") of two Bernoulli distributions. Let \mathcal{H} stand for Bernoulli distributions:

$$\mathcal{H} = \{\mathbf{u}^{\otimes d} : \mathbf{u} \geq 0, \sum_{i=1}^n u_i = 1\}$$

where $\mathbf{u}^{\otimes d}$ stands for the outer-product of $\mathbf{u} \in R^n$ with itself d times, i.e., an n -way array indexed by i_1, \dots, i_d , where $i_j \in \{1, \dots, n\}$, and whose value there is equal to $u_{i_1} \cdots u_{i_d}$. The model family \mathcal{Q} is a mixture of Bernoulli distributions:

$$\mathcal{Q} = \left\{ \sum_{j=1}^k \lambda_j P_j : \lambda \geq 0, \sum_j \lambda_j = 1, P_j \in \mathcal{H} \right\},$$

where specifically for our coin-toss example becomes:

$$\mathcal{Q} = \left\{ \lambda \binom{p}{1-p}^{\otimes 3} + (1-\lambda) \binom{q}{1-q}^{\otimes 3} : \lambda, p, q \in [0, 1] \right\}$$

We see therefore that the eight entries of $P^* \in \mathcal{Q}$ which minimizes $D(\hat{P}||P)$ over the set \mathcal{Q} is determined by three parameters λ, p, q . For the coin-toss example this looks like:

$$\begin{aligned} & \operatorname{argmin}_{0 \leq \lambda, p, q \leq 1} D \left(\hat{P} \parallel \lambda \binom{p}{1-p}^{\otimes 3} + (1-\lambda) \binom{q}{1-q}^{\otimes 3} \right) \\ &= \operatorname{argmax}_{0 \leq \lambda, p, q \leq 1} \sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1 \hat{P}_{i_1 i_2 i_3} \log \left(\lambda p^{n_{i_1 i_2 i_3}} (1-p)^{(3-n_{i_1 i_2 i_3})} + (1-\lambda) q^{n_{i_1 i_2 i_3}} (1-q)^{(3-n_{i_1 i_2 i_3})} \right) \end{aligned}$$

where $n_{i_1 i_2 i_3} = i_1 + i_2 + i_3$. Trying to work out an algorithm for minimizing the unknown parameters λ, p, q would be somewhat "unpleasant" (and even more so for other families of distributions \mathcal{H}) because of the log-over-a-sum present in the optimization function — if we could somehow turn this into a sum-over-log our task would be much easier. We would then be able to turn the problem into a succession of problems over \mathcal{H} rather than a single problem over $\mathcal{Q} = \sum_j \lambda_j \mathcal{H}$. Another point worth attention is the non-negativity of the output variables — simply minimizing the relative-entropy measure under the constraints of the class model \mathcal{Q} would not guarantee a non-negative solution. As we shall see, breaking down the problem into a successions of problems over \mathcal{H} would give us the "non-negativity for free" feature.

The technique for turning the log-over-sum into a sum-over-log as part of finding the ML solution for a mixture model is known as the Expectation-Maximization (EM) algorithm introduced by Dempster, Laird and Rubin in 1977. It is based on two ideas: (i) introduce auxiliary variables, and (ii) use of Jensen's inequality.

4.1 The EM Algorithm: General

Let $D = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ represent the training data where $\mathbf{x}_i \in \mathcal{X}$ is taken from some instance space \mathcal{X} which we leave unspecified. For now, we leave matters to be as general as possible and specifically we do not make independence assumptions on the data generation process.

The ML problem is to find a setting of parameters θ which maximizes the likelihood $P(\mathbf{x}_1, \dots, \mathbf{x}_m | \theta)$, namely, we wish to maximize $P(D | \theta)$ over parameters θ , which is equivalent to maximizing the log-likelihood:

$$\theta^* = \operatorname{argmax}_{\theta} \log P(D | \theta) = \log \left(\sum_{\mathbf{y}} P(D, \mathbf{y} | \theta) \right),$$

where \mathbf{y} represents the hidden variables. We will denote $L(\theta) = \log P(D | \theta)$. Let $q(\mathbf{y} | D, \theta)$ be some (arbitrary) distribution of the hidden variables \mathbf{y} conditioned on the parameters θ and the input sample D , i.e., $\sum_{\mathbf{y}} q(\mathbf{y} | D, \theta) = 1$. We define a *lower bound* on $L(\theta)$ as follows:

$$\begin{aligned} L(\theta) &= \log \left(\sum_{\mathbf{y}} P(D, \mathbf{y} | \theta) \right) \\ &= \log \left(\sum_{\mathbf{y}} q(\mathbf{y} | D, \theta) \frac{P(D, \mathbf{y} | \theta)}{q(\mathbf{y} | D, \theta)} \right) \\ &\geq \sum_{\mathbf{y}} q(\mathbf{y} | D, \theta) \log \frac{P(D, \mathbf{y} | \theta)}{q(\mathbf{y} | D, \theta)} \\ &= Q(q, \theta). \end{aligned}$$

The inequality comes from Jensen's inequality $\log \sum_j \alpha_j a_j \geq \sum_j \alpha_j \log a_j$ when $\sum_j \alpha_j = 1$. What we have obtained is an "auxiliary" function $Q(q, \theta)$ satisfying

$$L(\theta) \geq Q(q, \theta),$$

for all distributions $q(\mathbf{y} | D, \theta)$. The maximization of $Q(q, \theta)$ proceeds by interleaving the variables q and θ as we separately ascend on each set of variables. At the $(t + 1)$ iteration we fix the current value of θ to be $\theta^{(t)}$ of the t 'th iteration and maximize $Q(q, \theta^{(t)})$ over q , and then maximize $Q(q^{(t+1)}, \theta)$ over θ :

$$\begin{aligned} q^{(t+1)} &= \operatorname{argmax}_q Q(q, \theta^{(t)}) \\ \theta^{(t+1)} &= \operatorname{argmax}_{\theta} Q(q^{(t+1)}, \theta). \end{aligned}$$

The strategy of the EM algorithm is to maximize the lower bound $Q(q, \theta)$ with the hope that if we ascend on the lower bound function we will also ascend with respect to $L(\theta)$. The claim below guarantees that an ascend on Q will also generate an ascend on L :

Claim 1 (Jordan-Bishop) *The optimal $q(\mathbf{y} | D, \theta^{(t)})$ at each step is $P(\mathbf{y} | D, \theta^{(t)})$.*

Proof: We will show that $Q(P(\mathbf{y} | D, \theta^{(t)}), \theta^{(t)}) = L(\theta^{(t)})$ which proves the claim since $L(\theta) \geq Q(q, \theta)$ for all q, θ , thus the best q -distribution we can hope to find is one that makes the lower-bound

meet $L(\theta)$ at $\theta = \theta^{(t)}$.

$$\begin{aligned} Q(P(\mathbf{y} | D, \theta^{(t)}), \theta^{(t)}) &= \sum_{\mathbf{y}} P(\mathbf{y} | D, \theta^{(t)}) \log \frac{P(D, \mathbf{y} | \theta^{(t)})}{P(\mathbf{y} | D, \theta^{(t)})} \\ &= \sum_{\mathbf{y}} P(\mathbf{y} | D, \theta^{(t)}) \log \frac{P(\mathbf{y} | D, \theta^{(t)}) P(D | \theta^{(t)})}{P(\mathbf{y} | D, \theta^{(t)})} \\ &= \log P(D | \theta^{(t)}) \sum_{\mathbf{y}} P(\mathbf{y} | D, \theta^{(t)}) \\ &= L(\theta^{(t)}) \end{aligned}$$

□

The proof provides also the validity for the approach of ascending along the lower bound $Q(q, \theta)$ because at the point $\theta^{(t)}$ the two functions coincide, i.e., the lower bound function at $\theta = \theta^{(t)}$ is equal to $L(\theta^{(t)})$ therefore if we continue and *ascend* along $Q(\cdot)$ we are *guaranteed* to ascend along $L(\theta)$ as well² — therefore, convergence is guaranteed. It can also be shown (but omitted here) that the point of convergence is a stationary point of $L(\theta)$ (was shown originally by C.F. Jeff Wu in 1983 years after EM was introduced in 1977) under fairly general conditions. The second step of maximizing over θ then becomes:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{\mathbf{y}} P(\mathbf{y} | D, \theta^{(t)}) \log P(D, \mathbf{y} | \theta). \quad (4.2)$$

This defines the EM algorithm. Often the "Expectation" step is described as taking the expectation of:

$$E_{\mathbf{y} \sim P(\mathbf{y} | D, \theta^{(t)})} [\log P(D, \mathbf{y} | \theta)],$$

followed by a Maximization step of finding θ that maximizes the expectation — hence the term EM for this algorithm.

Eqn. 4.2 describes a principle but not an algorithm because in general, without making assumptions on the statistical relationship between the data points and the hidden variable the problem presented in eqn. 4.2 is unwieldy. We will reduce eqn. 4.2 to something more manageable by making the i.i.d. assumption. This is detailed in the following section.

4.2 EM with i.i.d. Data

The EM optimization presented in eqn. 4.2 can be simplified if we assume the data points (and the hidden variable values) are i.i.d.

$$P(D | \theta) = \prod_{i=1}^n P(\mathbf{x}_i | \theta), \quad P(D, \mathbf{y} | \theta) = \prod_{i=1}^n P(\mathbf{x}_i, y_i | \theta),$$

and

$$P(\mathbf{y} | D, \theta) = \prod_{i=1}^n P(y_i | \mathbf{x}_i, \theta).$$

²this manner of deriving EM was adapted from Jordan and Bishop's book notes, 2001.

For any $\alpha(y_i)$ we have:

$$\begin{aligned} \sum_{\mathbf{y}} \alpha(y_i) P(\mathbf{y} | D, \theta) &= \sum_{y_1} \cdots \sum_{y_n} \alpha(y_i) P(y_1 | \mathbf{x}_1, \theta) \cdots P(y_n | \mathbf{x}_n, \theta) \\ &= \sum_{y_i} \alpha(y_i) P(y_i | \mathbf{x}_i, \theta) \end{aligned}$$

this is because $\sum_{y_j} P(y_j | \mathbf{x}_j, \theta) = 1$. Substituting the simplifications above into eqn. 4.2 we obtain:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{j=1}^k \sum_{i=1}^m P(y_i = \alpha_j | \mathbf{x}_i, \theta^{(t)}) \log P(\mathbf{x}_i, y_i = \alpha_j | \theta) \quad (4.3)$$

where $y_i \in \{\alpha_1, \dots, \alpha_k\}$.

4.3 Back to the Coins Example

We will apply the EM scheme to our running example of mixture of Bernoulli distributions. We wish to compute

$$\begin{aligned} Q(\theta, \theta^{(t)}) &= \sum_{\mathbf{y}} P(\mathbf{y} | D, \theta^{(t)}) \log P(D, \mathbf{y} | \theta) \\ &= \sum_{i=1}^n \sum_{j=1}^2 P(y_i = j | \mathbf{x}_i, \theta^{(t)}) \log P(\mathbf{x}_i, y_i = j | \theta), \end{aligned}$$

and then maximize $Q()$ with respect to p, q, λ .

$$\begin{aligned} Q(\theta, \theta') &= \sum_{i=1}^n [P(y_i = 1 | \mathbf{x}_i, \theta') \log P(\mathbf{x}_i | y_i = 1, \theta) P(y_i = 1 | \theta)] \\ &+ \sum_{i=1}^n [P(y_i = 2 | \mathbf{x}_i, \theta') \log P(\mathbf{x}_i | y_i = 2, \theta) P(y_i = 2 | \theta)] \\ &= \sum_i [\mu_i \log(\lambda p^{n_i} (1-p)^{(3-n_i)}) + (1-\mu_i) \log((1-\lambda) q^{n_i} (1-q)^{(3-n_i)})] \end{aligned}$$

where θ' stands for $\theta^{(t)}$ and $\mu_i = P(y_i = 1 | \mathbf{x}_i, \theta')$. The values of μ_i are known since $\theta' = (\lambda_o, p_o, q_o)$ are given from the previous iteration. The Bayes formula is used to compute μ_i :

$$\begin{aligned} \mu_i &= P(y_i = 1 | \mathbf{x}_i, \theta') = \frac{P(\mathbf{x}_i | y_i = 1, \theta') P(y_i = 1 | \theta')}{P(\mathbf{x}_i | \theta')} \\ &= \frac{\lambda_o p_o^{n_i} (1-p_o)^{(3-n_i)}}{\lambda_o p_o^{n_i} (1-p_o)^{(3-n_i)} + (1-\lambda_o) q_o^{n_i} (1-q_o)^{(3-n_i)}} \end{aligned}$$

We wish to compute: $\max_{p,q,\lambda} Q(\theta, \theta')$. The partial derivative with respect to λ is:

$$\frac{\partial Q}{\partial \lambda} = \sum_i \mu_i \frac{1}{\lambda} - \sum_i (1-\mu_i) \frac{1}{1-\lambda} = 0,$$

from which we obtain the update formula of λ given μ_i :

$$\lambda = \frac{1}{k} \sum_{i=1}^n \mu_i.$$

The partial derivative with respect to p is:

$$\frac{\partial Q}{\partial p} = \sum_i \frac{\mu_i n_i}{p} - \sum_i \frac{\mu_i (3 - n_i)}{1 - p} = 0,$$

from which we obtain the update formula:

$$p = \frac{1}{\sum_i \mu_i} \sum_i \frac{n_i}{3} \mu_i.$$

Likewise the update rule for q is:

$$q = \frac{1}{\sum_i (1 - \mu_i)} \sum_i \frac{n_i}{3} (1 - \mu_i).$$

To conclude, we start with some initial "guess" of the values of p, q, λ , compute the values of μ_i and update iteratively the values of p, q, λ where at the end of each iteration the new values of μ_i are computed.

4.4 Gaussian Mixture

The Gaussian mixture model assumes that $P(\mathbf{x})$ where $\mathbf{x} \in R^d$ is a linear combination of Gaussian distributions

$$P(\mathbf{x}) = \sum_{j=1}^k P(\mathbf{x} | y = j) P(y = j)$$

where

$$P(\mathbf{x} | y = j) = \frac{1}{(2\pi)^{d/2} \sigma_j^d} \exp^{-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma_j^2}},$$

is Normally distributed with mean \mathbf{c}_j and covariance matrix $\sigma_j^2 I$. Let $D = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ be the i.i.d sample data and we wish to solve for the mean and covariances of the individual Gaussians (the "factors") and the mixing coefficients $\lambda_j = P(y = j)$. In order to make clear where the parameters are located we will write $P(\mathbf{x} | \phi_j)$ instead of $P(\mathbf{x} | y = j)$ where $\phi_j = (\mathbf{c}_j, \sigma_j^2)$ are the mean and variance of the j 'th factor. We denote by θ the collection of mixing coefficients λ_j and ϕ_j , $j = 1, \dots, k$. Let w_i^j be auxiliary variables per point x_i and per factor $y = j$ standing for:

$$w_i^j = P(y_i = j | \mathbf{x}_i, \theta).$$

The EM step (eqn. 4.3) is:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta = \{\lambda, \phi\}} \sum_{j=1}^k \sum_{i=1}^m w_i^j \log(\lambda_j P(\mathbf{x}_i | \phi_j)) \quad \text{s.t.} \quad \sum_j \lambda_j = 1. \quad (4.4)$$

Note the constraint $\sum_j \lambda_j = 1$. The update formula for w_i^j is done through the use of Bayes formula:

$$w_i^{j(t)} = \frac{P(y_i = j | \theta^{(t)})P(\mathbf{x}_i | y_i = j, \theta^{(t)})}{P(\mathbf{x}_i | \theta^{(t)})} = \frac{1}{Z_i} \lambda_j^{(t)} P(\mathbf{x}_i | \phi^{(t)}),$$

where Z_i is a scaling factor so that $\sum_j w_i^j = 1$.

The update formula for $\lambda_j, \mathbf{c}_j, \sigma_j$ follow by taking partial derivatives of eqn. (4.4) and setting them to zero. Taking partial derivatives with respect to λ_j, \mathbf{c}_j and σ_j we obtain the update rules:

$$\begin{aligned} \lambda_j &= \frac{1}{m} \sum_{i=1}^m w_i^j \\ \mathbf{c}_j &= \frac{1}{\sum_i w_i^j} \sum_{i=1}^m w_i^j \mathbf{x}_i, \\ \sigma_j^2 &= \frac{1}{d \sum_i w_i^j} \sum_{i=1}^m w_i^j \|\mathbf{x}_i - \mathbf{c}_j\|^2. \end{aligned}$$

In other words, the observations \mathbf{x}_i are weighted by w_i^j before a Gaussian is fitted (k times, one for each factor).

4.5 Application Examples

4.5.1 Gaussian Mixture and Clustering

The Gaussian mixture model is classically used for clustering applications. In a clustering application one receives a sample of points $\mathbf{x}_1, \dots, \mathbf{x}_m$ where each point resides in R^d . The task of the learner (in this case "unsupervised" learning) is to group the m points into k sets. Let $y_i \in \{1, \dots, k\}$ where $i = 1, \dots, m$ stands for the required labeling. The clustering solution is an assignment of values to y_1, \dots, y_m according to some clustering criteria.

In the Gaussian mixture model points are clustered together if they arise from the same Gaussian distribution. The EM algorithm provides a probabilistic assignment $P(y_i = j | x_i)$ which we denoted above as w_i^j .

4.5.2 Multinomial Mixture and "bag of words" Application

The multinomial mixture (the coins example we toyed with) is typically used for representing "count" data, such as when representing text documents as high-dimensional vectors. A vector representation of a text document associates a word from a fixed vocabulary to a coordinate entry of the vector. The value of the entry represents the number of times that particular word appeared in the document. If we ignore the order in which the words appeared and count only their frequency, a set of documents d_1, \dots, d_m and a set of words w_1, \dots, w_n could be jointly represented by a co-occurrence $n \times m$ matrix G where G_{ij} contains the number of times word w_i appeared in document d_j . If we scale G such that $\sum_{ij} G_{ij} = 1$ then we have a distribution $P(w, d)$. This kind of representation of a set of documents is called "bag of words".

For purposes of search and filtering it is desired to reveal additional information about words and documents such as to which "topic" a document belongs to or to which topics a word is associated with. This is similar to a clustering task where documents associated with the same topic are to be

clustered together. This can be achieved by considering the topics as the value of a latent variable y :

$$P(w, d) = \sum_y P(w, d | y)P(y) = \sum_y P(w | y)P(d | y)P(y),$$

where we made the assumption that $w \perp d | y$ (i.e., words and documents are conditionally independent given the topic). The conditional independent assumption gives rise to the multinomial mixture model. To be more specific, let $y \in \{1, \dots, k\}$ denote the k possible topics and let $\lambda_j = P(y = j)$ (note that $\sum_j \lambda_j = 1$), then the latent class model becomes:

$$P(w, d) = \sum_{j=1}^k \lambda_j P(w | y = j)P(d | y = j).$$

Note that $P(w | y = j)$ is a vector which we denote as $\mathbf{u}_j \in R^n$ and $P(d | y = j)$ is also a vector we denote by $\mathbf{v}_j \in R^m$. The term $P(w | y = j)P(d | y = j)$ stands for the outer-product $\mathbf{u}_j \mathbf{v}_j^\top$ of the two vectors, i.e., is a rank-1 $n \times m$ matrix. The Maximum-Likelihood estimation problem is therefore to find vectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ and $\mathbf{v}_1, \dots, \mathbf{v}_k$ and scalars $\lambda_1, \dots, \lambda_k$ such that the empirical distribution represented by the unit scaled matrix G is as close as possible (in relative-entropy measure) to the low-rank matrix $\sum_j \lambda_j \mathbf{u}_j \mathbf{v}_j^\top$ subject to the constraints of non-negativity and $\sum_j \lambda_j = 1$, \mathbf{u}_j and \mathbf{v}_j are unit-scaled as well ($\mathbf{1}^\top \mathbf{u}_j = \mathbf{1}^\top \mathbf{v}_j = 1$).

Let $\mathbf{x}_i = (w(i), d(i))$ stand for the i 'th example $i = 1, \dots, q$ where an example is a pair of word and document where $w(i) \in \{1, \dots, n\}$ is the index to the word alphabet and $d(i) \in \{1, \dots, m\}$ is the index to the document. The EM algorithm involves the following optimization step:

$$\begin{aligned} \theta^{(t+1)} &= \operatorname{argmax}_\theta \sum_{i=1}^q \sum_{j=1}^k P(y_i = j | \mathbf{x}_i, \theta^{(t)}) \log P(\mathbf{x}_i, y_i = j | \theta) \\ &= \operatorname{argmax}_\theta \sum_{i=1}^q \sum_{j=1}^k w_{ij}^{(t)} \log [\lambda_j u_{j,w(i)} v_{j,d(i)}] \quad \text{s.t.} \quad \mathbf{1}^\top \lambda = \mathbf{1}^\top \mathbf{u}_j = \mathbf{1}^\top \mathbf{v}_j = 1 \end{aligned}$$

An update rule for u_{jr} (the r 'th entry of \mathbf{u}_j) is derived below: the derivative of the Lagrangian is:

$$\begin{aligned} \frac{\partial}{\partial u_{jr}} \left[\sum_{i=1}^q w_{ij}^{(t)} \log u_{j,w(i)} - \mu u_{jr} \right] \\ &= \frac{\partial}{\partial u_{jr}} \left[N(r) \log u_{jr} \sum_{w(i)=r} w_{ij}^{(t)} - \mu u_{jr} \right] \\ &= \frac{N(r) \sum_{w(i)=r} w_{ij}^{(t)}}{u_{jr}} - \mu = 0 \end{aligned}$$

where $N(r)$ stands for the frequency of the word w_r in all the documents d_1, \dots, d_m . Note that $N(r)$ is the result of summing-up the r 'th row of G and that the vector $N(1), \dots, N(n)$ is the marginal $P(w) = \sum_d P(w, d)$. Given the constraint $\mathbf{1}^\top \mathbf{u}_j = 1$ we obtain the update rule:

$$u_{jr} \leftarrow \frac{N(r) \sum_{w(i)=r} w_{ij}^{(t)}}{\sum_{s=1}^n N(s) \sum_{w(i)=s} w_{ij}^{(t)}}.$$

Update rules for the remaining unknowns are similarly derived. Once EM has converged, then $\sum_{w(i)=r} w_{ij}^*$ is the probability of the word w_r to belong to the j 'th topic and $\sum_{d(i)=s} w_{ij}^*$ is the probability that the s 'th document comes from the j 'th topic.