A FAST ALGORITHM FOR DISCRETE HMM TRAINING USING OBSERVED TRANSITIONS

Madhusudana Shashanka

United Technologies Research Center, East Hartford, CT 06108
shashanka@alum.bu.edu

ABSTRACT

We present a new algorithm to estimate the parameters of a Hidden Markov Model (HMM), specifically the transition probability matrix of the hidden states and the emission probabilities, given an observed sequence of data. The algorithm uses the number of transitions present in the observed label sequence and computes the parameters in an iterative fashion. We present experiments that demonstrate significant speed gains obtained by the current algorithm as compared to traditional algorithms such as Baum-Welch iterations.

Index Terms— Hidden Markov Models, EM Algorithm

1. INTRODUCTION

Hidden Markov Models (HMMs) have found wide use in modeling temporal patterns in a variety of fields such as speech recognition, computational linguistics and bio-informatics. HMMs model an observed sequence of data as the outputs of a sequence of hidden states where it is assumed that the chain of hidden states is a Markov process. The applicability of HMMs to practical problems is possible because of the existence of methods such as the Baum-Welch reestimation procedure to estimate HMM parameters given a data sequence. A great overview of HMMs and applications can be found in a tutorial by Rabiner [1].

The motivation for this work arises from situations where application of HMM is ideal but impractical because of excessive demands on computational resources. For example, [2] point out the high computational cost of HMM training in the context of intrusion detection because of long sequences. Another such situation is where one wants to learn a single HMM model from multiple sequences [3] in which case parameter estimation becomes computationally expensive. The reason for this computational complexity comes from the way Baum-Welch algorithm works where computations are done at every step of the sequence and this process is repeated over several iterations.

We propose an alternate method to estimate HMM parameters where we eliminate the need for computations at every step of the sequence in every iteration. Observations are represented as a matrix of transition counts and the proposed algorithm estimates the parameters using this reduced representation of the data sequence. We propose a generative model for the new data representation and derive update rules for parameters using the Expectation-Maximization algorithm. Experiments on synthetic data demonstrate the superiority of the proposed algorithm in terms of computational complexity.

In Section 2, we present some background and set the notations used in the rest of the paper. We describe our algorithm in Section 3 and discuss related work in Section 4. Section 5 describes experimental results and we conclude the paper in Section 6.

2. HIDDEN MARKOV MODELS

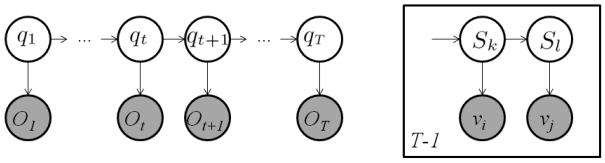
Notations used in this paper follow conventions used by [1]. Let the parameters that define an HMM be given by the following:

- N, the number of hidden states in the model. Let the set of states be denoted by $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ and let the state at time step t be denoted by q_t.
- M, the number of distinct observable labels per state, i.e. the discrete alphabet size. Let the set of labels be denoted by $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$.
- The state transition probability distribution $\mathbf{A} = \{a_{ij}\}$ where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \leqslant i, j \leqslant N$.
- The observation label probability distribution in state i, $\mathbf{B} = \{b_i(k)\}$, where $b_i(k) = P(v_k \text{ at } t | q_t = S_i), \quad 1 \leqslant i \leqslant N, 1 \leqslant k \leqslant M$.
- The initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i), 1 \leqslant i \leqslant N$.

A complete specification of an HMM requires specification of two model parameters N and M, specification of observation labels, and specification of the three probability measures $\mathbf{A}$, $\mathbf{B}$ and π. In the rest of the paper, we use the compact notation $\lambda = \{\mathbf{A}, \mathbf{B}, \pi\}$ to indicate the complete parameter set of the model.

Let $\bar{O} = O_1 O_2 \ldots O_T$ indicate a sequence of T observation labels. Rabiner [1] points out three basic problems of interest for HMMs: (a) given observations $\bar{O}$ and model λ, how do we efficiently compute $P(\bar{O}|\lambda)$? (b) given the observation sequence $\bar{O}$ and model λ, how do we choose a hidden state sequence that best explains the observed sequence? and (c) how do we adjust model parameters λ to maximize the likelihood $P(\bar{O}|\lambda)$?

This paper focuses on the last problem which is the problem of *training* an HMM. It is by far the most difficult problem and there is no known way to analytically solve for the values of parameters λ that maximizes the likelihood. Existing methods use iterative procedures that guarantee that the likelihood is locally maximized. Examples of such methods include the Baum-Welch algorithm [1] and gradient-descent techniques [4]. These methods use a *forward-backward* procedure where intermediate *forward* and *backward* variables are computed at every step of the observed label sequence and are used to compute new estimates of the HMM parameters in a given iteration. The whole process is repeated iteratively until a local maximum is reached. These methods are quite efficient compared to the naive method of computing all possible probabilities at every step of the observation label sequence. However, this iterative procedure can be significantly slow if the observed sequence is quite large and in certain applications can make HMMs impractical to use.



(a) (b)

Fig. 1. Graphical Models for (a) HMM, and (b) the generative model we use. In (b), each draw corresponds to a pair of observations in the sequence and the draws are repeated $T-1$ times.

3. ALGORITHM

In this section, we propose a new algorithm to train HMM parameters that is significantly faster than the traditional techniques used in HMM training. Instead of using the entire observed label sequence as the training data, the algorithm works on the number of transitions between every pair of labels that was observed in the sequence.

Let $\bar{O} = O_1 O_2 \ldots O_T$ denote the observed sequence of labels. Let the transitions between the observed labels be given by the $M \times M$ count matrix $\mathbf{C}$, *i.e.* the entry c_{ij} indicates the number of times the consecutive pair $\{v_i, v_j\}$ was observed in $\bar{O}$. Let the probability vector $\mathbf{p} = \{p_1, p_2, \ldots, p_N\}$ indicate the steady state probabilities of the hidden states.

We propose to use the matrix $\mathbf{C}$ to characterize the probability of observing a particular $O_t O_{t+1}$ pair in sequence in $\bar{O}$. The idea is to formulate this probability in terms of λ and then use Expectation-Maximization (EM) algorithm for this model to learn λ.

3.1. Generative Model

Let us first consider the generative process for $\mathbf{C}$. $\mathbf{C}$ is assumed to be the histogram of an independent set of $T-1$ draws from an underlying probability distribution over the $M \times M$ pairs of $\{v_i, v_j\}$. Let the matrix $\bar{\mathbf{C}}$ represent this probability distribution where the ij-th entry $\bar{c}_{ij}$ represents the probability of observing the label pair $\{v_i, v_j\}$ in successive timesteps.

Consider a given draw where timesteps corresponding to the label-pair drawn are $\{t, t+1\}$. For this draw, (1) we first choose a hidden state $q_t = S_k$ with probability p_k, (2) we observe a label v_i with probability $b_k(i) = P(v_i | q_t = S_k)$, (3) we choose the subsequent hidden state $q_{t+1} = S_l$ with probability a_{kl}, and (4) we observe the second label v_j with probability $b_l(j) = P(v_j | q_{t+1} = S_l)$. Figure 1 illustrates the graphical model.

Given the above steps, it is easy to see that the joint distribution of the hidden state pair and the observed label pair is given by $P(q_t = S_k, q_{t+1} = S_l, O_t = v_i, O_{t+1} = v_j) = p_k \cdot b_k(i) \cdot a_{kl} \cdot b_l(j)$. Summing the above joint distribution over all possible hidden state pairs, we obtain the model for probability $P(O_t = v_i, O_{t+1} = v_j)$ as $\bar{c}_{ij} = \sum_{k=1}^{N} \sum_{l=1}^{N} p_k \cdot b_k(i) \cdot a_{kl} \cdot b_l(j)$. Writing $p_k \cdot a_{kl}$ as $\bar{a}_{kl}$ - the joint state probability distribution, we write the model as

$$\bar{c}_{ij} = \sum_{k=1}^{N} \sum_{l=1}^{N} b_k(i) \cdot \bar{a}_{kl} \cdot b_l(j) \quad \text{i.e.} \quad \bar{\mathbf{C}} = \mathbf{B}\bar{\mathbf{A}}\mathbf{B}^T. \qquad (1)$$

3.2. Parameter Estimation

We use the Expectation Maximization algorithm to estimate the HMM parameters $\mathbf{A}$ and $\mathbf{B}$. EM alternates two steps: (1) an Expectation (E) step where the *a posteriori* probabilities of the latent variables are computed based on the current estimates of parameters, and (2) a maximization (M) step, where parameters are updated such that the expected complete data log-likelihood is maximized.

In the model given by equation (1), the hidden variables are S_k and S_l. For the E-step, we obtain the *a posteriori* probability for the latent variables as

$$P_{klij} = P(S_k, S_l | v_i, v_j) = \frac{b_k(i) \cdot \bar{a}_{kl} \cdot b_l(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} b_k(i) \cdot \bar{a}_{kl} \cdot b_l(j)}. \qquad (2)$$

In the M-step, we maximize the expected complete data log-likelihood. The equations are

$$\begin{aligned} \bar{a}_{kl} &= \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} P_{klij}}{\sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} \sum_{k=1}^{N} \sum_{l=1}^{N} P_{klij}}, \\ b_k(i) &= \frac{\sum_{j=1}^{M} c_{ij} \sum_{l=1}^{N} P_{klij} + \sum_{i=1}^{M} c_{ij} \sum_{k=1}^{N} P_{klij}}{\sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} \Big(\sum_{l=1}^{N} P_{klij} + \sum_{k=1}^{N} P_{klij} \Big)}. \end{aligned} \qquad (3)$$

Iterating through equations (2) and (3) is guaranteed to provide converging estimates of the parameters. a_{kl} can be computed from $\bar{a}_{kl}$ using the relation $a_{kl} = \bar{a}_{kl} / (\sum_{l=1}^{N} \bar{a}_{kl})$. Details of the derivation are provided in the appendix.

3.3. Algorithm and Complexity

The algorithm can be summarized as follows. Symbols $\odot$ and $\oslash$ indicate element-wise multiplication and division respectively.

```
Input: Matrix C
Initialize Ā and B
Iterate
  R ← C ⊘ (BĀB^T)
  Ā' ← Ā ⊙ (B^T RB),  B' ← B ⊙ (RBĀ^T + R^T BĀ)
  Ā ← Normalize all entries of Ā'
  B ← Normalize columns of B'
End Iterations
A ← Normalize rows of Ā.
```

In each iteration, there are eight matrix multiplications - four with $O(MN^2)$ and four with $O(M^2N)$ operations. Also, there are MN element-wise divisions and $MN + N^2$ element-wise multiplications leading to $O(MN^2 + M^2N + MN + N^2)$ operations in all. Since $N \ll M$ in practice, the complexity of the proposed algorithm is $O(IM^2N)$ where I is the total number of iterations. If we include the step to compute the matrix $\mathbf{C}$ from the sequence, the complexity is $O(IM^2N + T)$.

In comparison, the complexity of Baum-Welch algorithm is given by $O(IN^2T)$ where T is the length of the sequence. If $T \gg M$, the our algorithm provides significant speed gains and this is demonstrated in experiments.

4. RELATED WORK

The proposed algorithm is an approximation method since higher-order transitions in the observed sequence are ignored. This kind of approximation is used in the control theory literature in the context of optimal predictor chains and Mori-Zwanzig representations [5]. Specifically, non-Markovian systems can be represented as a reduced model with several terms where the leading-order term represents the Markovian aspect while other terms capture history and noise. When the non-Markovian system is a HMM, the first term is a Markov chain that optimally represents the HMM. It can be shown that the

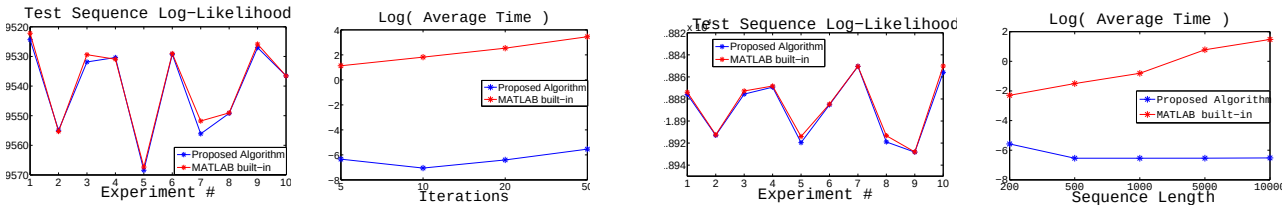


Fig. 2. Comparison of the proposed algorithm with Baum-Welch Algorithm on a sequence of length 5000 and different number of iterations (left two panels), and various sequence lengths with 20 iterations (right two panels). We ran 10 different experiments where we learned the HMM parameters from the two algorithms using the same initial values. Using the learned parameters, we fit a different test sequence and the resulting log-likelihoods are plotted (panels 1 and 3). We see that the proposed algorithm outperforms the Baum-Welch algorithm by several orders of magnitude in the average run-time (panels 2 and 4).

joint-state distribution of this "optimal predictor chain" is given by Equation (1). In [5], the goal is to estimate the parameters of the optimal Markov chain in terms of the HMM parameters while our goal is to solve the inverse problem of estimating the HMM parameters using the approximation model of Equation (1).

In the last few years, several researchers have proposed algorithms for HMM learning as alternatives to Baum-Welch. One approach is to re-parametrize the problem. [6] proposes an efficient SVD based approach while [7,8] propose using split-data likelihoods for recursive estimation by considering blocks of different lengths (instead of pairs as considered in this paper). The other approach is to trade off estimation accuracy for algorithmic efficiency. Examples include pairwise likelihoods [9] and composite marginal likelihoods [10] for general state space models. Our work is more related to the latter approach.

The author was made aware of a recent paper [11] that takes a similar approach to ours. The authors propose to use observed transitions and use a model similar to Equation (1). However, they use Nonnegative Matrix Factorization with a least-squares criterion for parameter updates while we use EM algorithm which is equivalent to using KL-divergence as the criterion. They use a kernel based approach to extend their algorithm to continuous observations and the same method is applicable to our algorithm as well. However, we skip the details due to space constraints.

5. EXPERIMENTS

In this section, we describe experiments that compare the traditional Baum-Welch algorithm with our proposed approach. There were two main goals for the experiments: (a) compare the accuracy (fit) of the learned parameters in terms of log-likelihood, and (b) compare the runtimes for the algorithms. All algorithms were implemented in the MATLAB environment. For Baum-Welch algorithm, we used the built-in functions available in MATLAB.

For the first data set, we chose 3 hidden states and 7 observation labels. We first generated a sequence of observations to serve as training data. We trained parameters $\mathbf{A}$ and $\mathbf{B}$ using our algorithm as well as the Baum-Welch algorithm. In the first part, we chose four cases where we set the number of iterations to 5, 10, 20 and 50 respectively. For each case, we ran ten "runs" where the initial values of $\mathbf{A}$ and $\mathbf{B}$ (to be used by both algorithms) were chosen randomly at the beginning of the run. The parameters were learned by each method and was used to fit a test sequence of length 5000. In the second part, we fixed the number of iterations to 20 but varied the length of the test sequence and chose 5 cases corresponding to sequence lengths of 200, 500, 1000, 5000 and 10000. For each case, parameters learned by both algorithms were used to fit the test sequence and ten "runs" were performed with different initializations. Figure 2 summarizes the results.

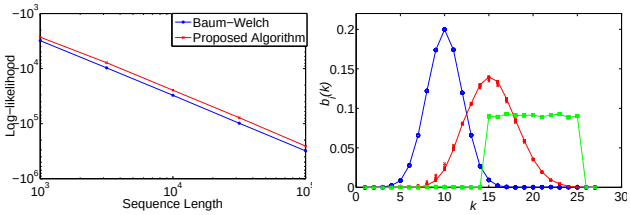


Fig. 3. Log-likelihoods (left-panel) obtained by running the algorithms on dataset used by [11]. The right-panel illustrates the recovered observation probability distributions by our algorithm repeated over 10 experiments.

For the second data set, we chose to repeat the experiment conducted in [11] where they chose a model with 3 hidden states. The transition matrix $\mathbf{A}$ was set to $[(0, 0.9, 0.1), (0, 0, 1), (1, 0, 0)]$. The observations were created in two steps. First, values v' were created according to the following models - $b_1(v') \sim \mathcal{N}(11, 2)$, $b_2(v') \sim \mathcal{N}(16, 3)$ and $b_3(v') \sim U(16, 26)$. The generated values were rounded to the nearest integer to obtain observations v. Sequences of different lengths were generated spanning from 10^3 to 10^5. We ran both the Baum-Welch algorithm and our algorithm on the dataset and the results are summarized in Figure 3. We can see that the proposed algorithm recovers the observed probability distributions correctly as shown in the right panel and the log-likelihoods obtained are marginally superior to the log-likelihoods from Baum-Welch.

Our results clearly demonstrate the utility of the approach compared to the Baum-Welch algorithm. We obtain significant advantages in time-complexity without trading off accuracy of the results.

We finally describe a third experiment to illustrate another aspect of the algorithm that cannot be handled by Baum-Welch algorithm. Consider a scenario where instead of observing discrete labels, we are given a probability of having observed each label in the vocabulary. In other words, at every step, we observe a vector of probabilities. Let $\mathbf{X}$ represent this matrix over several timesteps. We can obtain the observed transition matrix $\mathbf{C}$ by taking the matrix product between $\mathbf{X}$ and $\mathbf{X}^T$ where we shift the columns of the matrix by one-step before transposing. Once we have the matrix $\mathbf{C}$, we can proceed as before by applying our algorithm. However, we cannot apply Baum-Welch algorithm to data of this type. To illustrate this, we generated a synthetic dataset of 400 vectors where the vectors alternate between one of two profiles every 50 steps. This matrix $\mathbf{X}$ is shown on Panel (a) of Figure 4. 98% of the transitions are from one profile to the same profile while the profiles switch in 2% of the transitions. Panel (b) shows the observed transition matrix $\mathbf{C}$ obtained from $\mathbf{X}$. Panels (c) and (d) show the observed probability distributions and the transition matrix respectively obtained from running our algorithm on $\mathbf{C}$. As we can see, the observed emission probabilities correspond to the vectors present in $\mathbf{X}$ and the transition matrix correctly reflect the transitions in $\mathbf{X}$.

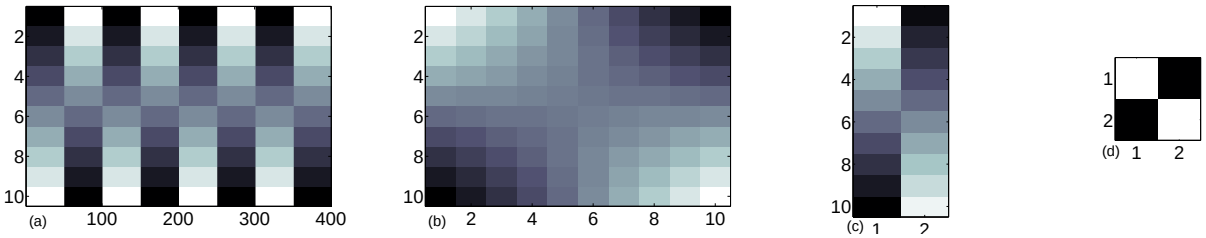


Fig. 4. Illustration of our algorithm applied to a sequence of vectors (Panel (a)). Our algorithm is applied on the observed transition matrix (Panel (b)) that is derived from the input. The learned emission probability distributions (Panel (c)) and transition matrix (panel (d)) are also shown.

6. CONCLUSIONS

We have proposed a novel algorithm to train the parameters of a discrete Hidden Markov Model. Instead of the given data sequence, we use an alternate representation of transition counts that is more concise. We described the generative model used and derived equations for parameter estimation. We presented the algorithm and described the time-complexity. We conducted experiments on synthetic data and compared our results with results from implementations of the Baum-Welch algorithm. The results demonstrate the effectiveness of the proposed approach and speed gains one can obtain. The simplicity, accuracy and efficiency of the proposed algorithm makes it applicable in a wide range of scenarios and in contexts where traditional methods are too cumbersome (large sequence lengths and multiple sequences). In future work, we will extend this work to handle continuous observations and generalize the framework to handle blocks of arbitrary sizes instead of pairs.

7. REFERENCES

[1] L R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, Feb 1989.

[2] XA Hoang and J Hu, "An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls.," in *IEEE Intl. Conf. on Networks*, 2004, vol. 2, pp. 470–474.

[3] R I A Davis and B C Lovell, "Improved estimation of hidden markov model parameters from multiple observation sequences," in *Intl. Conf. on Pattern Recognition*, 2002, pp. 168–171.

[4] S E Levinson, L R Rabiner, and M M Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *Bell Sys. Tech. Journal*, vol. 62, no. 4, 1983.

[5] CL Beck, S Lall, T Liang, and M West, "Model Reduction, Optimal Prediction, and the Mori-Zwanzig Representation of Markov Chains," in *IEEE Conf. on Decision and Control*, 2009.

[6] D Hsu, SM Kakade, and T Zhang, "A Spectral Algorithm for learning Hidden Markov Models," in *Conf. on Learning Theory*, 2009.

[7] T Ryden, "Consistent and Asymptotically Normal Parameter Estimates for Hidden Markov Models," *Ann. Statistics*, vol. 22, no. 4, 1994.

[8] T Ryden, "On Recursive Estimation for Hidden Markov Models," *Stoch. Processes and Applications*, vol. 66, 1997.

[9] C Varin and P Vidoni, "Pairwise Likelihood Inference for General State Space Models," *Econometric Reviews*, vol. 28, 2009.

[10] C Varin, "On composite marginal likelihoods," *Adv. in Stat. Anal.*, vol. 92, 2008.

[11] B Lakshminarayanan and R Raich, "Nonnegative Matrix Factorization for Parameter Estimation in Hidden Markov Models," in *Workshop on Machine Learning for Signal Processing*, 2010.

8. APPENDIX: PARAMETER ESTIMATION

In this appendix, we derive update equations for parameters of the model given by equation (1) using EM algorithm. For the E-step, we obtain the *a posteriori* probability for latent variables S_k and S_l as shown in equation (2).

In the M-step, we maximize the expected complete data log-likelihood. Let Λ represent the set of parameters of the model, i.e. $\Lambda = \{\mathbf{A}, \mathbf{B}\}$. The expected log-likelihood can be written as $\mathcal{L} = E_{\bar{S}|\bar{v};\Lambda} \log P(\bar{S}, \bar{v})$, where $\bar{S}$ and $\bar{v}$ represent the set of all draws of the hidden state pairs (S_k, S_l) and label pairs (v_i, v_j) respectively, and

$$P(\bar{S}, \bar{v}) \propto \prod_d P(S_{k_d}, S_{l_d}, v_{i_d}, v_{j_d}) = \prod_d b_{k_d}(i_d) \cdot \bar{a}_{k_d l_d} \cdot b_{l_d}(j_d).$$

In the above equation, the subscript d indicates the d-th draw where hidden state pair (S_{k_d}, S_{l_d}) and label pair (v_{i_d}, v_{j_d}) are picked. Now, we expand the log-likelihood as

$$\begin{aligned}
\mathcal{L} &= E_{\bar{S}|\bar{v};\Lambda} \sum_d \log P(S_{k_d}, S_{l_d}, v_{i_d}, v_{j_d}) \\
&= \sum_d E_{(S_{k_d}, S_{l_d})|(v_{i_d}, v_{j_d});\Lambda} \log \Big(b_{k_d}(i_d) \cdot \bar{a}_{k_d l_d} \cdot b_{l_d}(j_d) \Big) \\
&= \sum_d \sum_{k=1}^{N} \sum_{l=1}^{N} P_{kli_d j_d} \log \Big(b_k(i_d) \cdot \bar{a}_{kl} \cdot b_l(j_d) \Big).
\end{aligned}$$

Changing the summation over draws d to a summation over labels by accounting for how many times each label-pair occurred (entries of $\mathbf{C}$), we can write $\mathcal{L}$ as

$$\mathcal{L} = \sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} \sum_{k=1}^{N} \sum_{l=1}^{N} P_{klij} \Big(\log b_k(i) + \log \bar{a}_{kl} + \log b_l(j) \Big). \quad (4)$$

In order to take care of the normalization constraints, the above equation must be augmented by appropriate Lagrange multipliers τ_k and ρ_k as follows,

$$\mathcal{Q} = \mathcal{L} + \sum_{k=1}^{N} \tau_k \Big(1 - \sum_{i=1}^{M} b_k(i) \Big) + \rho \Big(1 - \sum_{k=1}^{N} \sum_{l=1}^{N} \bar{a}_{kl} \Big). \quad (5)$$

Maximizing $\mathcal{Q}$ with respect to $\bar{a}_{kl}$ and $b_k(i)$ leads to the following set of equations:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} c_{ij} P_{klij} = \rho \bar{a}_{kl}, \quad (6)$$

$$\sum_{j=1}^{M} c_{ij} \sum_{l=1}^{N} P_{klij} + \sum_{i=1}^{M} c_{ij} \sum_{k=1}^{N} P_{klij} = \tau_k b_k(i). \quad (7)$$

Using the fact that $\sum_{k=1}^{N} \sum_{l=1}^{N} \bar{a}_{kl} = 1$ and $\sum_{i=1}^{M} b_k(i) = 1$, we can eliminate the Lagrange multipliers from the above equations to obtain the M-step re-estimation equations as shown in equations (3). a_{kl} can be computed from $\bar{a}_{kl}$ using the relation $a_{kl} = \bar{a}_{kl} / (\sum_{l=1}^{N} \bar{a}_{kl})$. The final update equations are given by equations (2) and (3) and iterating through them is guaranteed to provide converging estimates of the parameters.