Efficient Probability Mass Function Estimation from Partially Observed Data

Joseph K. Chege[1], *Mikus J. Grasis*[1], *Alla Manina*[1], *Arie Yeredor*[2], *and Martin Haardt*[1]
[1]Communications Research Laboratory, Ilmenau University of Technology, Ilmenau, Germany
[2]School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel

***Abstract*—Estimating the joint probability mass function (PMF) of a set of random variables from partially observed data is a crucial part of statistical learning and data analysis, with applications in areas such as recommender systems and data classification. Recently, it has been proposed to estimate the joint PMF based on the maximum likelihood (ML) of the data, fitted to a low-rank canonical polyadic decomposition (CPD) model of the joint PMF. To this end, a hybrid alternating-directions expectation-maximization (AD-EM) algorithm was proposed to solve the ML optimization problem, consisting of computationally expensive AD iterations followed by an EM refinement stage. It is well known that the convergence rate of EM decreases as the fraction of missing data increases. In this paper, we address the slow convergence of the EM algorithm. By adapting the squared iterative methods (SQUAREM) acceleration scheme to the context of PMF estimation, we propose the SQUAREM-PMF algorithm to speed up the convergence of the EM algorithm. Moreover, we demonstrate that running the computationally cheaper EM algorithm alone after an appropriate initialization is sufficient. Numerical results on both synthetic and real data in the context of movie recommendation show that our algorithm outperforms state-of-the-art PMF estimation algorithms.**

***Index Terms*—Joint PMF estimation, tensor decomposition, maximum likelihood (ML), expectation-maximization (EM), recommendation systems.**

I. Introduction

Estimating the probability mass function (PMF) from partial observations of a discrete random vector (RV) is a core problem in statistics, which has recently been considered in a variety of disciplines, ranging from signal processing and machine learning to econometrics and social sciences. Having access to the joint PMF has been considered the gold standard in categorical data analysis [1] as it allows to compute any marginal or conditional distribution of the involved variables. Based on observing a realization of some of its elements, classical estimates, such as the maximum *a posteriori* (MAP) or minimum mean square error (MMSE) estimates of the missing elements (or features) are readily available. The necessity arises quite commonly in practice. For instance, in data classification, a set of observed features is provided and the task is to predict the label corresponding to these features. Alternatively, consider recommender systems, e.g., for movies. In this case, the core database contains a "K users by N movies" rating matrix, which contains, for each of the K users, ratings (entered by that user) of some (usually very few) of the N movies. The goal is to infer, based on that partially filled matrix, how a given user would rate each of the movies which she/he has not watched, in order to recommend to her/him a new movie for which her/his projected rating is high.

In practice, however, the full PMF, which can be structured as an N-way tensor, is rarely known, and needs to be estimated from the core database during a "training" or a "learning" stage. Unfortunately, this is not a trivial task, since using straightforward histogram-type estimation of the tensor to within a reasonable accuracy quickly becomes intractable as the required amount of observed data grows exponentially with the tensor's order N. Having partly observed realizations only exacerbates the situation as in this case histogram-based estimates require an even larger amount of observed data, so as to properly cover all possible co-occurrences.

Using tensor algebra, the authors in [1] have shown that if reliable estimates of marginal distributions of order greater than or equal to 3 are available, the full joint PMF can be recovered. Based on their finding that a joint PMF can always be represented by a naïve Bayes model (provided the rank of the canonical polyadic decomposition (CPD) model is high enough), the authors show how the complete joint PMF can be recovered from the marginals via a coupled non-negative tensor factorization approach. The resulting optimization problem is solved by an alternating optimization (AO) scheme based on the alternating direction method of multipliers (ADMM) and is formulated in the ordinary least-squares (LS) sense.

While LS-based methods are conducive for algorithm design, they are not naturally suitable for measuring distances between PMF tensors. In particular, it is not severely penalized when attributing an extremely small (or even zero) probability to certain elements of the estimated PMF, even when the empirical evidence may suggest that the respective vector values are feasible. Therefore, the authors in [2] proposed to replace the LS criterion in the approximate coupled factorization in [1] by the Kullback-Leibler divergence (KLD). Moreover, the follow-up work [3] showed that when a KLD-based criterion is applied directly to all of the observed partial data (rather than only to estimated subtensors of a fixed order), the resulting PMF estimate coincides with the maximum likelihood (ML) estimate. The resulting problem is then solved via an alternating directions (AD) scheme. Additionally, it is shown how to employ expectation-maximization (EM, [4]) for obtaining the ML estimate.

While the EM-based algorithm is computationally cheaper than an AD-type optimization algorithm, theoretical results have shown that the convergence rate of EM decreases as the

fraction of missing data increases [5]. In addition, the authors in [3] observed that EM may be sensitive to initialization. The authors therefore proposed a hybrid iterative computation scheme, which runs a few AD-type optimization steps first in order to reach a "sufficiently close" initialization point for the EM algorithm and then proceeds with iterations of the computationally cheaper EM algorithm.

In this paper, we address the slow convergence of the EM algorithm proposed in [3]. By adapting the SQUAREM method [6] to the context of PMF estimation, we propose the SQUAREM-PMF algorithm to accelerate the convergence of the EM algorithm. Additionally, we give a concise summary on the mathematical rationale of the method to provide the intuition behind the speed-ups. We demonstrate through carefully designed synthetic-data simulations that SQUAREM-PMF accelerates the convergence of EM. Moreover, we show that the initial iterations of the AD-type scheme are not necessary when initializing the EM iterations with factor matrices and component weights drawn from a uniform distribution and with appropriate normalization to the probability simplex. We further test EM and SQUAREM-PMF on real data and show that SQUAREM-PMF performs comparably to EM and outperforms the AO-ADMM method from [1].

A. Notation

A scalar, a vector, a matrix and a tensor are denoted by x, $\boldsymbol{x}$, $\boldsymbol{X}$ and $\boldsymbol{\mathcal{X}}$, respectively. X and Y represent N-dimensional random vectors. Further, we denote the transpose operator and the Euclidean norm of a vector as $^\mathsf{T}$ and $\|\cdot\|$, respectively. The vertical stacking of the columns of $\boldsymbol{X}$ into a column vector is denoted by $\mathrm{vec}(\boldsymbol{X})$, while the $p \times p$ identity matrix is denoted by $\boldsymbol{I}_p$.

II. Preliminaries

In order to describe our improved method, we first introduce the considered data model, the minimized objective function and the update equations for the EM scheme derived in [3].

Given a discrete random vector $X = [X_1, \dots, X_N]^\mathsf{T} \in \mathbb{R}^N$ with X_n taking discrete integer values in $[1, I_n]$ for $n = 1, \dots, N$, its joint PMF is described by a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, where $\boldsymbol{\mathcal{X}}(i_1, \dots, i_N) = \mathsf{Pr}\{X_1 = i_1, \dots, X_N = i_N\}$. In practice, $\boldsymbol{\mathcal{X}}$ often admits a low-rank CPD $\boldsymbol{\mathcal{X}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}_1, \dots, \boldsymbol{A}_N]\!]$ with R components, with the factors as well as the vector $\boldsymbol{\lambda}$ confined to the probability simplex [1]. As in the previous work [3], we model the case where the variable X_n is unobserved with probability p (independently among all variables), resulting in the (partially) observed random vector $Y = [Y_1, \dots, Y_N]^\mathsf{T}$ with entries

$$Y_n = \begin{cases} X_n & \text{w.p. } 1-p \\ 0 & \text{w.p. } p \end{cases}, \quad n = 1, \dots, N. \tag{1}$$

Our goal is to estimate $\boldsymbol{\mathcal{X}}$ given a finite number T of i.i.d. realizations of Y.

The probability of observing a particular realization $\boldsymbol{y}$ of Y is given by

$$\begin{aligned} \mathsf{Pr}\{\boldsymbol{y}\} &= q \cdot \mathsf{Pr}\{X_{n_1} = y_{n_1}, \dots, X_{n_B} = y_{n_B}\} \\ &= q \cdot \sum_{r=1}^{R} \lambda_r \boldsymbol{A}_{n_1}(y_{n_1}, r) \cdots \boldsymbol{A}_{n_B}(y_{n_B}, r), \end{aligned} \tag{2}$$

where $B \in [0, N]$ denotes the number of nonzero elements of $\boldsymbol{y}$, $n_1, \dots, n_B$ denotes their respective indices, and $q = p^{N-B}(1-p)^B$. The remaining $N - B$ factors inside the sum vanish by marginalization of the N observed elements [1]. Given T i.i.d. observations $\boldsymbol{y}[1], \dots, \boldsymbol{y}[T]$, their joint log-likelihood function takes the form

$$\begin{aligned} &\log \mathsf{Pr}\{\boldsymbol{y}[1], \dots, \boldsymbol{y}[T]; \boldsymbol{\theta}\} \\ &\quad = \sum_{t=1}^{T} \log \sum_{r=1}^{R} \lambda_r \prod_{b=1}^{B[t]} \boldsymbol{A}_{n_b[t]}(y_{n_b[t]}[t], r) + c, \end{aligned} \tag{3}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\lambda}, \boldsymbol{A}_1, \dots, \boldsymbol{A}_N\}$ represents the model parameters, $B[t]$ is the number of nonzero (observed) elements in $\boldsymbol{y}[t]$, $n_1[t], \dots, n_{B[t]}[t]$ are their indices, and, finally, $c = \sum_{t=1}^{T} \log q[t]$ is a constant which is irrelevant to the maximization of (3). The optimization problem for ML estimation is therefore given by

$$\begin{aligned} \min_{\{\boldsymbol{A}_n\}_{n=1}^N, \boldsymbol{\lambda}} \; & -\sum_{t=1}^{T} \log \sum_{r=1}^{R} \lambda_r \prod_{b=1}^{B[t]} \boldsymbol{A}_{n_b[t]}(y_{n_b[t]}[t], r) \\ \text{subject to} \quad & \boldsymbol{\lambda} > \boldsymbol{0}, \quad \boldsymbol{1}^\mathsf{T}\boldsymbol{\lambda} = 1 \\ & \boldsymbol{A}_n \geq \boldsymbol{0}, \quad \boldsymbol{1}^\mathsf{T}\boldsymbol{A}_n = \boldsymbol{1}^\mathsf{T}, \quad n = 1, \dots, N. \end{aligned} \tag{4}$$

The EM algorithm consists of two steps. In the E-Step, the *a posteriori* distribution of the latent variable given the current observations $\boldsymbol{y}[1], \dots, \boldsymbol{y}[T]$ and parameters $\boldsymbol{\theta}^{(k)}$ (at the k-th iteration) is computed. By defining $\boldsymbol{z}[t] = [\boldsymbol{y}^\mathsf{T}[t], s[t]]^\mathsf{T} \in \mathbb{R}^{N+1}$, consisting of the observed vectors, augmented by the unknown Bayesian state $s[t] \in \{1, \dots, R\}$ from which the complete vector $\boldsymbol{x}[t]$ is drawn, we find

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)}) &= \mathbb{E}\{\boldsymbol{z}[1], \dots, \boldsymbol{z}[T]; \boldsymbol{\theta} \,|\, \boldsymbol{y}[1], \dots, \boldsymbol{y}[T]; \boldsymbol{\theta}^{(k)}\} \\ &= \sum_{r=1}^{R} \left(C_r^{(k)} \log \lambda_r + \sum_{n=1}^{N} \sum_{i=1}^{I_n} K_r^{(k)}(n, i) \log \boldsymbol{A}_n(i, r) \right), \end{aligned} \tag{5}$$

where $c_{t,r}(\boldsymbol{\theta}^{(k)}) = \mathsf{Pr}(s[t] = r \,|\, \boldsymbol{y}[t]; \boldsymbol{\theta}^{(k)})$, $C_r^{(k)} = \sum_{t=1}^{T} c_{t,r}(\boldsymbol{\theta}^{(k)})$ and $K_r^{(k)}(n, i) = \sum_{t: y_n[t] = i} c_{t,r}(\boldsymbol{\theta}^{(k)})$, where the latter is a summation over the observed indices. In the M-Step, $\boldsymbol{\theta}^{(k)}$ is updated by the parameter values which maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)})$. The maximizing solutions can be shown to admit a closed form consisting of simple divisions [3], i.e.,

$$\lambda_r^{(k+1)} = \frac{C_r^{(k)}}{\sum_{g=1}^{R} C_g^{(k)}} \quad \text{and} \quad \boldsymbol{A}_n^{(k+1)}(i, r) = \frac{K_r^{(k)}(n, i)}{\sum_{j=1}^{I_n} K_r^{(k)}(n, j)}. \tag{6}$$

III. Proposed Solution

The EM algorithm is attractive due to its computationally cheap closed-form updates as well as its stability since it guarantees a monotonic increase in the likelihood. However, the algorithm typically exhibits linear convergence whose rate

is inversely related to the proportion of missing information in the observed data [5]. This means that if a large portion of data is missing, the convergence may be quite slow. In this section, we describe the SQUAREM algorithm proposed in [6] to accelerate EM convergence and propose an adaptation procedure that allows the algorithm to be applicable in the context of PMF estimation.

A. The EM Fixed Point

The fixed point of a function is defined as an element of the function's domain that is mapped to itself by the function. Concretely, if $f(x) = x$ for a function f, then x is a fixed point of f. Let $\boldsymbol{\theta} \in \boldsymbol{\Omega} \subseteq \mathbb{R}^p$ be a parameter vector in the space $\boldsymbol{\Omega}$. The EM algorithm implicitly defines a mapping $\boldsymbol{\theta} \mapsto \boldsymbol{f}(\boldsymbol{\theta})$ to itself such that

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{f}(\boldsymbol{\theta}^{(k)}) \in \mathbb{R}^p, \quad k = 0, 1, 2, \ldots \tag{7}$$

where $\boldsymbol{f}$ is the EM mapping and k is the current iteration. Thus, if $\boldsymbol{\theta}^{(k)}$ converges to the point $\boldsymbol{\theta}_{\text{opt}}$, then $\boldsymbol{\theta}_{\text{opt}}$ is a fixed point of the algorithm since $\boldsymbol{\theta}_{\text{opt}} = \boldsymbol{f}(\boldsymbol{\theta}_{\text{opt}})$.

Taking the Taylor series expansion of (7) about the point $\boldsymbol{\theta}_{\text{opt}}$, we have that

$$\boldsymbol{f}(\boldsymbol{\theta}^{(k)}) \approx \boldsymbol{f}(\boldsymbol{\theta}_{\text{opt}}) + \boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}})(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_{\text{opt}}), \tag{8}$$

and hence

$$\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}_{\text{opt}} \approx \boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}})(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_{\text{opt}}), \tag{9}$$

where $\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) \in \mathbb{R}^{p \times p}$ is the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}_{\text{opt}}$. Thus, (8) shows that in a small neighborhood of $\boldsymbol{\theta}_{\text{opt}}$, the EM algorithm can be approximated by a linear equation.

The linear approximation to the EM fixed-point mapping allows the application of a linear extrapolation scheme to locate the fixed point. Define a residual function $\boldsymbol{g}(\boldsymbol{\theta}) \triangleq \boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{\theta}$. Then, the Taylor series expansion of $\boldsymbol{g}(\boldsymbol{\theta})$ about a point $\boldsymbol{\theta}^{(k)}$ is given by

$$\boldsymbol{g}(\boldsymbol{\theta}) \approx \boldsymbol{g}(\boldsymbol{\theta}^{(k)}) + \boldsymbol{J}(\boldsymbol{\theta}^{(k)})(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}). \tag{10}$$

To find the fixed point, we set (10) to zero, since $\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{0} \Rightarrow \boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{\theta}$. Solving (10) for $\boldsymbol{\theta}$ yields

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(k)} - \boldsymbol{J}^{-1}(\boldsymbol{\theta}^{(k)})\boldsymbol{g}(\boldsymbol{\theta}^{(k)}). \tag{11}$$

Our goal is to find an iterative scheme for locating the fixed point of the EM algorithm. Using (10) and the approximation $\boldsymbol{J}(\boldsymbol{\theta}^{(k)}) \approx (1/\alpha_k)\boldsymbol{I}_p$ [6], we may write two Taylor series expansions for $\boldsymbol{g}(\boldsymbol{\theta})$, one about $\boldsymbol{\theta}^{(k)}$ and the other about $\boldsymbol{f}(\boldsymbol{\theta}^{(k)}) = \boldsymbol{\theta}^{(k+1)}$, i.e.,

$$\begin{aligned} \boldsymbol{g}_0(\boldsymbol{\theta}) &\approx \boldsymbol{g}(\boldsymbol{\theta}^{(k)}) + \frac{1}{\alpha_k}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}), \\ \boldsymbol{g}_1(\boldsymbol{\theta}) &\approx \boldsymbol{g}(\boldsymbol{\theta}^{(k+1)}) + \frac{1}{\alpha_k}(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k+1)}). \end{aligned} \tag{12}$$

Setting the two equations to zero and solving for $\boldsymbol{\theta}$ yields two different linear approximations for the fixed point, i.e.,

$$\begin{aligned} \boldsymbol{\theta}^{(0)} &= \boldsymbol{\theta}^{(k)} - \alpha_k \boldsymbol{g}(\boldsymbol{\theta}^{(k)}), \\ \boldsymbol{\theta}^{(1)} &= \boldsymbol{\theta}^{(k+1)} - \alpha_k \boldsymbol{g}(\boldsymbol{\theta}^{(k+1)}). \end{aligned} \tag{13}$$

By constraining α_k to be negative, we minimize the discrepancy measure $-\|\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(0)}\|^2/\alpha_k$ between the two estimates of the fixed point. Define $\boldsymbol{r}^{(k)} \triangleq \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}$ and $\boldsymbol{v}^{(k)} = \boldsymbol{g}(\boldsymbol{\theta}^{(k+1)}) - \boldsymbol{g}(\boldsymbol{\theta}^{(k)})$. The optimal α_k is then given by

$$\alpha_k = -\|\boldsymbol{r}^{(k)}\|/\|\boldsymbol{v}^{(k)}\|. \tag{14}$$

Thus, the fixed-point approximation in (11) can be expressed in terms of an iterative scheme, i.e.,

$$\begin{aligned} \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \alpha_k \boldsymbol{g}(\boldsymbol{\theta}^{(k)}) \\ &= \boldsymbol{\theta}^{(k)} - \alpha_k(\boldsymbol{f}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}) \\ &= \boldsymbol{\theta}^{(k)} - \alpha_k \boldsymbol{r}^{(k)}. \end{aligned} \tag{15}$$

It is instructive to express (15) in terms of the error between the current estimate $\boldsymbol{\theta}^{(k)}$ and the optimal estimate (after convergence) $\boldsymbol{\theta}_{\text{opt}}$ as these will form the foundation for the squaring method on which SQUAREM is based. To this end, we first note that $\boldsymbol{f}(\boldsymbol{\theta})$ can be expanded in a Taylor series about $\boldsymbol{\theta}_{\text{opt}}$ to give

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{\theta}^{(k)}) &\approx \boldsymbol{\theta}_{\text{opt}} + \boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}})(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_{\text{opt}}) \\ \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{\theta}^{(k)})) &\approx \boldsymbol{\theta}_{\text{opt}} + \boldsymbol{J}^2(\boldsymbol{\theta}_{\text{opt}})(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_{\text{opt}}). \end{aligned} \tag{16}$$

Defining the error $\boldsymbol{\varepsilon}^{(k)} \triangleq \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_{\text{opt}}$ and using (16), we express the vectors $\boldsymbol{r}^{(k)}$ and $\boldsymbol{v}^{(k)}$ as

$$\begin{aligned} \boldsymbol{r}^{(k)} &= \boldsymbol{f}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)} \\ &= \boldsymbol{\theta}_{\text{opt}} + \boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}})\boldsymbol{\varepsilon}^{(k)} - \boldsymbol{\theta}^{(k)} \\ &= (\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p)\boldsymbol{\varepsilon}^{(k)} \end{aligned} \tag{17}$$

and

$$\begin{aligned} \boldsymbol{v}^{(k)} &= \boldsymbol{g}(\boldsymbol{\theta}^{(k+1)}) - \boldsymbol{g}(\boldsymbol{\theta}^{(k)}) \\ &= \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{\theta}^{(k)})) - 2\boldsymbol{f}(\boldsymbol{\theta}^{(k)}) + \boldsymbol{\theta}^{(k)} \\ &= (\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p)^2\boldsymbol{\varepsilon}^{(k)}. \end{aligned} \tag{18}$$

Therefore, by subtracting $\boldsymbol{\theta}_{\text{opt}}$ from (15), the following error relation holds for the EM algorithm

$$\begin{aligned} \boldsymbol{\varepsilon}^{(k+1)} &= \boldsymbol{\varepsilon}^{(k)} - \alpha_k(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}) \\ &= \boldsymbol{\varepsilon}^{(k)} - \alpha_k(\boldsymbol{\theta}_{\text{opt}} - \boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}})\boldsymbol{\varepsilon}^{(k)} - \boldsymbol{\theta}^{(k)}) \\ &= [\boldsymbol{I}_p - \alpha_k(\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p)]\boldsymbol{\varepsilon}^{(k)}. \end{aligned} \tag{19}$$

B. Obtaining SQUAREM via Squaring

In developing SQUAREM, posing the following quadratic problem comes in handy

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b}^{\mathsf{T}}\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^p, \tag{20}$$

where $\boldsymbol{Q} \in \mathbb{R}^{p \times p}$ is a positive-definite and symmetric matrix and $\boldsymbol{b} \in \mathbb{R}^p$. Solving this problem is equivalent to finding $\boldsymbol{x}$ such that $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ since taking the derivative and setting it to zero gives $\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$. Useful insights can be gained from this linear problem since it takes the same form as the EM fixed point problem in (9). Note that since $\boldsymbol{Q}$ is positive-definite, this problem has a unique solution given by $\boldsymbol{x}_{\text{opt}} = \boldsymbol{Q}^{-1}\boldsymbol{b}$.

The classical steepest descent or Cauchy method for (20) is

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \mu_k \boldsymbol{g}^{(k)}, \tag{21}$$

Table I: The adapted SQUAREM algorithm for PMF estimation (SQUAREM-PMF).

SQUAREM-PMF
Input: $\{\boldsymbol{A}_{n,0}\}_{n=1}^{N}$, $\boldsymbol{\lambda}_0$
Output: ML estimates $\{\hat{\boldsymbol{A}}_n\}_{n=1}^{N}$, $\hat{\boldsymbol{\lambda}}$
1. Construct $\boldsymbol{\theta}_0 = [\mathrm{vec}^{\mathsf{T}}(\boldsymbol{A}_{1,0}), \ldots, \mathrm{vec}^{\mathsf{T}}(\boldsymbol{A}_{N,0}), \boldsymbol{\lambda}_0^{\mathsf{T}}]^{\mathsf{T}}$
2. **for** $k = 1, \ldots, K$
3. $\quad \boldsymbol{\theta}_1 = \boldsymbol{f}(\boldsymbol{\theta}_0)$
4. $\quad \boldsymbol{\theta}_2 = \boldsymbol{f}(\boldsymbol{\theta}_1)$
5. $\quad \boldsymbol{r}^{(k)} = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0$
6. $\quad \boldsymbol{v}^{(k)} = (\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1) - \boldsymbol{r}^{(k)}$
7. $\quad \alpha_k = -\|\boldsymbol{r}^{(k)}\|/\|\boldsymbol{v}^{(k)}\|$
8. $\quad$ modify α_k (see Table II)
9. $\quad \boldsymbol{\theta}_0 = \boldsymbol{f}(\boldsymbol{\theta}^{(k)})$
10. $\quad$ **if** $\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}_0\| < \epsilon$
11. $\quad\quad$ Reconstruct $\{\boldsymbol{A}_n^{(k)}\}_{n=1}^{N}$ and $\boldsymbol{\lambda}^{(k)}$ from $\boldsymbol{\theta}^{(k)}$
12. $\quad\quad$ **break**
13. $\quad$ **end if**
14. **end for**

where $\boldsymbol{g}^{(k)} = \nabla f(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b}$ and μ_k is the step size. The convergence rate depends on the matrix $\boldsymbol{Q}$ and can be slow if $\boldsymbol{Q}$ is ill-conditioned. To overcome this problem, the Cauchy-Barzillai-Borwein (CBB) method [7], whose convergence rate is superior to that of the Cauchy method, has been proposed. The CBB method yields the following update equation for (20)

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - 2\mu_k \boldsymbol{g}^{(k)} + \mu_k^2 \boldsymbol{Q}\boldsymbol{g}^{(k)}. \tag{22}$$

A simple relationship between (21) and (22) can be established by deriving error relations for both methods. Letting $\boldsymbol{e}^{(k+1)} \triangleq \boldsymbol{x}^{(k)} - \boldsymbol{x}_{\text{opt}}$, we have for the Cauchy method (21)

$$\begin{aligned} \boldsymbol{e}^{(k+1)} &= \boldsymbol{e}^{(k)} - \mu_k \boldsymbol{g}^{(k)} \\ &= \boldsymbol{e}^{(k)} - \mu_k(\boldsymbol{Q}\boldsymbol{x}^{(k)} - \boldsymbol{b}) \\ &= \boldsymbol{e}^{(k)} - \mu_k \boldsymbol{Q}(\boldsymbol{x}^{(k)} - \boldsymbol{Q}^{-1}\boldsymbol{b}) \\ &= (\boldsymbol{I}_p - \mu_k \boldsymbol{Q})\boldsymbol{e}^{(k)}, \end{aligned} \tag{23}$$

and for the CBB method,

$$\begin{aligned} \boldsymbol{e}^{(k+1)} &= \boldsymbol{e}^{(k)} - 2\mu_k \boldsymbol{g}^{(k)} + \mu_k^2 \boldsymbol{Q}\boldsymbol{g}^{(k)} \\ &= \boldsymbol{e}^{(k)} - 2\mu_k \boldsymbol{Q}\boldsymbol{e}^{(k)} + \mu_k^2 \boldsymbol{Q}^2 \boldsymbol{e}^{(k)} \\ &= (\boldsymbol{I}_p - 2\mu_k \boldsymbol{Q} + \mu_k^2 \boldsymbol{Q}^2)\boldsymbol{e}^{(k)} \\ &= (\boldsymbol{I}_p - \mu_k \boldsymbol{Q})^2 \boldsymbol{e}^{(k)}. \end{aligned} \tag{24}$$

Comparing (23) and (24), it is evident that the CBB method is just a "squared" version of the Cauchy method. By analogy, we can therefore "square" the error relation of the EM fixed point in (19) to obtain the so-called SQUAREM algorithm. Since the EM fixed point can be approximated by a linear equation (cf. (9)), it follows that the squaring approach can be applied to improve the convergence of the EM algorithm. Thus, the error relation for SQUAREM is

$$\boldsymbol{\varepsilon}^{(k+1)} = \big[\boldsymbol{I}_p - \alpha_k\big(\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p\big)\big]^2 \boldsymbol{\varepsilon}^{(k)}. \tag{25}$$

As in the case of EM (cf. (15)), we expand (25) as follows

$$\begin{aligned} \boldsymbol{\varepsilon}^{(k+1)} &= [\boldsymbol{I}_p - 2\alpha_k(\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p) + \alpha_k^2(\boldsymbol{J}(\boldsymbol{\theta}_{\text{opt}}) - \boldsymbol{I}_p)^2]\boldsymbol{\varepsilon}^{(k)} \\ &= \boldsymbol{\varepsilon}^{(k)} - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)}. \end{aligned} \tag{26}$$

Table II: Proposed adaptation procedure for the step size α_k.

Adaptation of α_k
1. **if** $\alpha_k > -1$
2. $\quad\quad \alpha_k = -1$
3. $\quad\quad \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_0 - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)} = \boldsymbol{\theta}_2$
4. $\quad$ **else**
5. $\quad\quad \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_0 - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)}$
6. $\quad\quad$ **if** nonnegativity constraint violated
7. $\quad\quad\quad \alpha_{k,i} = \frac{r_i^{(k)} \pm \sqrt{r_i^{(k)2} - v_i^{(k)}\theta_{0,i}}}{v_i^{(k)}}$, $i = 1, \ldots, p$
8. $\quad\quad\quad$ find the legal segment(s) on the α_k-axis
9. $\quad\quad\quad \bar{\alpha}_k \leftarrow$ segment edge closest to optimal α_k
10. $\quad\quad\quad$ **if** $\bar{\alpha}_k > \alpha_k$
11. $\quad\quad\quad\quad \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_0 - 2\bar{\alpha}_k \boldsymbol{r}^{(k)} + \bar{\alpha}_k^2 \boldsymbol{v}^{(k)}$
12. $\quad\quad\quad$ **else**
13. $\quad\quad\quad\quad \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_0 - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)}$
14. $\quad\quad\quad\quad \boldsymbol{\theta}^{(k)} \leftarrow \mathcal{P}(\boldsymbol{\theta}^{(k)})$
15. $\quad\quad\quad$ **end if**
16. $\quad\quad$ **end if**
17. $\quad\quad$ compute $L(\boldsymbol{\theta}^{(k)})$
18. $\quad\quad$ **while** $L(\boldsymbol{\theta}^{(k)}) > L(\boldsymbol{\theta}^{(k-1)})$
19. $\quad\quad\quad \alpha_k = (\alpha_k - 1)/2$
20. $\quad\quad\quad \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_0 - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)}$
21. $\quad\quad\quad \boldsymbol{\theta}^{(k)} \leftarrow \mathcal{P}(\boldsymbol{\theta}^{(k)})$
22. $\quad\quad\quad$ compute $L(\boldsymbol{\theta}^{(k)})$
23. $\quad\quad$ **end while**
24. $\quad$ **end if**

From (26), it follows that

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - 2\alpha_k \boldsymbol{r}^{(k)} + \alpha_k^2 \boldsymbol{v}^{(k)}. \tag{27}$$

C. The SQUAREM-PMF algorithm

While the SQUAREM algorithm in [6] accelerates the convergence of the EM algorithm, it is applied in a more general context where there are no constraints on the parameters. However, for PMF estimation, the estimated parameters must fulfill the probability simplex constraints, i.e., $\hat{\boldsymbol{A}}_n \geq \boldsymbol{0}, \forall n$, $\hat{\boldsymbol{\lambda}} > \boldsymbol{0}$ (nonnegativity); and $\boldsymbol{1}^{\mathsf{T}}\hat{\boldsymbol{A}}_n = \boldsymbol{1}^{\mathsf{T}}, \forall n$, $\boldsymbol{1}^{\mathsf{T}}\hat{\boldsymbol{\lambda}} = 1$ (sum-to-one). Fortunately, the update equation (27) ensures that the sum-to-one constraint is always fulfilled. On the other hand, the nonnegativity constraint may be violated because some choices of α_k yield parameter vectors with negative elements. Therefore, we propose an adaptation procedure for α_k which ensures that the nonnegativity constraint is fulfilled.

Table I presents the SQUAREM-PMF algorithm, which largely follows the same steps as SQUAREM [6]. We focus on the modification of α_k in Step 8, which is summarized in more detail in Table II. Steps 1-3 and 17-23 ensure that the algorithm converges to a stationary point regardless of the starting point, while preserving the stability of EM [6]. This is achieved by selecting α_k such that the negative log-likelihood (NLL) $L(\boldsymbol{\theta}^{(k)})$ (cf. (4)) of the updated parameter vector is less than the NLL of the previous parameter vector.

To ensure that the nonnegativity constraint is fulfilled, we adopt the following approach, described in Steps 5-16. After updating the parameter vector, we check whether all the elements of $\boldsymbol{\theta}^{(k)}$ are nonnegative. If not, we set each element of $\boldsymbol{\theta}^{(k)}$ to zero and solve the resulting p quadratic equations for α_k, ending up with two solutions, say $(\alpha_{k,i,1}, \alpha_{k,i,2})$, for each

equation. Each such solution pair defines segments on the α_k-axis of the form $(-\infty,\, \alpha_{k,i,1}] \cup [\alpha_{k,i,2},\, \infty)$ or $[\alpha_{k,i,1},\, \alpha_{k,i,2}]$, within which any value enforces nonnegativity on the corresponding element of $\boldsymbol{\theta}^{(k)}$. Therefore, the intersection of all p segments yields one or more legal segments, i.e., with values of α_k that enforce nonnegativity on *all* elements of $\boldsymbol{\theta}^{(k)}$.

Informed by the fact that we should always use the optimal step size (cf. Table I, Step 7) whenever feasible for maximum acceleration [6], we select the segment edge closest in distance to the optimal α_k as the step size, $\bar{\alpha}_k$. This step size is used only if it is larger than the optimal α_k to avoid multiple likelihood computations later; otherwise we use the optimal α_k and project $\boldsymbol{\theta}^{(k)}$ onto the probability simplex using an algorithm such as the one presented in [8].

The nonnegativity constraint may be violated once again after modifying α_k in Step 19. Thus, we apply the projection algorithm [8] once more on $\boldsymbol{\theta}^{(k)}$. Note that the earlier approach described in Steps 5-16 cannot be used at this stage, as the objective is to ensure that α_k is chosen such that the NLL always decreases. Indeed, there is always a tradeoff between the rate of convergence and the stability of the algorithm [6].

It is important to note that in our simulations, we observed that using the projection algorithm without selecting the step size $\bar{\alpha}_k$ resulted in a huge performance degradation in the runtime. This is an indication that our approach is justified and points to a possible relationship between $\bar{\alpha}_k$ and the NLL.

IV. Simulations

In this section, we investigate the performance of the PMF estimation algorithms through carefully designed numerical simulations. In the first part, we use synthetic data to study the convergence performance of SQUAREM-PMF compared to EM, AD-EM, and triplet-based AO-ADMM. In the second part, we investigate the performance of the algorithms using real data in the context of movie recommendation. Here, we compare the performance of SQUAREM-PMF to triplet-based AO-ADMM and to EM.

A. Synthetic-Data Experiments

We consider a similar setup to [3] with $N = 5$ random variables, each taking one of $I_n = 10$ discrete values for $n = 1, \ldots, N$ such that the full PMF tensor $\boldsymbol{\mathcal{X}}$ has 10^5 elements. The true PMF is realized by a rank $R = 5$ CPD with the elements of the factor matrices $\boldsymbol{A}_n \in \mathbb{R}^{10\times 5}$ and of the loading vector $\boldsymbol{\lambda} \in \mathbb{R}^5$ being drawn independently from a uniform distribution, and being normalized to satisfy the probability simplex constraints.

We obtain $T = 10^5$ observed vectors $\boldsymbol{y}[t]$, $t = 1, \ldots, T$ as follows. The latent state $s[t]$ is drawn according to the prior probability given by $\boldsymbol{\lambda}$, then the elements of a complete vector $\boldsymbol{x}[t]$ (i.e., $x_n[t]$) are drawn independently according to $\boldsymbol{A}_n(:, s[t])$, i.e., the $s[t]$-th column of $\boldsymbol{A}_n$. To obtain the observed vector $\boldsymbol{y}[t]$, elements of $\boldsymbol{x}[t]$ are randomly hidden with an outage probability of $p = 0.25$. EM, SQUAREM-PMF, and AO-ADMM are initialized by drawing the elements of the factor matrices and loading vectors randomly from a uniform distribution (and normalizing appropriately to fulfill the probability simplex constraints), while for AD-EM, 20 AD iterations are used to initialize the subsequent EM part.

Table III: Comparison of average runtime, average number of iterations and convergence success rate (out of 1000 trials) for AD-EM, EM and SQUAREM-PMF.

	AD-EM	**EM**	**SQUAREM-PMF**
Average runtime [s]	1046	588	**161**
Average no. of iterations	2415	2779	**505**
Convergence success rate	954	932	**1000**

Fig. 1 compares the convergence of EM, AD-EM, AO-ADMM, and SQUAREM-PMF in terms of the negative log-likelihood (cf. (4)) minus the minimum mean value (per trial) across all four algorithms ($\text{NLL} - \text{NLL}_{\text{min}}$), as well as the MSE of the parameter vector $\boldsymbol{\theta}^{(k)}$, given by $\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}\|^2/M$, where $\boldsymbol{\theta} \in \mathbb{R}^M$ is the ground-truth parameter vector. Since each SQUAREM-PMF iteration consists of three EM updates (Steps 3, 4, and 9), the maximum number of iterations is set to $K = 10^4$ and $K = 3\times 10^4$ for SQUAREM-PMF and EM / AD-EM, respectively. Furthermore, we set $K = 10^3$ and $K = 10^5$ for AO and ADMM, respectively. The results are presented as a function of the number of SQUAREM-PMF iterations, with the EM / AD-EM plots consisting of the value at every third iteration (i.e., 3, 6, 9, ...) for proper scaling. Similarly, we select ADMM values at equal intervals for each AO sweep so that we obtain 10^4 values as required for plotting. The results are averaged over $L = 1000$ independent trials. We set the stopping threshold ϵ to be 10^{-7}.

We observe that SQUAREM-PMF converges faster than EM and achieves a comparable performance. Note that the final NLL is of the order of nearly 10^6 and hence the difference between EM and SQUAREM-PMF in the final $\text{NLL} - \text{NLL}_{\text{min}}$ is negligible. In addition, SQUAREM-PMF decreases the NLL monotonically and thus preserves the stability of EM. We also observe that EM converges to a better MSE compared to AD-EM, demonstrating that the initial (computationally expensive) AD iterations are not necessary. It can be seen that all three EM-based algorithms (which use all the observed data) perform better than the marginal-based AO-ADMM, despite the fact that the latter is allowed to run for many more iterations.

Fig. 2 shows the empirical complementary cumulative distribution function (CCDF) for the number of iterations and the runtime. We omit AO-ADMM since it would require a huge number of iterations (and, hence, much more runtime) to converge to a comparable result to the EM-based algorithms. The acceleration provided by SQUAREM-PMF in terms of the number of iterations and the runtime is clearly visible. This can be appreciated by looking at the average runtime in Table III. We see that, on average, SQUAREM-PMF speeds up convergence by a factor of about 3.6 and requires far fewer iterations. The convergence success rate is another measure

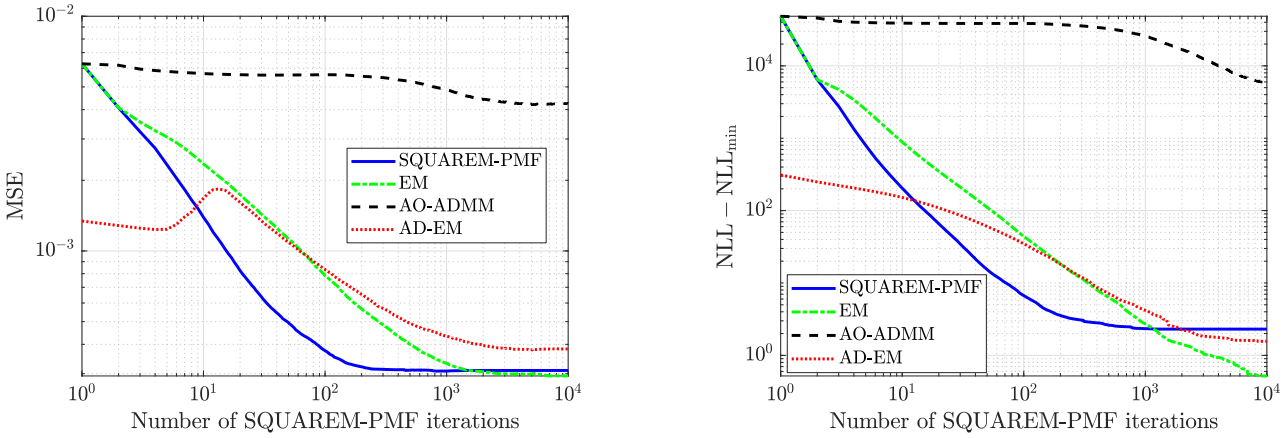

Figure 1: Convergence of the MSE of the parameter vector $\boldsymbol{\theta}^{(k)}$ and the $\mathrm{NLL} - \mathrm{NLL}_{\mathrm{min}}$ for EM, AD-EM, AO-ADMM, and SQUAREM-PMF.

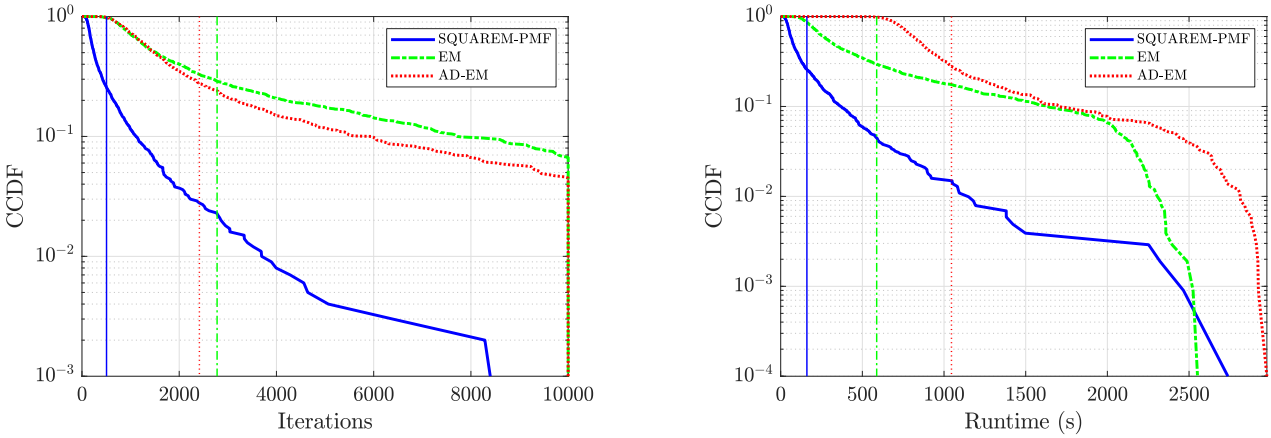

Figure 2: CCDF of the number of iterations and the runtime for SQUAREM-PMF, EM, and AD-EM. The vertical lines represent the mean value for each algorithm.

of interest, i.e., how many trials fulfill the stopping criterion as opposed to running the maximum number of iterations allowed. For all trials, SQUAREM-PMF converges before exhausting the maximum number of iterations, while for some trials, EM does not fulfill the stopping criterion and would need more iterations to converge.

B. Real-Data Experiments

We also evaluate the performance of the EM and SQUAREM-PMF algorithms for the task of recommendation systems by using the MovieLens dataset [9]. This dataset contains movie ratings on a 5-star scale with half-star increments, i.e., $\{0.5, \ldots, 5.0\}$, which are provided by a number of users, and movies are described by their genre. Initially, we select three genres from the entire dataset, i.e., action, animation, and comedy, and keep 10 top-rated movies to form a user-movie matrix. Subsequently, we extract the ratings of users who have rated more than two movies and map the ratings to a discrete scale $\{1, \ldots, 10\}$, i.e., the alphabet size of each movie is $I_n = 10$, $n = 1, \ldots, 10$. In this way, we obtain the initial rating matrix $\boldsymbol{Y} \in \mathbb{R}^{T \times N}$, where T stands for the number of users and N is the number of movies. This initial rating matrix has an outage probability $p = 0.4$, i.e., 40 % of the ratings are missing.

Our goal is to estimate the joint PMF of the ratings of $N = 10$ random variables (movies), which take values from $\{1, \ldots, 10\}$. Therefore, the joint PMF tensor $\boldsymbol{\mathcal{X}}$ is a 10-th order tensor with 10^{10} elements. Note that in the case of real-data experiments, the ground-truth joint PMF and the rank R of its CPD are unknown. However, the latter is a single discrete variable that can be tuned, e.g., by going through the training, validation, and testing procedures as in machine learning.

To this end, we shuffle the users within the initial rating matrix, consisting of 84751 users. Then 70 % of the data samples are used for training, 10 % for validation, and 20 % for testing. The training matrix is used to calculate third-order marginal PMFs for AO-ADMM, from which the factor matrices and loading vector are estimated. The EM and

Table IV: RMSE and MAE of MovieLens rating predictions from different algorithms.

Algorithm	Estimator	RMSE	MAE
AO-ADMM	MMSE	0.8452±0.004	0.6595±0.003
	MAP	0.9715±0.006	0.6558±0.004
EM	MMSE	**0.8192±0.004**	**0.6430±0.003**
	MAP	0.9731±0.006	0.6498±0.005
SQUAREM-PMF	MMSE	**0.8193±0.004**	**0.6429±0.003**
	MAP	0.9735±0.006	0.6496±0.004
Global average		0.9385±0.004	0.7305±0.003
User average		0.9399±0.005	0.7270±0.004
Movie average		0.9399±0.005	0.7270±0.004
Random guess		2.1077±0.010	1.7145±0.009

SQUAREM-PMF algorithms use $\boldsymbol{Y}$ directly to estimate the model. In the validation stage, we seek to estimate R and the AO-ADMM learning rate. To achieve this, for each user in the validation sub-matrix, we hide one rating. The conditional expectation (i.e., the MMSE estimate) of the missing rating is given by

$$\hat{s}_{n,t} = \sum_{i_n=1}^{I_n} i_n \, \mathsf{Pr}(i_n \,|\, s_{1,t}, \ldots, s_{n-1,t}, s_{n+1,t}, \ldots, s_{N,t}), \quad (28)$$

where the ratings of the t-th user are $\boldsymbol{s}[t] = [s_{1,t}, \ldots, s_{N,t}]$, and $s_{n,t}$ is set to zero if the rating is unobserved. This procedure is repeated for ranges of values of R and AO-ADMM learning rates. The model with minimized root mean squared error (RMSE) between the true and predicted rating (averaged over all users) is used further for the retraining, i.e., we combine datasets from training and validation and estimate $\{\boldsymbol{A}_n\}_{n=1}^N$ and $\boldsymbol{\lambda}$ again. The RMSE averaged over $L = 20$ independent trials is given by

$$\mathrm{RMSE} = \sqrt{\frac{1}{L \cdot T_{\mathrm{val}}} \sum_{\ell=1}^{L} \sum_{t=1}^{T_{\mathrm{val}}} |\hat{s}_{n,t,\ell} - s_{n,t,\ell}|^2} \quad (29)$$

where T_{val} denotes the number of users in the validation dataset. Finally, we use the retrained model for testing the performance of the algorithms.

For each user in the testing matrix, we hide one rating and predict it using the estimated model. In addition to the conditional expectation (MMSE estimate) in (28), this is done via the MAP rule which is given by

$$\hat{s}_{n,t} = \underset{i_n \in \{1,\ldots,I_n\}}{\arg\max} \; \mathsf{Pr}(i_n \,|\, s_{1,t}, \ldots, s_{n-1,t}, s_{n+1,t}, \ldots, s_{N,t}). \quad (30)$$

As naïve predictors, we use also the global average of all ratings, the user average, the movie average, and random guessing. To evaluate the quality of the predictors, we use the RMSE from (29) (with T_{test} users) and mean absolute error (MAE), defined as

$$\mathrm{MAE} = \frac{1}{L \cdot T_{\mathrm{test}}} \sum_{\ell=1}^{L} \sum_{t=1}^{T_{\mathrm{test}}} |\hat{s}_{n,t,\ell} - s_{n,t,\ell}|, \quad (31)$$

where T_{test} refers to the number of users in the testing matrix.

In Table IV the performance of all three algorithms in terms of the RMSE and the MAE is compared. We observe that both EM and SQUAREM-PMF outperform AO-ADMM. While the latter works with a subset of available ratings, EM and SQUAREM-PMF use all the available data to predict the missing ratings. Importantly, SQUAREM-PMF performs comparably to EM but converges faster, further demonstrating its efficiency.

V. Conclusion

We have investigated the problem of efficiently estimating a low-rank PMF from partially observed data. In particular, we have addressed the slow convergence of the EM algorithm proposed in [3]. By adapting the SQUAREM acceleration scheme [6] to the context of PMF estimation, we propose the SQUAREM-PMF algorithm to speed up the convergence of the EM algorithm. Using synthetic data, we have shown that SQUAREM-PMF accelerates EM by a factor of about 3.6 and requires much fewer iterations to converge. Moreover, running the EM algorithm alone without prior initialization using the AD-type scheme is sufficient. Through simulations on real data, we have demonstrated that SQUAREM-PMF is applicable to tasks such as movie recommendation where it performs comparably to EM and outperforms AO-ADMM, a state-of-the-art PMF estimation algorithm.

Acknowledgment

The authors gratefully acknowledge the support of the German Research Foundation (DFG) under the PROMETHEUS project (Project no. HA 2239/16-1).

References

[1] N. Kargas, N. D. Sidiropoulos, and X. Fu, "Tensors, Learning, and "Kolmogorov Extension" for Finite-Alphabet Random Vectors," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4854–4868, Sep. 2018.

[2] A. Yeredor and M. Haardt, "Estimation of a Low-Rank Probability-Tensor from Sample Sub-Tensors via Joint Factorization Minimizing the Kullback-Leibler Divergence," in *Proc. 2019 27th European Signal Processing Conference (EUSIPCO)*, Sept. 2019.

[3] A. Yeredor and M. Haardt, "Maximum Likelihood Estimation of a Low-Rank Probability Mass Tensor from Partial Observations," *IEEE Signal Processing Letters*, vol. 26, no. 10, pp. 1551–1555, Oct. 2019.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 2, pp. 1–22, 1977.

[5] S. K. Ng, T. Krishnan, and G. J. McLachlan, "The EM algorithm," in *Handbook of computational statistics*. Springer, 2012, pp. 139–172.

[6] R. Varadhan and C. Roland, "Simple and globally convergent methods for accelerating the convergence of any EM algorithm," *Scandinavian Journal of Statistics*, vol. 35, no. 2, pp. 335–353, 2008.

[7] M. Raydan and B. F. Svaiter, "Relaxed Steepest Descent and Cauchy-Barzilai-Borwein Method," *Computational Optimization and Applications*, vol. 21, pp. 155–167, 2002.

[8] W. Wang and M. Á. Carreira-Perpiñán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," 2013, *arXiv*:1309.1541.

[9] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, pp. 1–19, 2015.