Scalable Probabilistic Tensor Factorization for Binary and Count Data

Piyush Rai*, **Changwei Hu***, **Matthew Harding**†, **Lawrence Carin***
*Department of Electrical & Computer Engineering, Duke University
†Sanford School of Public Policy & Department of Economics, Duke University
Durham, NC 27708, USA
{piyush.rai,ch237,matthew.harding,lcarin}@duke.edu

Abstract

Tensor factorization methods provide a useful way to extract latent factors from complex multirelational data, and also for predicting missing data. Developing tensor factorization methods for massive tensors, especially when the data are binary- or count-valued (which is true of most real-world tensors), however, remains a challenge. We develop a scalable probabilistic tensor factorization framework that enables us to perform efficient factorization of massive binary and count tensor data. The framework is based on (i) the Pólya-Gamma augmentation strategy which makes the model fully locally conjugate and allows closed-form parameter updates when data are binary- or count-valued; and (ii) an efficient *online* Expectation Maximization algorithm, which allows processing data in small mini-batches, and facilitates handling massive tensor data. Moreover, various types of constraints on the factor matrices (e.g., sparsity, non-negativity) can be incorporated under the proposed framework, providing good interpretability, which can be useful for qualitative analyses of the results. We apply the proposed framework on analyzing several binary- and count-valued real-world data sets.

1 Introduction

Tensor factorization methods [Kolda and Bader, 2009] offer a useful way to learn latent factors from complex multiway data. These methods decompose the original tensor data into a set of factor matrices (one for each mode or "way" of the tensor), which can be used as a *latent* feature representation for the objects in each of the tensor mode, and can be used for other tasks, such as tensor completion. Among tensor factorization methods, probabilistic approaches [Chu and Ghahramani, 2009; Xu *et al.*, 2013; Rai *et al.*, 2014] are especially appealing because of a proper generative model of the data, which allows modeling different data types and handling missing data in a natural way.

Real-world tensor data are often binary- or count-valued [Nickel *et al.*, 2011; Chi and Kolda, 2012]. For example, a multirelational social network [Nickel *et al.*, 2011] can be described as a three-way binary tensor with two modes denoting people and the third mode denoting the types of relationships. Likewise, from a database of research publications, one may construct a three-way (AUTHORS $\times$ WORDS $\times$ VENUES) count-valued tensor, where the three dimensions could be authors, words, and publication venues and each entry of the tensor denotes the number of time an author used a specific word at a specific venue. Tensor factorization on this *multiway* data can be used for *topic modeling* on such a publications corpus (the latent factors would correspond to topics). Another application could be in recommender systems; having learned the latent factors of authors and venues, one can use these factors for author-author recommendation (for potential co-authors) or author-venue recommendation (recommending the most appropriate venues for a given author).

Although several tensor factorization methods have been proposed in the recent years [Kolda and Bader, 2009; Chu and Ghahramani, 2009] and there has been a significant recent interest on developing scalable tensor factorization methods [Kang *et al.*, 2012; Inah *et al.*, 2015; Papalexakis *et al.*, 2012; Beutel *et al.*, 2014], most of these methods treat data as real-valued, and are therefore inappropriate for handling binary and count data; also see Related Work (Section 7). Motivated by the prevalence of such binary and count-valued tensors, we present a probabilistic tensor factorization framework which can handle binary and count-valued tensors, while being scalable for massive tensor data. Our starting point will be a conjugate, fully Bayesian model for both binary and count data, for which we develop an efficient Gibbs sampler. The framework is based on the Pólya-Gamma data augmentation strategy [Polson *et al.*, 2012] which allows transforming likelihoods such as logistic (binary data) and negative binomial (count data) into a Gaussian. We then develop Expectation Maximization (EM), and, subsequently, an efficient online EM algorithm [Cappé and Moulines, 2009] which allows scaling up to massive tensor data. Our online EM algorithm performs comparably or better than Gibbs sampling and EM, while being considerably faster to run (and requiring much less storage). Although here we focus on binary and count data, the framework can also handle other types of likelihoods, e.g., multinomial (for categorical data). We present experimental results on various data sets, including a publications corpus, and a massive transactions data set consisting of food-item purchase records from a large set of households.

2 Canonical PARAFAC (CP) Decomposition

Given a tensor $\mathcal{Y}$ of size $n_1 \times n_2 \times \cdots \times n_K$, with n_k denoting the size of $\mathcal{Y}$ along the k^{th} mode (or "way") of the tensor, the goal in a Canonical PARAFAC (CP) decomposition [Kolda and Bader, 2009] is to decompose $\mathcal{Y}$ into a set of K factor matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}$ where $\mathbf{U}^{(k)} = [\boldsymbol{u}_1^{(k)}, \ldots, \boldsymbol{u}_R^{(k)}]$, $k = \{1, \ldots, K\}$, denotes the $n_k \times R$ factor matrix associated with mode k. In its most general form, CP decomposition expresses the tensor $\mathcal{Y}$ via a weighted sum of R rank-1 tensors

$$\mathcal{Y} \sim f(\sum_{r=1}^{R} \lambda_r . \boldsymbol{u}_r^{(1)} \odot \ldots \odot \boldsymbol{u}_r^{(K)}) = f(\Psi) \qquad (1)$$

where f specifies the noise model and depends on the data being modeled (e.g., f can be Gaussian for real-valued, Bernoulli-logistic for binary-valued, Poisson for count-valued tensors); λ_r is the weight associated with the r^{th} rank-1 component; the $n_k \times 1$ column vector $\boldsymbol{u}_r^{(k)}$ represents the r^{th} latent factor of mode k; and $\odot$ denotes vector outer product. We use subscript $\boldsymbol{i} = \{i_1, \ldots, i_K\}$ to denote the K-dimensional index of the $\boldsymbol{i}$-th entry in the tensor $\mathcal{Y}$. The data are given as a set of N observations $\{y_{\boldsymbol{i}}\}_{\boldsymbol{i}=1}^{N}$ from the tensor and the goal is to perform CP decomposition (1) using these observations. Note that (1) can also be written, individually, for each observation $y_{\boldsymbol{i}}$ as

$$y_{\boldsymbol{i}} \sim f(\psi_{\boldsymbol{i}}) \qquad (2)$$

where $\psi_{\boldsymbol{i}} = \sum_{r=1}^{R} \lambda_r \prod_{k=1}^{K} u_{i_k r}^{(k)}$ is a real-valued number.

3 CP Decomposition for Logit Models

In this paper, we develop efficient tensor decomposition methods for data where the likelihood (2) is of the form

$$\frac{\{\exp(\psi_i)\}^{y_i}}{\{1 + \exp(\psi_i)\}^{m_i}} \qquad (3)$$

This general form subsumes several likelihood models [Polson *et al.*, 2012]. For example, for $m_i = 1$, we get the logistic likelihood for binary data; and for $m_i = y_i + \xi$, the negative binomial likelihood for count data (where ξ is the overdispersion parameter). In this exposition, we will focus on these two types of data (binary and count), which are the most prevalent in applications of tensor decomposition methods, but the framework we develop can be straightforwardly extended to handle other data types such as categorical data via multinomial likelihood [Polson *et al.*, 2012].

A key thing to notice about the likelihood models of the form (3) is that, using the Pólya-Gamma augmentation scheme [Polson *et al.*, 2012] allows "Gaussian-ifying" the likelihood given in (3). Specifically, for each observation $y_{\boldsymbol{i}}$, introducing a Pólya-Gamma random variable $\omega_{\boldsymbol{i}}$ [Polson *et al.*, 2012] results in the following

$$\frac{\{\exp(\psi_i)\}^{y_i}}{\{1 + \exp(\psi_i)\}^{m_i}} \propto \exp(\kappa_i \psi_i - \omega_i \psi_i^2/2) \qquad (4)$$

where $\kappa_i = y_i - m_i/2$ and $\omega_i \sim \text{PG}(m_i, \psi_i)$ denotes a draw from a Pólya-Gamma distribution. Since the form of the likelihood on the r.h.s. of (4) is Gaussian, this results in a conjugate model when the model parameters also have Gaussian prior distributions. In particular, in probabilistic CP decomposition models of the form (1), the model parameters $\boldsymbol{u}_r^{(k)}$ and λ_r are usually given Gaussian prior distributions [Xiong *et al.*, 2010; Rai *et al.*, 2014], and therefore the Pólya-Gamma augmentation strategy naturally leads to fully conjugate Bayesian analysis even though the original likelihood 3 is non-conjugate. This property was recently exploited in [Rai *et al.*, 2014]; however, their work has the following limitations: (1) only the case of binary data was considered, and (3) inference in the model was done using batch Gibbs sampling which can be infeasible for massive tensors.

In contrast, in this paper, we present a scalable framework for CP decomposition, based on Pólya-Gamma augmentation that applies to not just binary but also count data (and extendable to other data types such as categorical). For both these cases, we present scalable inference algorithms to handle massive tensor data, for which Gibbs sampling [Rai *et al.*, 2014] methods do not scale. Our inference algorithms are based on *online* Expectation Maximization [Cappé and Moulines, 2009] which allows processing data in mini-batches, and easily scales to massive tensors.

4 Pólya-Gamma CP for Binary & Count Data

We first describe the general CP decomposition model with Pólya-Gamma augmentation for likelihoods of the form

$$p(y_{\boldsymbol{i}} | \psi_{\boldsymbol{i}}, m_{\boldsymbol{i}}) \propto \frac{\{\exp(\psi_i)\}^{y_i}}{\{1 + \exp(\psi_i)\}^{m_i}} \qquad (5)$$

where $\psi_{\boldsymbol{i}} = \sum_{r=1}^{R} \lambda_r \prod_{k=1}^{K} u_{i_k r}^{(k)}$ is a real-valued number. We further assume a Gaussian prior distribution on each of the factor matrices $\mathbf{U}^{(k)} = [\boldsymbol{u}_1^{(k)}, \ldots, \boldsymbol{u}_R^{(k)}]$

$$\boldsymbol{u}_r^{(k)} \sim \mathcal{N}or(0, \mathbf{I}) \qquad (6)$$

and assume that the coefficients λ_r's are drawn from the multiplicative gamma process (MGP) prior [Bhattacharya and Dunson, 2011; Rai *et al.*, 2014], with a truncation-level R over the rank of the decomposition. The MGP prior shrinks the λ_r's for the unnecessary factors to close to zero, thereby inferring the effective tensor rank from the data, and has the form

$$\begin{aligned} &\lambda_r \sim \mathcal{N}(0, \tau_r^{-1}), \quad 1 \leq r \leq R \\ &\tau_r = \prod_{l=1}^{r} \delta_l, \quad \delta_l \sim \text{Gamma}(a_c, 1) \quad a_c > 1 \end{aligned} \qquad (7)$$

Using the fact that (5) can be re-expressed as a Gaussian using the transformation (4), and the fact that the priors on the model parameters $\boldsymbol{u}_r^{(k)}$'s and the λ_r's are also Gaussian, we can derive a fully Bayesian inference procedure using Gibbs sampling. In [Rai *et al.*, 2014], the Gibbs sampler for binary tensors was described, so we skip it here for brevity; therefore, and in Section 4.1, we focus only on the Gibbs sampler for the count data case, which was not considered in [Rai *et al.*, 2014]. We will refer to our model as PGCP.

4.1 Gibbs Sampler for PGCP with Count Data

We first re-express $\psi_{\boldsymbol{i}} = \sum_{r=1}^{R} \lambda_r \prod_{k=1}^{K} u_{i_k r}^{(k)}$ as

$$\psi_i = u_{i_k r}^{(k)} c_{i_k r}^{(k)} + d_{i_k r}^{(k)} \tag{8}$$

where $c_{i_k r}^{(k)}$ and $d_{i_k r}^{(k)}$ are defined as $c_{i_k r}^{(k)} = \lambda_r \prod_{k' \neq k} u_{i_{k'} r}^{(k')}$ and $d_{i_k r}^{(k)} = \sum_{r' \neq r} \lambda_{r'} \prod_{k=1}^{K} u_{i_k r'}^{(k)}$. Using (8), (4), $u_{i_k r} \sim \mathcal{N}or(0,1)$, and using results from Gaussian-Gaussian conjugacy, the Gibbs sampling updates for $u_{nr}^{(k)}$, $1 \leq n \leq n_k$, for the case of count-valued tensor data, are given as $u_{nr}^{(k)} \sim \mathcal{N}or(\alpha_{nr}^{(k)}, {\tau_{nr}^{(k)}}^{-1})$, with the precision and mean defined as

$$\begin{aligned}\tau_{nr}^{(k)} &= 1 + \sum_{\boldsymbol{i}, i_k = n} {c_{i_k r}^{(k)}}^2 \omega_{\boldsymbol{i}}, \quad 1 \leq n \leq n_k \\ \alpha_{nr}^{(k)} &= (\tau_{nr}^{(k)})^{-1} \sum_{\boldsymbol{i}, i_k = n} c_{i_k r}^{(k)} \{0.5(y_{\boldsymbol{i}} - \xi) - \omega_{\boldsymbol{i}} d_{i_k r}^{(k)}\}\end{aligned} \tag{9}$$

In a similar manner, writing $\psi_{\boldsymbol{i}} = (\prod_{k=1}^{K} u_{i_k r}^{(k)})\lambda_r + (\sum_{r' \neq r} \lambda_{r'} \prod_{k=1}^{K} u_{i_k r'}^{(k)}) = a_{\boldsymbol{i}}^r \lambda_r + b_{\boldsymbol{i}}^r$, using (4) and the fact $\lambda_r \sim \mathcal{N}(0, \tau_r^{-1})$, the Gibbs sampling updates for λ_r, $1 \leq r \leq R$ are $\lambda_r \sim \mathcal{N}or(\hat{\mu}_r, \hat{\tau}_r^{-1})$, with the precision and mean defined as

$$\begin{aligned}\hat{\tau}_r &= \tau_r + \sum_{\boldsymbol{i}} {a_{\boldsymbol{i}}^r}^2 \omega_{\boldsymbol{i}} \\ \hat{\mu}_r &= \hat{\tau}_r^{-1} \sum_{\boldsymbol{i}} a_{\boldsymbol{i}}^r \{0.5(y_{\boldsymbol{i}} - \xi) - \omega_{\boldsymbol{i}} b_{\boldsymbol{i}}^r\}\end{aligned} \tag{10}$$

The other MGP parameters (the δ_l's) can be sampled easily from a Gamma posterior, following [Rai *et al.*, 2014]. We skip the details here for brevity. **Sampling the overdispersion parameter ξ:** In the negative binomial model for counts, we also need to estimate the overdispersion parameter ξ. We assume a prior $\xi \sim \text{Gamma}(a_0, 1/h)$ and, following [Zhou *et al.*, 2012b], the Gibbs sampling updates for ξ are

$$\xi \sim \text{Gamma}(a_0 + \sum_{i=1}^{N} L_{\boldsymbol{i}}, 1/(h + \sum_{i=1}^{N} \log(1 + \exp(\psi_{\boldsymbol{i}})))) \tag{11}$$

where $L_{\boldsymbol{i}} \sim \text{Poisson}(\xi \log(1 + \exp(1 + \exp(\psi_{\boldsymbol{i}}))$.

5 Inference via Expectation Maximization

Although the Gibbs sampler described in Section 4.1 is efficient for moderate-sized count-valued tensors, running it on massive tensor data can be infeasible, both due to high storage as well as high computational costs.

In this section, we first describe an Expectation-Maximization (EM) algorithm for parameter estimation in the PGCP model (for both binary- as well as count-valued tensor data), and then describe an online EM algorithm that can process data in mini-batches and scale to massive-sized tensors.

To derive the EM algorithm for the PGCP model, we need to write down the *expected* complete data (log)-likelihood for each of the model parameters. The latent variables here are the Pólya-Gamma variables $\omega_{\boldsymbol{i}}$'s. Luckily, the expectation of a PG random variable $\omega \sim \text{PG}(b, c)$ is available in closed-form [Polson *et al.*, 2012], and is given by

$$\mathbb{E}[\omega] = \frac{b}{2c} \tanh(c/2) \tag{12}$$

In the case of binary data where $\omega_{\boldsymbol{i}} \sim \text{PG}(1, \psi_{\boldsymbol{i}})$, the expectation is given by $\mathbb{E}[\omega_i] = (0.5/\psi_i)\tanh(\psi_{\boldsymbol{i}}/2)$, whereas in the case of count data where $\omega_{\boldsymbol{i}} \sim \text{PG}(y_i + \xi, \psi_{\boldsymbol{i}})$, the expectation is given by $\mathbb{E}[\omega_i] = (0.5(y_i + \xi)/\psi_i)\tanh(\psi_{\boldsymbol{i}}/2)$. In the rest of this section, we will abuse notation and simply use ω_i in place of $\mathbb{E}[\omega_i]$. Also, in what follows, κ_i is defined as $\kappa_i = y_i - m_i/2$, where $m_i = 1$ for binary data and $m_i = y_i + \xi$ for count data. With these ingredients, the EM algorithm for the PGCP model then proceeds as follows: **E-Step:** Compute expectations of $\omega_1, \ldots, \omega_N$. **M-Step:** Maximize the expected complete data log-likelihood w.r.t. each model parameter. For the $u_{nr}^{(k)}$ $(n = 1, \ldots, n_k)$, it follows from (8) that the expected complete data log-likelihood $Q(u_{nr}^{(k)})$ is given by the following quadratic

$$-\frac{1}{2} \sum_{i:i_k = n} \omega_i (u_{nr}^{(k)} c_{i_k r}^{(k)})^2 + \sum_{i:i_k = n} (\kappa_i - d_{i_k r}) c_{i_k r}^{(k)} u_{nr}^{(k)}$$

Further, we can vectorize the above and write the complete data log-likelihood for $\boldsymbol{u}_r^{(k)} = [u_{i_1 r}^{(k)}, \ldots, u_{n_k r}^{(k)}]^\top$ as

$$Q(\boldsymbol{u}_r^{(k)}) = -\frac{1}{2} {\boldsymbol{u}_r^{(k)}}^\top \mathbf{S}^{(r,k)} \boldsymbol{u}_r^{(k)} + {\boldsymbol{u}_r^{(k)}}^\top \mathbf{t}^{(r,k)} \tag{13}$$

where $\mathbf{S}^{(r,k)}$ is a diagonal matrix, and $\mathbf{S}^{(r,k)}$ and $\mathbf{t}^{(r,k)} \in \mathbb{R}^{n_k}$ are defined as $\mathbf{S}_{nn}^{(r,k)} = \sum_{i:i_k = n} c_{i_k r}^2 \omega_i$ and $\mathbf{t}_n^{(r,k)} = \sum_{i:i_k = n} (\kappa_i - d_{i_k r} \omega_i) c_{i_k r}$. Note that since $\mathbf{S}^{(r,k)}$ is diagonal, maximizing (13) w.r.t. $\boldsymbol{u}_r^{(k)}$ is very efficient with no matrix inversions needed. Also note that, it is possible to incorporate a prior distribution on $\boldsymbol{u}_r^{(k)}$ (e.g., Gaussian, sparsity, etc.)

$$Q(\boldsymbol{u}_r^{(k)}) = -\frac{1}{2} {\boldsymbol{u}_r^{(k)}}^\top \mathbf{S}^{(r,k)} \boldsymbol{u}_r^{(k)} + {\boldsymbol{u}_r^{(k)}}^\top \mathbf{t}^{(r,k)} + \log p(\boldsymbol{u}_r^{(k)})$$

Alternatively, one can solve a constraint optimization problem for the objective (13) by placing any desired constraint on $u_r^{(k)}$ (e.g., sparsity, non-negativity, etc.).

Likewise, the expected complete data log-likelihood for the $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_R)$ is

$$Q(\boldsymbol{\lambda}) = -\frac{1}{2} \boldsymbol{\lambda}^\top \mathbf{S}_\lambda \boldsymbol{\lambda} + \boldsymbol{\lambda}^\top \mathbf{t}_\lambda \tag{14}$$

where $\mathbf{S}_\lambda$ is again an $R \times R$ diagonal matrix with diagonal entry given by $\mathbf{S}_{\lambda r} = \sum_i {a_i^r}^2 \omega_i$ and $\mathbf{t}_\lambda \in \mathbb{R}^R$ is defined as $\mathbf{t}_{\lambda r} = \sum_i (\kappa_i - b_i^r \omega_i) a_i^r$.

In the case of count data, we need to estimate the overdispersion parameter ξ of the negative binomial. This can be done using the Newton-Raphson procedure in the M-step.

6 Online Expectation Maximization

The batch EM algorithm presented in Section 5 is more efficient than the batch Gibbs sampling presented in Section 4.1. In many cases, however, the data can be too massive to handle even for batch EM. In this section, we present an online EM algorithm [Cappé and Moulines, 2009; Scott and Sun, 2013] that works by processing data in mini-batches and therefore has the capability to scale to massive tensor data (in terms of the tensor dimensions and/or in terms of the number of tensor observations).

Note that the batch EM algorithm presented in Section 5 operates on the sufficient statistics (e.g., $\mathbf{S}^{(r,k)}, \mathbf{t}^{(r,k)}, \mathbf{S}_\lambda, \mathbf{t}_\lambda$, etc.) of the model parameters. These sufficient statistics are computed using *all* the observations from the tensor. Noting that these statistics are defined as summation over the individual observations from the tensor, it is possible to recursively update these sufficient statistics as more and more data arrives. This eliminates the need of having to store all of the tensor observations in the memory and computing these statistics using the entire data in each round of the EM algorithm. Instead, with the online EM algorithm [Cappé and Moulines, 2009], these statistics can be computed using the current mini-batch of the data, and added to the previous statistics. We leverage this idea to develop an efficient, online EM algorithm for the proposed PGCP model.

In the online EM algorithm, we express the sufficient statistics at time t as a convex combination of the old statistics from time $t-1$ and contribution from the new mini-batch of data. For example, for the parameters $\boldsymbol{u}_r^{(k)}$, the estimates of the sufficient statistics $\mathbf{S}_{nn}^{(r,k,t)}$ and $\mathbf{t}_n^{(r,k,t)}$ after seeing a new minibatch $\mathcal{I}_t$ of data at round t is given by

$$\mathbf{S}_{nn}^{(r,k,t)} = (1-\gamma_t)\mathbf{S}_{nn}^{(r,k,t-1)} + \gamma_t \sum_{i \in \mathcal{I}_t : i_k = n} c_{i_k r}^2 \omega_i$$

$$\mathbf{t}_n^{(r,k,t)} = (1-\gamma_t)\mathbf{t}_n^{(r,k,t-1)} + \gamma_t \sum_{i \in \mathcal{I}_t : i_k = n} (\kappa_i - d_{i_k r}\omega_i) c_{i_k r}$$

The sufficient statistics of the other parameters can also be updated in a similar manner, and used in the M step.

We note that, as shown in [Cappé and Moulines, 2009], updating the sufficient statistics recursively as above, and using these in the M step, is equivalent to a stochastic gradient ascent-style updates of the parameters; e.g., for the parameter $\boldsymbol{u}_r^{(k)}$, it would be equivalent to the following update

$$\boldsymbol{u}_r^{(k,t)} = \boldsymbol{u}_r^{(k,t-1)} + \gamma_t I(\boldsymbol{u}_r^{(k,t-1)})^{-1} \nabla Q(\boldsymbol{u}_r^{(k,t-1)})$$

where $I(.)$ denotes the Fisher Information Matrix. The learning rate $\gamma_t \propto (t+t_0)^{-c}$ for some $c \in (0.5, 1]$ is chosen to ensure that $\sum_{t=1}^{\infty} \gamma_t = \infty$ and $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$; these conditions are necessary for the convergence of online EM [Cappé and Moulines, 2009]. Following [Cappé and Moulines, 2009], we set c to be close to 0.5, which worked well in practice in our experiments. Moreover, as recommended in [Cappé and Moulines, 2009; Scott and Sun, 2013], we also smooth the parameter estimates using Polyak-Ruppert averaging where the parameters are averaged over the final $T - T_0$ where T_0 is some large enough value such that the algorithm has settled down.

Finally, note that, although motivated differently, our online EM mirrors stochastic variational inference [Hoffman *et al.*, 2013] methods, where the parameters of the variational approximation of the posterior distribution are updated recursively in a similar fashion with each incoming minibatch of data.

7 Related Work

Tensor factorization methods have become increasingly prevalent nowadays, thanks to the rise of multirelational and multiway data sets in a wide variety of application domains. While much of the work on tensor factorization methods assumes the tensor entries to be real-valued, some generalizations to data from exponential family distributions [Hayashi *et al.*, 2010], generalized linear models [Yılmaz *et al.*, 2011], or specifically for binary-valued tensors [Nickel *et al.*, 2011; Rai *et al.*, 2014] have been also proposed. However, these models can be computationally infeasible to run on massive tensors (e.g., the method in [Hayashi *et al.*, 2010] requires tensor unfolding operations which can easily render it impractical for large tensors) as they all require batch processing. For count-valued tensor data, [Chi and Kolda, 2012] proposed a Poisson tensor factorization method, but this method cannot handle missing values in the tensor, and can therefore not be used for tensor completion. Moreover, these methods also have other limitations such as the need to pre-specify the rank of decomposition *a priori*, or to select it via cross-validation.

With the advent of massive-scale tensor data in applications such as knowledge-bases and social networks, there has been a considerable recent interest on scaling up tensor factorization methods by developing distributed and parallel methods [Kang *et al.*, 2012; Inah *et al.*, 2015; Papalexakis *et al.*, 2012; Beutel *et al.*, 2014]. Most of these methods, however, have one or more of the following limitations: (1) they assume a parallel or distributed setting, (2) missing data cannot be easily handled/predicted, (3) they lack a proper generative model of the data (most of these assume real-valued data), and are not designed for a Bayesian setting, and (4) the rank of the decomposition needs to be chosen via cross-validation. In contrast, our proposed framework does not have any of these limitations.

In [Hayashi *et al.*, 2012], the authors proposed an EM algorithm for exponential family tensor factorization, using Laplace and Gaussian process approximations, and extended it to online EM; however, the proposed algorithm, based on tensor unfolding, has computational complexity that is *cubic* in the sum of the tensor dimensions, and is prohibitive even for moderate-size tensors. In contrast, our algorithms have per-iteration complexity that is linear in the tensor dimension as well as linear in the number of observations (or in the minibatch size in the case of online EM) in the tensor, and therefore can scale up nicely even for massive tensor data.

Finally, scalable Bayesian tensor decomposition is a relatively under-explored area. Some recent attempts include [Zhe *et al.*, 2015; Rai *et al.*, 2014; Zhe *et al.*, 2013]. However, these are limited to real-valued and/or binary-valued tensors (except for the work in [Zhou *et al.*, 2014], which assumes categorical data), and scale poorly w.r.t the size of the tensor and/or rely on batch inference. In contrast, our online EM based framework applies to binary and count data (real-valued data can be trivially handled without the need for Pólya-Gamma augmentation), and extends easily to other data types, such as categorical data (via multinomial likelihood), or data with binomial likelihoods.

8 Experiments

We experiment with several real-world data sets that consist of both binary and count tensors. These include:

- **Kinship:** This is a $104 \times 104 \times 26$ binary tensor [Nickel *et al.*, 2011] consisting of 26 types of relations among a set of 104 individuals.
- **Digg:** This is a $581 \times 124 \times 48$ binary tensor [Zhe *et al.*, 2013] with 0.024% nonzeros.
- **DBLP:** This is a $10,000 \times 200 \times 10,000$ binary tensor [Zhe *et al.*, 2015] constructed from the DBLP database, and contains author-conference-keyword relations (a tensor entry is 1 if the relation holds). This data has 0.001% nonzeros.
- **GDELT:** This is a $220 \times 220 \times 20 \times 52$ count tensor [Leetaru and Schrodt, 2013] consisting of country-country interactions based on 20 types of actions; we use a subset of this data collected over a period of one year (year 2012).
- **Scholars:** This is a $2370 \times 8663 \times 4066$ count tensor constructed from a publications database of Duke University [1], where each entry of the tensor denotes the number of times an authors uses a word at a publication venue.
- **Transactions:** This is a $117,054 \times 438 \times 67,095$ count tensor constructed from a database of transactions data of food-item purchases at various stores in the US [2]; the three tensor modes correspond to households, stores, and items purchased.

In the experiments, we refer to our model PGCP (Pólya-Gamma CP decomposition) with Gibbs, EM and online EM inference, as PGCP-Gibbs, PGCP-EM, and PGCP-Online EM, respectively. For binary data, we compare with MGP-CP [Rai *et al.*, 2014], and (2) Infinite Tucker Factorization [Xu *et al.*, 2013]. For count data, we compare with: (1) PTF [Chi and Kolda, 2012] [3] which assumes a Poisson factorization model, and (2) LRA-NTD[Zhou *et al.*, 2012a] which is a non-negative tensor factorization method but data is assumed to be real-valued.

Experimental settings: All the methods use random initialization. For the online EM, we set the mini-batch size to one-tenth of the total data size. For our methods, the upper bound R on the tensor rank was set to 10 for DBLP data and Kinship data; Digg data, R was set to 5 (larger values yield similar results, albeit at a greater computational cost). For GDELT, Scholar, and Transactions data sets, R was set to 100. Unless specified otherwise, the Gibbs sampler was run for 1000 iterations with 500 burn-in iterations. Unless specified otherwise, the online EM and EM algorithms were run for 1000 iterations. All experiments are done on a standard desktop with Intel i7 3.4GHz processor and 24GB RAM.

[1]https://scholars.duke.edu/

[2]The data is provided to us by USDA.

[3]Since the method in [Chi and Kolda, 2012] cannot handle missing data, we implement a fully Bayesian version of the model and use it in our experiments

8.1 Tensor Completion

We first experiment on the task of tensor completion where we compare all the methods for their ability to predict missing entries in the tensor. For binary tensors, we split the data into 80% training and 20% test, while ensuring that the nonzeros are also split the same way between the training and test data. For count data, we use 95% data as the training data and use the remaining 5% as the heldout data (in Section 8.4, we also show an additional experiment with varying fractions of the training data). The tensor completion results are shown in Table 1 for binary tensor data, and Table 2 for count tensor data. Each result is averaged over five random splits of the training and test data. On both binary as well as count data, the PGCP (all the three variants) outperform the other baselines (except Digg where Inf-Tucker baseline does marginally better). Moreover, the EM and online EM variants of PGCP perform at par or better than Gibbs sampling, suggesting that on massive data sets, the Gibbs sampler, in addition to running slower, may also be mixing poorly (in Section 8.2, we compare the three variants of PGCP in terms of running times).

8.2 Scalability

In this section, we assess the performance of the online EM as compared to the batch counterparts (batch EM and batch Gibbs) in terms of its convergence speed vs the wall-clock run time. Each method was run for a specified amount of time (long enough that it roughly converges to a stable AUC or log-likelihood score). The results from various data sets are shown in Figures (1) and (2). As these figures show, the online EM algorithm reaches reasonably good quality solutions fairly quickly as compared to batch EM and batch Gibbs. In most cases, it even attains better quality solutions than its batch counterparts. In particular, this can be attributed to the fact that the Gibbs sampler may exhibit slow mixing on large data sets. This experiment demonstrates that the PGCP model with the online EM algorithm can be a viable alternative to Gibbs sampling and batch EM on massive tensor data sets.

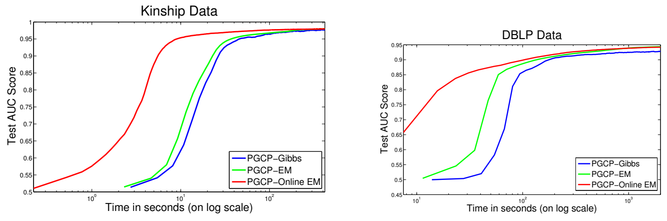

Figure 1: (Left) Scalability on Kinship. (Right) Scalability on DBLP. Blue: PGCP-Gibbs, Green: PGCP-EM, Red: PGCP-Online EM

8.3 Topic Modeling Results

Our model can also be used for topic modeling of multiway data. On the Scholar data (AUTHORS $\times$ WORDS $\times$ VENUES tensor), we do an experiment with PGCP-Online EM by imposing non-negativity constraint on the factor matrices and examine the factor matrix of the words dimension of the tensor. Table 3 shows three of the factors and the top 8 words

	Kinship		Digg		DBLP	
	AUC	**Log-lik.**	**AUC**	**Log-lik.**	**AUC**	**Log-lik.**
MGP-CP [Rai *et al.*, 2014]	0.9796	-0.0601	0.6104	-0.0037	0.9299	-0.2310
Inf-Tucker [Xu *et al.*, 2013]	0.9710	-0.1087	**0.6632**	**-0.0027**	-	-
PGCP-EM	**0.9823**	**-0.0551**	0.6412	-0.0033	**0.9450**	-0.2324
PGCP-Online EM	0.9796	-0.0592	0.6572	-0.0030	0.9423	**-0.2006**

Table 1: Tensor completion on binary data (we report AUC - Area under the ROC curve, and heldout likelihood). Note: For the binary data, since MGP-CP and PGCP-Gibbs are essentially the same, we do not separately report PGCP-Gibbs results here. Also, the Inf-Tucker baseline gave out-of-memory error on the DBLP data, so we are not able to report its results on the DBLP data.

	GDELT		Scholar		Transactions	
	MAE	**Log-lik.**	**MAE**	**Log-lik.**	**MAE**	**Log-lik.**
PTF [Chi and Kolda, 2012]	55.48	-4425695	1.64	-860808	1.47	-2425433
LRA-NTD [Zhou *et al.*, 2012a]	65.05	N/A	-	-	-	-
PGCP-Gibbs	43.22	-2632322	0.792	-146586	0.7256	-2243346
PGCP-EM	44.42	-2694564	**0.732**	**-145198**	**0.6951**	**-2066424**
PGCP-Online EM	**43.16**	**-2614195**	0.734	-145219	0.7092	-2071723

Table 2: Tensor completion on count data (we report MAE - mean-absolute-error, and heldout likelihood). Note: The LRA-NTD baseline gave out-of-memory error on the Scholar and Transactions data, so we are not able to report its results on those data sets.

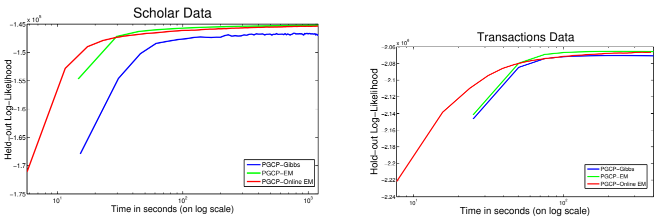

Figure 2: (Left) Scalability on Scholar data. (Right) Scalability on Transactions data. Blue: PGCP-Gibbs, Green: PGCP-EM, Red: PGCP-Online EM

in each of these factors (based on their factor scores). Looking at the top words in these factors suggests that these factors correspond to topics machine learning & statistics (ML), immunology, and cardiac imaging, which are representative subject areas of the Scholar publications corpus.

ML	IMMUNOLOGY	CARDIAC IMAGING
MODEL	CELLS	PHANTOM
STATISTICAL	ANTIBODIES	MOTION
REGRESSION	PLASMA	PATIENT
CATEGORICAL	VACCINE	IMAGING
SPARSE	VIRUS	CARDIAC
BAYESIAN	HUMAN	SLICES
LATENT	RESPONSES	PERFUSION
DICTIONARY	INFECTION	ARTIFACTS

Table 3: Most probable words in topics related to ML, Immunology, and Cardiac Imaging

8.4 Varying Amounts of Training Data

We also investigate the behavior of PGCP with varying amounts of training data. For this experiment, we first split the Scholar data into 80% training and 20% test data, and then use a varying fraction of the training data (here we only report the results with PGCP-online EM). In Figure 3, we show results for (1) heldout log-likelihood vs fraction of training data, and (2) running-time vs fraction of training data.

As Figure 3 (left) shows, the heldout log-likelihood expectedly gets better with increasing training data; however, even with small fractions of the training data, the model achieves reasonably high log-likelihoods, suggesting its robustness.

Figure 3 (right) shows the total running time (summed over the total number of iterations) for each fraction of the training data. As shown, the total running time scales roughly linearly in the number of observations in the training data.

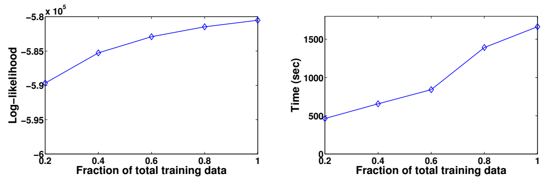

Figure 3: Varying the amount of observed data: (Left) tensor completion accuracy vs fraction of training data, (Right) scalability vs fraction of training data

9 Conclusion and Future Work

We have presented a scalable framework for tensor factorization that can handle binary and count data via a conjugate model obtained using a data augmentation strategy. Leveraging this conjugacy, our framework can perform a fully Bayesian analysis via closed-form Gibbs sampling, as well as allows efficient inference via EM and online EM algorithms. Our work opens doors to the a scalable *and* fully Bayesian analysis of binary- and count-valued tensors, which are now becoming increasingly prevalent in several applications. The work here can be extended in several interesting directions. Although, here, we only considered binary and count data, our framework, which is based on the Pólya-Gamma augmentation, can also be straightforwardly extended to handle categorical data, or data with binomial likelihoods.

Another interesting extension can be incorporating sources of auxiliary information for the tensor for improved tensor decomposition. For example, often data sources with side-information about each of the tensor modes are available [Narita *et al.*, 2012; Yılmaz *et al.*, 2011; Ermiş *et al.*, 2013; Beutel *et al.*, 2014; Rai *et al.*, 2015; Khan and Kaski, 2014]. By sharing the factor matrices of such modes across these multiple data sources, such side-information can be effectively leveraged. To further scale up our framework, many steps of the Gibbs sampler in Section 4.1, as well as of the proposed EM and online EM algorithms, can be parallelized/distributed.

Acknowledgments

The research reported here was supported in part by ARO, DARPA, DOE, NGA and ONR.

References

[Beutel *et al.*, 2014] A. Beutel, A. Kumar, E. Papalexakis, P. P. Talukdar, C. Faloutsos, and E. P. Xing. Flexifact: Scalable flexible factorization of coupled tensors on hadoop. In *SDM*, 2014.

[Bhattacharya and Dunson, 2011] A. Bhattacharya and D. Dunson. Sparse bayesian infinite factor models. *Biometrika*, 2011.

[Cappé and Moulines, 2009] O. Cappé and E. Moulines. Online expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[Chi and Kolda, 2012] E. C. Chi and T. G. Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[Chu and Ghahramani, 2009] W. Chu and Z. Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. In *AISTATS*, 2009.

[Ermiş *et al.*, 2013] B. Ermiş, E. Acar, and A. T. Cemgil. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining and Knowledge Discovery*, 2013.

[Hayashi *et al.*, 2010] K. Hayashi, T. Takenouchi, T. Shibata, Y. Kamiya, D. Kato, K. Kunieda, K. Yamada, and K. Ikeda. Exponential family tensor factorization for missing-values prediction and anomaly detection. In *ICDM*, 2010.

[Hayashi *et al.*, 2012] K. Hayashi, T. Takenouchi, T. Shibata, Y. Kamiya, D. Kato, K. Kunieda, K. Yamada, and K. Ikeda. Exponential family tensor factorization: an online extension and applications. *Knowledge and information systems*, 33(1):57–88, 2012.

[Hoffman *et al.*, 2013] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[Inah *et al.*, 2015] J. Inah, E. Papalexakis, U. Kang, and C. Faloutsos. Haten2: Billion-scale tensor decompositions. In *ICDE*. IEEE, 2015.

[Kang *et al.*, 2012] U Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*, 2012.

[Khan and Kaski, 2014] Suleiman A Khan and Samuel Kaski. Bayesian multi-view tensor factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer, 2014.

[Kolda and Bader, 2009] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[Leetaru and Schrodt, 2013] K. Leetaru and P. A. Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA Annual Convention*, volume 2, page 4, 2013.

[Narita *et al.*, 2012] A. Narita, K. Hayashi, R. Tomioka, and H. Kashima. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25, 2012.

[Nickel *et al.*, 2011] M. Nickel, V. Tresp, and H. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[Papalexakis *et al.*, 2012] E. Papalexakis, C. Faloutsos, and N. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.

[Polson *et al.*, 2012] N. Polson, J. Scott, and J. Windle. Bayesian inference for logistic models using Polya-Gamma latent variables, http://arxiv.org/abs/1205.0310, 2012.

[Rai *et al.*, 2014] P. Rai, Y. Wang, S. Guo, G. Chen, D. Dunson, and L Carin. Scalable bayesian low-rank decomposition of incomplete multiway tensors. In *ICML*, 2014.

[Rai *et al.*, 2015] P. Rai, Y. Wang, and L. Carin. Leveraging features and networks for probabilistic tensor decomposition. In *AAAI*, 2015.

[Scott and Sun, 2013] J. G. Scott and L. Sun. Expectation-maximization for logistic regression. *arXiv preprint arXiv:1306.0040*, 2013.

[Xiong *et al.*, 2010] L. Xiong, X. Chen, T. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, 2010.

[Xu *et al.*, 2013] Z. Xu, F. Yan, and Y. Qi. Bayesian nonparametric models for multiway data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[Yılmaz *et al.*, 2011] K. Y. Yılmaz, A. T. Cemgil, and U. Simsekli. Generalised coupled tensor factorisation. In *NIPS*, 2011.

[Zhe *et al.*, 2013] S. Zhe, Y. Qi, Y. Park, I. Molloy, and S. Chari. Dintucker: Scaling up gaussian process models on multidimensional arrays with billions of elements. *arXiv preprint arXiv:1311.2663*, 2013.

[Zhe *et al.*, 2015] S. Zhe, Z. Xu, X. Chu, Y. Qi, and Y. Park. Scalable nonparametric multiway data analysis. In *AISTATS*, 2015.

[Zhou *et al.*, 2012a] G. Zhou, A. Cichocki, and Xie S. Fast nonnegative matrix/tensor factorization based on low-rank approximation. *IEEE Transactions on Signal Processing*, 60(6):2928–2940, 2012.

[Zhou *et al.*, 2012b] M. Zhou, L. Li, D. Dunson, and L.e Carin. Lognormal and gamma mixed negative binomial regression. In *ICML*, 2012.

[Zhou *et al.*, 2014] J. Zhou, A. Bhattacharya, A. H. Herring, and D. B. Dunson. Bayesian factorizations of big sparse tensors. *Journal of the American Statistical Association*, (just-accepted):00–00, 2014.