Nonparametric Decomposition of Sparse Tensors

Conor Tillinghast [1] **Shandian Zhe** [1]

Abstract

Tensor decomposition is a powerful framework for multiway data analysis. Despite the success of existing approaches, they ignore the sparse nature of the tensor data in many real-world applications, explicitly or implicitly assuming dense tensors. To address this model misspecification and to exploit the sparse tensor structures, we propose Nonparametric dEcomposition of Sparse Tensors (NEST), which can capture both the sparse structure properties and complex relationships between the tensor nodes to enhance the embedding estimation. Specifically, we first use completely random measures to construct tensor-valued random processes. We prove that the entry growth is much slower than that of the corresponding tensor size, which implies sparsity. Given finite observations (*i.e.,* projections), we then propose two nonparametric decomposition models that couple Dirichlet processes and Gaussian processes to jointly sample the sparse entry indices and the entry values (the latter as a nonlinear mapping of the embeddings), so as to encode both the structure properties and nonlinear relationships of the tensor nodes into the embeddings. Finally, we use the stick-breaking construction and random Fourier features to develop a scalable, stochastic variational learning algorithm. We show the advantage of our approach in sparse tensor generation, and entry index and value prediction in several real-world applications.

1. Introduction

Multiway data, such as interactions between multiple nodes (or entities), are ubiquitous in real-world applications. These data are naturally represented by tensors. For example, we can use a three-mode (*customer*, *service-item*, *provider*) tensor to summarize consumer behaviors. Accordingly, tensor decomposition is a fundamental framework for multiway data analysis. In general, tensor decomposition aims to estimate an embedding representation of the nodes in each mode, with which to recover the observed entry values. These embeddings can reflect hidden structures within the tensor nodes, *e.g.,* communities and outliers, and be used in various downstream tasks, such as click-through-rate prediction and commodity recommendation.

Although many excellent tensor decomposition methods have been developed (Tucker, 1966; Harshman, 1970; Chu and Ghahramani, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014), they explicitly or implicitly assume that the tensors have dense entries and overlook the sparse nature of the data in numerous real-world applications. For example, many algorithms (Kolda and Bader, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014; Xu et al., 2012) rely on tensor algebras and require that all the possible entries must have been generated — although most entry values can be zero — so that they can operate on the entire tensor (folding, unfolding, multiplying with matrices, *etc.*). Although quite a few methods can focus on a small set of entries and only decompose the observed entry values (Rai et al., 2014; Zhe et al., 2016b; Du et al., 2018), they essentially assume that the tensor entries are generated by a random function of the embeddings, which is equivalent to assuming the tensor is exchangeable. That is, the distribution of the tensor is invariant to the permutation of the nodes in each mode. According to the Aldous-Hoover theorem (Aldous, 1981; Hoover, 1979), the tensor is either trivially empty or dense, *i.e.,* the number of present entries grows linearly with the tensor size ($\Theta(\prod_k M_k)$ where M_k is the number of nodes in mode k).

However, real-world tensor data are usually very sparse. The number of present entries is way less than the tensor size ($o(\prod_k M_k)$). Therefore, existing models are often misspecified, and unable to capture the valuable structure properties within the observed sparse entries. To address these limitations, we propose NEST, Bayesian nonparametric decomposition models for sparse tensors. Our method not only is flexible enough to capture the complex relationships between the tensor nodes in generating the entry values, but also can account for the generation of sparse entry indices, assimilating both the hidden relationships and sparse structure properties into the embedding representations. Specifically,

[1]School of Computing, University of Utah. Correspondence to: Shandian Zhe <zhe@cs.utah.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

to address the model misspecification, we first use Gamma processes (ΓP) — a popular completely random measure —- to construct sparse tensor-valued processes. We show that the number of present entries is asymptotically much smaller than the corresponding tensor size, *i.e.,* $o(\prod_k M_k)$, which guarantees sparsity. Next, given the finite observations, we propose two nonparametric decomposition models that couple Dirichlet processes (DP) (*i.e.,* normalized ΓP) and Gaussian processes (GP). The first model uses DPs to sample a random measure for each mode. The weights of the measure are viewed as the sociability of the nodes and are used to sample the indices of the observed entries. The locations of the measure are considered as intrinsic properties of the nodes. The weights and locations are concatenated to form the embeddings to sample the values of observed entries with a GP. The second model samples multiple random measures in each mode, where the weights are considered as multiple sociabilities of each node under different (overlapping) groups/communities. These sociabilities are concatenated as the embeddings to sample both the entry indices and values (with a GP for the latter). In this way, both the sparse structure properties and nonlinear relationships are absorbed into the embedding representations. Finally, we use the stick-breaking construction and random Fourier features for GP approximation to derive a tractable variational evidence lower bound, based on which we develop a stochastic variational learning algorithm to enable scalable and efficient model estimation.

For evaluation, we first tested tensor sampling. NEST indeed generates increasingly sparse entries along with the growth of tensor size, while the existing (exchangeable) models generates dense data. We also showcase the generated sparse tensors by NEST. We then evaluated NEST in three real-world applications. In predicting both entry values and entry indices (*i.e.,* link prediction), NEST outperforms the state-of-the-art multilinear and nonparametric decomposition models, often significantly.

2. Background

Tensor Decomposition. A K-mode tensor is denoted by $\mathcal{Y} \in \mathbb{R}^{M_1 \times \ldots \times M_K}$, where each mode k includes M_k nodes (or entities), *e.g.,* customers or products. We index each entry by $\mathbf{i} = (i_1, \ldots, i_K)$, and denote the value of that entry by $y_{\mathbf{i}}$. Given a set of observed entries and their values, $\mathcal{D} = \{(\mathbf{i}_n, y_{\mathbf{i}_n})\}_{n=1}^N$, we aim to learn a set of embeddings to represent the nodes in each mode, $\mathcal{U} = \{\mathbf{U}^1, \ldots, \mathbf{U}^K\}$ where each row j in $\mathbf{U}^k$ is the embedding vector of the j-th node in mode k, denoted by $\mathbf{u}_j^k$. In practice, tensor data are usually very sparse, *i.e.,* the number of present entries is much smaller than the tensor size, *i.e.,* $N \ll \prod_{k=1}^K M_k$. Tensor data can be represented by hypergraphs, where a present entry indicates a hyper-edge connecting K nodes, and the entry value is the edge weight (see Fig. 1 in the supplementary material). Note that a zero-valued entry is totally different from a nonexistent entry. The former means the hyper-edge exists but the edge weight is 0, while the latter means the hyper-edge does not exist at all.

Dense Tensors and Exchangeable Arrays. Classical tensor decomposition methods demand every entry should be observed and so the tensor can be operated as a mathematical object. For example, Tucker decomposition (Tucker, 1966) models $\mathcal{Y} = \mathcal{W} \times_1 \mathbf{U}^1 \times_2 \ldots \times_K \mathbf{U}^K$, where $\mathcal{W} \in \mathbb{R}^{r_1 \times \ldots \times r_K}$ is a parametric core tenor, and $\times_k$ is the mode-k tensor-matrix product (Kolda, 2006), which generalizes the matrix-matrix product. If we restrict $\mathcal{W}$ to be diagonal, Tucker decomposition is reduced to CANDECOMP/PARAFAC (CP) decomposition (Harshman, 1970).

However, assuming a fully observed tensor is usually unrealistic in practice. Many methods hence conduct an element-wise decomposition over a small set of observed entry values (Zhe et al., 2016a; Zhe and Du, 2018; Rai et al., 2015; Hu et al., 2015; Schein et al., 2015). These methods, from a Bayesian point of view, can be considered as generating tensor entries in the following way. First, we sample the embeddings from a prior distribution $p(\mathcal{U})$, *e.g.,* Gaussian (Zhe et al., 2016b; Du et al., 2018). Given the embeddings $\mathcal{U}$, we sample the presence of each entry independently,

$$z_{\mathbf{i}} \sim \text{Bern}(f(\mathbf{x}_{\mathbf{i}})), \tag{1}$$

where $\mathbf{x}_i = [\mathbf{u}_{i_1}^1; \ldots; \mathbf{u}_{i_K}^K]$, $\text{Bern}(\cdot)$ is the Bernoulli distribution, and $f(\cdot)$ is a function of the associated embeddings, *e.g.,* the element-wise Tucker or CP form (Du et al., 2018; Hu et al., 2015). To capture the (possible) nonlinear relationship between the tensor nodes, Xu et al. (2012); Zhe et al. (2016a) placed a Gaussian process (GP) prior over $f(\cdot)$. To sample the entry value, we can use other distributions, *e.g.,* $y_{\mathbf{i}} \sim \mathcal{N}(f(\mathbf{x}_{\mathbf{i}}), \sigma^2)$ for continuous values and Poisson distribution for count values (Schein et al., 2015).

While some existing models can avoid assuming the observation of the full tensor by performing element-wise decomposition, they intrinsically sample exchangeable arrays that (asymptotically) lead to dense tensors almost surely (*i.e.,* with probability one). Specifically, the tensor entries (and values) are generated by a random function of the embeddings (see (1)). These embeddings, no matter from what prior they are sampled from, can be considered as a transform of uniform random variables. Therefore, if we use these models to sample a sequence of arrays with growing numbers of nodes in each mode, according to the Aldous-Hoover theorem (Aldous, 1981; Hoover, 1979), in the limit of infinite (countable) nodes, the joint distribution of the tensor is exchangeable. That is, the distribution is invariant to arbitrary permutations of the nodes in each mode. The asymptotic portion of

the present entries is $\xi = \int f(\mathbf{x}_i)p(\mathbf{x}_\mathbf{i})\mathrm{d}\mathbf{x}_\mathbf{i}$ where $p(\mathbf{x}_\mathbf{i})$ is the prior (see (1)). As a consequence, the tensor is either trivially empty (for $\xi = 0$) or dense almost surely, since $\Theta(\xi \prod_{k=1}^K M_k) = \Theta(\prod_{k=1}^K M_k)$.

3. Model

Despite the success of existing methods, their intrinsic nature of sampling dense tensors can be a severe model misspecification for sparse data observed in numerous practical applications. With this assumption, they miss capturing the valuable structure properties within the sparsely present entries that can potentially also benefit embedding estimations. To overcome these limitations, we propose NEST, a nonparametric decomposition approach of sparse tensors, presented as follows.

3.1. Sparse Tensor Processes

First, we construct tensor-valued random process, which guarantees almost surely sparsity at the limit (*i.e.,* when the number of sampled entries grows to infinity). Specifically, for each mode k, we use a Gamma process (ΓP) (Hougaard, 1986; Brix, 1999) — a commonly used completely random measure (CRM) (Kingman, 1967) — to sample a discrete measure, where the locations are drawn from a finite interval $[0, \alpha]$ to represent the tensor nodes. We then construct a product intensity measure for a Poisson point process (PPP), with which we sample the indices of the present entries. The model is given by

$$\begin{aligned} W_k^\alpha &\sim \Gamma\mathrm{P}(\lambda_\alpha)(1 \le k \le K), \\ T &\sim \mathrm{PPP}(W_1^\alpha \times \cdots \times W_K^\alpha), \end{aligned} \tag{2}$$

where λ_α is the Lebesgue measure with the support restricted to $[0, \alpha](\alpha > 0)$, W_k^α and T are discrete measures,

$$W_k^\alpha = \sum_{j=1}^{\infty} w_{kj}^\alpha \cdot \delta_{\theta_j^k}, \qquad T = \sum_{\mathbf{i} \in \mathcal{S}} c_\mathbf{i} \cdot \delta_{\boldsymbol{\theta}_\mathbf{i}},$$

where $\delta_{[\cdot]}$ is the Dirac Delta measure, $\{w_{kj}^\alpha\}$ and $\{\theta_j^k\}$ are the weights and locations, each θ_j^k represents the j-th node in mode k, $\mathcal{S}$ are the (unique) indices of all the sampled entries, $\boldsymbol{\theta}_\mathbf{i} = (\theta_{i_1}^1, \ldots, \theta_{i_K}^K)$ is the position of the corresponding point sampled by the PPP, and $c_\mathbf{i}$ is the multiplicity (or count) of the point $\boldsymbol{\theta}_\mathbf{i}$. Note that a sampled point has uniquely determined an entry index.

Although the model samples an infinite number of nodes (*i.e.,* locations) in each mode, the sampled tensor entries are finite almost surely. This is because given any fixed $\alpha > 0$, the total mass of the PPP, $b_\alpha = \prod_k W_k^\alpha([0, \alpha]) < \infty$. The count of the points sampled by the PPP follows a Poisson distribution parameterized by b_α, $p(C) = \frac{b_\alpha^C}{C!} e^{-b_\alpha}$. Hence $p(C = \infty) = 0$ and $p(C < \infty) = 1$. Our model will generate infinite entries only when $\alpha \to \infty$.

We are interested in the active nodes that actually involve in those entries, namely, the nodes whose indices show up in the sampled entry indices. If we use the active nodes to construct a tensor, the number of present entries should be much smaller than the size of this tensor — that is our intuition about sparsity. Let us denote by M_k^α the number of active nodes in mode k, and by N^α the number sampled entries. We have $M_k^\alpha = \#\{\theta_j^k | T(A_{k,\theta_j^k}^\alpha) > 0\}$, where $\#$ means "the number of", and $A_{k,\theta_j^k}^\alpha = [0, \alpha] \times \cdots \times \{\theta_j^k\} \times \cdots \times [0, \alpha]$. The sparsity of our model is guaranteed by the following lemma.

Lemma 3.1. *If the tensor entries are sampled by the model defined in (2), we have $N^\alpha = o(\prod_{k=1}^K M_k^\alpha)$ almost surely as $\alpha \to \infty$, namely,*

$$\lim_{\alpha \to \infty} \frac{N^\alpha}{\prod_{k=1}^K M_k^\alpha} = 0 \quad a.s.$$

The proof is given in the supplementary material. Note that, formally, the sparsity is defined asymptotically, namely, the growth of entries is slower than the growth of tensor size. This is consistent with the definition of sparse graphs (Caron and Fox, 2014; Crane and Dempsey, 2015; Cai et al., 2016). We can further extend the model (2) by sampling multiple measures in each mode k, and constructing a summation of the product intensity measure for the PPP to sample the entries,

$$\begin{aligned} W_{k,r}^\alpha &\sim \Gamma\mathrm{P}(\lambda_\alpha) \ \ (1 \le k \le K, 1 \le r \le R), \\ T &\sim \mathrm{PPP}(\sum_{r=1}^R W_{1,r}^\alpha \times \cdots \times W_{K,r}^\alpha). \end{aligned} \tag{3}$$

According to the Poisson process superposition theorem (Cinlar and Agnew, 1968), it is straightforward to show this model also guarantees sparsity.

Corollary 3.1.1. *If the tensor entries are sampled by the model defined in (3), we also have $N^\alpha = o(\prod_{k=1}^K M_k^\alpha)$ almost surely as $\alpha \to \infty$.*

The proof details are given in the supplementary material. We will use model (3) to estimate multiple sociabilities (reflected in overlapping communities or groups) as the embeddings of the tensor nodes (Model-II in Sec. 3.2).

3.2. Nonparametric Decomposition of Finite Observations

In practice, we can only observe a finite number of tensor entries. Accordingly, we adjust our models (2) and (3) to

sample the observed N entry indices and values. Since N is fixed, we can replace the Gamma processes by their normalized version, *i.e.,* Dirichlet processes (DP) (Ferguson, 1973), to sample N times, each time an entry. This is equivalent to the original models, but is much more convenient for inference.

Model-I. Specifically, we develop two decomposition models. Denote the dimension of the embeddings by R. The first model is based on (2), and considers the embedding of each tensor node is composed of two parts — a scalar sociability and an intrinsic property vector ($R-1$ dimensional). For each mode k, we first sample a random measure from a DP (*i.e.,* normalized ΓP),

$$L_k \sim \mathrm{DP}\big(\beta, \rho\big), \tag{4}$$

where β is the strength parameter and ρ the base probability measure. We set ρ to the product of $R-1$ uniform probability measures in $[0, \alpha]$, and $\beta = \alpha^{R-1}$. This is equivalent to extending the Lebesgue measure λ_α in (2) to $R-1$ dimensional space, with support $[0,\alpha] \times \ldots \times [0,\alpha]$. Note that this does not affect the sparsity guarantee, *i.e.,* Lemma 3.1 (see the supplementary material). We now have

$$L_k = \sum_{j=1}^{\infty} \omega_j^k \cdot \delta_{\boldsymbol{\theta}_j^k}. \tag{5}$$

We view the weights $\{\omega_j^k\}$ as the sociabilities of the nodes in mode k to interact with others so as to generate observed entries, and the locations $\{\boldsymbol{\theta}_j^k\}$ the inherent properties of these nodes that are only used to generate the entry values. The embedding of node j in mode k is defined as $\mathbf{u}_k^j = [\omega_j^k; \boldsymbol{\theta}_j^k]$. Now, given $\{L_k\}_{k=1}^K$, we independently sample N entries and their values. For the n-th entry, we first sample the node index and location in each mode k from

$$\boldsymbol{\theta}_{i_n^k} \sim L_k, \tag{6}$$

where $i_n^k \in \{1, 2, \ldots\}$. We then assemble the node indices to obtain the entry index, $\mathbf{i}_n = (i_n^1, \ldots, i_n^K)$. Given the entry index, we sample the entry value from

$$y_{\mathbf{i}_n} \sim \mathcal{N}(\cdot | f(\mathbf{x}_{\mathbf{i}_n}), \tau^{-1}), \tag{7}$$

where $\mathbf{x}_{\mathbf{i}_n} = [\mathbf{u}_{i_n^1}^1; \ldots; \mathbf{u}_{i_n^K}^K]$ is the associated embeddings, $f(\cdot)$ is a latent function, and τ is the inverse noise variance. In this work, we only consider continuous values and hence use the Gaussian distribution. We can adopt other distributions for different types of values. In order to capture the complex, nonlinear relationships between the tensor nodes, we further place a Gaussian process (GP) prior (Rasmussen and Williams, 2006) over $f(\cdot)$. Hence, the function values at all the observed entries, $\mathbf{f} = [f(\mathbf{x}_{\mathbf{i}_1}), \ldots, f(\mathbf{x}_{\mathbf{i}_N})]^\top$, follow a multivariate Gaussian prior distribution,

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K}) \tag{8}$$

where $\mathbf{K}$ is a kernel matrix on $\{\mathbf{x}_{\mathbf{i}_n}\}_{n=1}^N$, each $[\mathbf{K}]_{mn} = \kappa(\mathbf{x}_{\mathbf{i}_m}, \mathbf{x}_{\mathbf{i}_n})$, and $\kappa(\cdot,\cdot)$ is a kernel function.

Model-II. In our second model, we aim to capture richer structural information from the sparse tensor entries. To this end, we assume the nodes in each mode have R overlapping communities (or groups). Correspondingly, for each mode k, we sample R random measures from a Griffiths-Engen-McCloskey (GEM) distribution (Griffiths, 1980; Engen, 1975; McCloskey, 1965),

$$L_k^r \sim \mathrm{GEM}(\beta) \quad (1 \le r \le R, 1 \le k \le K). \tag{9}$$

A GEM distribution samples the weights in the same way as in DPs but does not sample locations (or in other words, the locations are positive integers). Therefore, we obtain

$$L_k^r = \sum_{j=1}^{\infty} \omega_{jr}^k \cdot \delta_j. \tag{10}$$

We view each weight ω_{jr}^k as the r-th sociability of node j in mode k. We use these sociabilities to construct the embedding, $\mathbf{u}_j^k = [\omega_{j1}^k, \ldots, \omega_{jR}^k]^\top$. Given the embeddings, we construct a discrete probability measure to sample the entry indices,

$$T = \sum_{\mathbf{i}=(1,\ldots,1)}^{(\infty,\ldots,\infty)} w_{\mathbf{i}} \delta_{\mathbf{i}}, \tag{11}$$

where $\mathbf{i} = (i_1, \ldots, i_K)$ is entry index, and

$$w_{\mathbf{i}} = \frac{1}{R} \sum_{r=1}^{R} \prod_{k=1}^{K} u_{i_k r}^k. \tag{12}$$

Note that we have $\sum_{\mathbf{i}} w_{\mathbf{i}} = 1$. Hence, this is a normalized version of the mean measure in (3). We independently sample N entries from T,

$$\mathbf{i}_n \sim T. \tag{13}$$

Then given each entry $\mathbf{i}_n$, we sample the entry value from

$$y_{\mathbf{i}_n} \sim \mathcal{N}(\cdot | f(\mathbf{x}_{\mathbf{i}_n}), \tau^{-1}). \tag{14}$$

Again, we place a GP prior over $f(\cdot)$ to estimate the (possible) nonlinear relationships of the tensor nodes in terms of their embeddings.

4. Algorithm

The inference of our model is challenging not only in the infinite discrete measures (see (5) and (10)), but also in the GP prior that samples the entry values (see (8)). When the number of observed entries is large, we have to compute a huge covariance matrix, its inverse and log determinant,

which is very costly. To overcome these issues, we use the stick-breaking construction (Sethuraman, 1994) and random Fourier features (Rahimi et al., 2007; Lázaro-Gredilla et al., 2010) to develop a scalable variational learning algorithm.

We will mainly discuss the estimation of the first model (Model-I). It is straightforward to extend the method for the second model (Model-II). Specifically, to obtain each DP sample L^k in (4), we use the stick-breaking construction. We first sample

$$v_j^k \sim \text{Beta}(1, \beta) \;\; (1 \leq j \leq \infty) \tag{15}$$

and then construct each ω_j^k in (5) by

$$\omega_j^k = v_j^k \prod_{t=1}^{j-1}(1 - v_t^k). \tag{16}$$

We sample each location $\boldsymbol{\theta}_j^k = [\theta_{j1}^k, \ldots, \theta_{jR-1}^k]$ in (5) from

$$p(\boldsymbol{\theta}_j^k) = \prod_{r=1}^{R-1} \text{Uniform}(\theta_{jr}^k | 0, \alpha). \tag{17}$$

Next, we consider a sparse GP approximation to deal with a large number of entry values. We will use random Fourier features, *i.e.,* the spectrum GP (Lázaro-Gredilla et al., 2010). The idea comes from the fact that a stationary kernel $\kappa(\mathbf{a}_1, \mathbf{a}_2) = \kappa(\mathbf{a}_1 - \mathbf{a}_2)$ is the Fourier transform of a probability density scaled by $\kappa(\mathbf{0})$ (Lázaro-Gredilla et al., 2010; Akhiezer and Glazman, 2013), $\kappa(\mathbf{a}_1 - \mathbf{a}_2) = \int \kappa(\mathbf{0}) e^{i\mathbf{s}^\top(\mathbf{a}_1 - \mathbf{a}_2)} \text{d}\mathbf{s} = \kappa(\mathbf{0}) \mathbb{E}_{p(\mathbf{s})}[e^{i\mathbf{s}^\top \mathbf{a}_1} \left(e^{i\mathbf{s}^\top \mathbf{a}_2}\right)^\dagger]$, where $\dagger$ is the complex conjugate. Therefore, we can view the kernel as an expectation, and from $p(\mathbf{s})$ we can draw F independent frequencies, $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_F\}$, to construct a Monte-Carlo approximation of the kernel,

$$\begin{aligned}
\kappa(\mathbf{a}_1, \mathbf{a}_2) &\approx \frac{\kappa(\mathbf{0})}{F} \sum_{m=1}^{F} e^{i\mathbf{s}_m^\top \mathbf{a}_1} \left(e^{i s_m^\top \mathbf{a}_2}\right)^\dagger = \frac{\kappa(\mathbf{0})}{F} \\
&\cdot \sum_{m=1}^{F} \left(\cos(\mathbf{s}_m^\top \mathbf{a}_1) \cos(\mathbf{s}_m^\top \mathbf{a}_2) + \sin(\mathbf{s}_m^\top \mathbf{a}_1) \sin(\mathbf{s}_m^\top \mathbf{a}_2) \right) \\
&= \frac{\kappa(\mathbf{0})}{F} \boldsymbol{\phi}(\mathbf{a}_1)^\top \boldsymbol{\phi}(\mathbf{a}_2),
\end{aligned} \tag{18}$$

where

$$\boldsymbol{\phi}(\mathbf{a}) = [\cos(\mathbf{s}_1^\top \mathbf{a}), \sin(\mathbf{s}_1^\top \mathbf{a}), \ldots, \cos(\mathbf{s}_F^\top \mathbf{a}), \sin(\mathbf{s}_F^\top \mathbf{a})].$$

We can see that $\boldsymbol{\phi}(\cdot)$ is a $2F$ dimensional nonlinear feature mapping, which is often referred to as Fourier features. Based on (18), we can approximate the GP with linear Bayesian regression over the random Fourier features. Specifically, we use the RBF kernel, $\kappa(\mathbf{a}_1, \mathbf{a}_2) = \exp(-\frac{1}{2}\eta \|\mathbf{a}_1 - \mathbf{a}_2\|^2)$. The corresponding frequency distribution (in the Fourier transform) is $p(\mathbf{s}) = \mathcal{N}(\mathbf{s}|\mathbf{0}, \eta\mathbf{I})$. We sample F frequencies $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_F\}$ ($F \ll N$) , from $p(\mathcal{S}) = \prod_{m=1}^{F} \mathcal{N}(\mathbf{s}_m|\mathbf{0}, \eta\mathbf{I})$, and a weight vector $\mathbf{h}$ for the random Fourier features, from $p(\mathbf{h}) = \mathcal{N}(\mathbf{h}|\mathbf{0}, \frac{1}{F}\mathbf{I})$. The nonlinear function $f(\cdot)$ in (7) is then constructed by

$$f(\mathbf{x}_{\mathbf{i}_n}) = \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})^\top \mathbf{h}, \tag{19}$$

where the input $\mathbf{x}_{\mathbf{i}_n}$ is from the embeddings associated with the entry $\mathbf{i}_n$, namely, the sociabilities $\{\omega_{i_{nk}}^k\}_k$ and locations $\{\boldsymbol{\theta}_{i_{nk}}^k\}_k$. Since the sociabilities can be very small and close to 0 (see (16)), we use their logarithm to ensure an effective estimation. Hence, we have

$$\mathbf{x}_{\mathbf{i}_n} = \left[\log(\omega_{i_{n1}}^1); \boldsymbol{\theta}_{i_{n1}}^1; \ldots; \log(\omega_{i_{nK}}^K); \boldsymbol{\theta}_{i_{nK}}^K\right]. \tag{20}$$

If we marginalize out $\mathbf{h}$, we can recover the joint Gaussian distribution in (8), and the kernel function takes the approximate form (18). By keeping and inferring $\mathbf{h}$, we do not need to compute the huge full covariance matrix, and therefore can scale to a large number of observations.

Denote by M_k the number of active nodes in mode k. Given the observations $\mathcal{D} = \{(\mathbf{i}_n, y_n)\}_{n=1}^N$, the joint probability of our model is given by

$$\begin{aligned}
&p(\{v_j^k, \boldsymbol{\theta}_j^k\}_{1 \leq j \leq M_k, 1 \leq k \leq K}, \mathbf{h}, \mathcal{S}, \{\mathbf{i}_n, y_n\}_{n=1}^N) \\
&= \text{Gam}(\tau|a_0, b_0) \mathcal{N}(\mathbf{h}|\mathbf{0}, \frac{\eta_0}{F}\mathbf{I}) \prod_{m=1}^{F} \mathcal{N}(\mathbf{s}_m|\mathbf{0}, \eta\mathbf{I}) \\
&\cdot \prod_{k=1}^{K} \prod_{j=1}^{M_k} \text{Beta}(v_j^k|1, \beta) \prod_r \text{Uniform}(\theta_{jr}^k|0, \alpha) \\
&\cdot \prod_{n=1}^{N} \left(\prod_{k=1}^{K} \omega_{i_{nk}}^k \right) \mathcal{N}\left(y_{\mathbf{i}_n} | \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})^\top \mathbf{w}, \tau^{-1}\right).
\end{aligned} \tag{21}$$

Note that the stick-breaking variables and locations that bind to the inactive nodes (*i.e.,* their indices do not appear in the observed entries) have been marginalized out. We use variational inference (Wainwright and Jordan, 2008) to estimate all v_j^k and θ_{jr}^k for the active nodes, and the posterior of $\mathbf{h}$. Specifically, we introduce a variational posterior $q(\mathbf{h}) = \mathcal{N}(\mathbf{h}|\boldsymbol{\mu}, \mathbf{L}\mathbf{L}^\top)$ where $\mathbf{L}$ is a lower triangular matrix to ensure the covariance is positive definite. We construct a model evidence lower bound,

$$\mathcal{L} = \mathbb{E}_q \left[\log \left(p(\text{Joint})\right)\right] + \mathbb{H}(q(\mathbf{h})), \tag{22}$$

where $p(\text{Joint})$ is (21) and $\mathbb{H}(\cdot)$ is the entropy. Obviously, the ELBO is additive over the observed entries,

$$\begin{aligned}
\mathcal{L} &= -\text{KL}(q(\mathbf{h}) \| p(\mathbf{h})) \\
&+ \log(p(\Psi)) + \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \log(\omega_{i_{nk}}^k) \right) \\
&+ \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \mathbb{E}_q \left[\log \left(\mathcal{N}(y_n | \boldsymbol{\phi}(\mathbf{x}_n)^\top \mathbf{w}, \tau^{-1})\right)\right] \right),
\end{aligned}$$

where Ψ are all the variables other than $\mathbf{h}$, and $p(\Psi)$ is their prior in (21). Accordingly, we use mini-batch stochastic optimization to efficiently maximize $\mathcal{L}$ for model estimation.

Algorithm Complexity: The time complexity of our inference is $O(NF^2)$ where N is the number of observed entries. Since F is fixed and $F \ll N$, the time complexity is linear in N. The space complexity is $O(F^2 + F + \sum_k M_k R)$, which is to store the embeddings, the frequencies, and the posterior of the weight vector.

5. Related Work

Classical and popular tensor decomposition methods include Tucker (Tucker, 1966) and CP (Harshman, 1970) decompositions. While many other approaches have also been proposed, such as (Shashua and Hazan, 2005; Chu and Ghahramani, 2009; Sutskever et al., 2009; Acar et al., 2011; Hoff, 2011; Kang et al., 2012; Yang and Dunson, 2013; Rai et al., 2014; Choi and Vishwanathan, 2014; Hu et al., 2015; Zhao et al., 2015; Rai et al., 2015; Du et al., 2018), most of them are inherently based on Tucker or CP forms, which are multilinear and inadequate to estimate complex, nonlinear relationships in data. Recently, several Bayesian nonparametric decomposition models (Xu et al., 2012; Zhe et al., 2015; 2016a;b; Pan et al., 2020; Tillinghast et al., 2020) have been developed. They use GPs to estimate the entry values as a (possible) nonlinear function of the embeddings, and hence can flexibly capture a variety of complex relationships in tensors. The exact inference of GP models is known to be prohibitively expensive for massive training examples. To overcome this problem, many sparse GP approximations have been developed, *e.g.,* (Schwaighofer and Tresp, 2003; Titsias, 2009; Lázaro-Gredilla et al., 2010; Hensman et al., 2013; 2017); see an excellent survey in (Quiñonero-Candela and Rasmussen, 2005). Zhe et al. (2016b) used the variational sparse GP (Titsias, 2009; Hensman et al., 2013) while Pan et al. (2020) the spectrum GP approximation, *i.e.,* random Fourier features; both methods achieve the state-of-the-art performance in entry value prediction.

Despite the success of the existing tensor decomposition methods, they essentially assume dense data and hence are misspecified for many sparse tensors in real-world applications. In theory, theses methods (when generalized as Bayesian models) are instances of the random function prior models for exchangeable arrays (Lloyd et al., 2012), where the number of entries is proportional to the entire array size in the limit. The pioneering work of Caron and Fox (2014; 2017) proposes to use completely random measures (Kingman, 1967; 1992; Lijoi et al., 2010) to generate asymptotically sparse graphs. Williamson (2016) considered the finite case when the number of edges is known, and hence also used DPs to develop link prediction models. NEST can be viewed as an extension of these pioneering works in the tensor domain. However, by coupling with GPs, NEST not only models the generation of the sparse (hyper-)edges, but also the entry values, *i.e.,* edge weights. The sociabilities of the tensor nodes are used as embeddings to sample both the edges and edge weights. In so doing, NEST simultaneously decomposes the sparse tensor structure and their entry values, assimilating both the structure properties and nonlinear relationships of the nodes into the embedding representation. In addition, we have developed a scalable and efficient variational model inference algorithm for large data. Recently, Crane and Dempsey (2015); Cai et al. (2016) proposed the edge-exchange random graph generation models, which also exhibit sparsity. A more thorough discussion is given in (Crane and Dempsey, 2018).

6. Experiment

6.1. Tensor Sampling

We first examined if NEST can indeed generate sparse tensors. To this end, we used the prior of NEST (see Sec. 3.2) to sample a set of tensors with growing numbers of present entries. Specifically, each time, we first sample the total mass of the mean measure of the PPP in (2) or (3) to construct a Poisson distribution, with which we sampled the number of present entries. Then we sample the indices of these entries according to the our model description in Sec. 3.2. We looked at the number of active nodes (*i.e.,* the nodes that involve in the generated entries) in each mode, and calculated the ratio between the entry number and the size of the tensor constructed from the active nodes (which we refer to as the active tensor). The ratio indicates the sparsity. Note that the active tensor is the smallest tensor that can include all the sampled entries. The number of entries is determined by α in (2) and (3). The larger α, the more entries will be generated and also the larger the size of the (active) tensor. We increased α from 1 to 10, and examined how the ratio between the entry number and tensor size (*i.e.,* sparsity) varies. For each particular α, we ran our sampling procedure for 100 times, and calculated the average ratio, which gives a reliable estimate of the sparsity. We denote our two models by NEST-1 and NEST-2. To sample the DP and GEM weights in (5) and (10), we used the stick-breaking construction. Since the weights became extremely small or simply below the machine precision when $j > 2000$, we truncated them to zero accordingly. We set $R = 3$ for both NEST-1 and NEST-2. As a comparison, we also used two popular Bayesian tensor decomposition models to sample tensor entries: CP-Bayes (Zhao et al., 2015; Du et al., 2018) and GPTF (Zhe et al., 2016b; Pan et al., 2020), where we first sampled the embeddings from the standard Gaussian distribution, and then independently sampled each entry from (1). In CP-Bayes, the function $f(\cdot)$

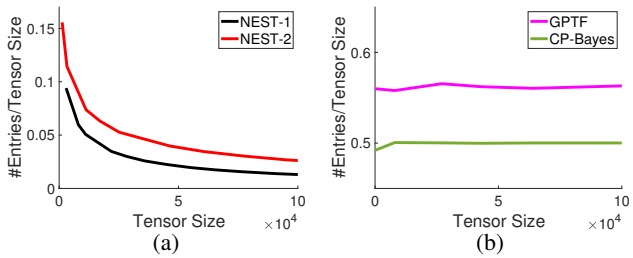


Figure 1. The ratio of the sampled entries *vs.* tensor size. Each ratio was averaged from 100 runs.

is the element-wise CP form,

$$f(\mathbf{x_i}) = \sum_{k=1}^{K} \mathbf{1}^\top (\mathbf{u}_{i_1}^1 \circ \ldots \circ \mathbf{u}_{i_K}^K)$$

where $\mathbf{x}_i = [\mathbf{u}_{i_1}^1; \ldots; \mathbf{u}_{i_K}^K]$ and $\circ$ is the element-wise product. In GPTF, $f(\cdot)$ is sampled from a GP prior. However, the exact GP prior cannot sample a large number of entries (due to the huge covariance matrix). To address this issue, we used the random Fourier features to construct a sparse GP approximation (the same as in our model estimation). We used 100 frequencies ($F = 100$). We kept increasing the the number of nodes in each mode so as to increase the tensor size and examined how the proportion of the sampled entries varied. At each tensor size, we repeated the sampling procedure for 100 times, and calculated the average ratio between the entry number and tensor size. Note that both CP-Bayes and GPTF generate exchangeable infinite arrays in the limit.

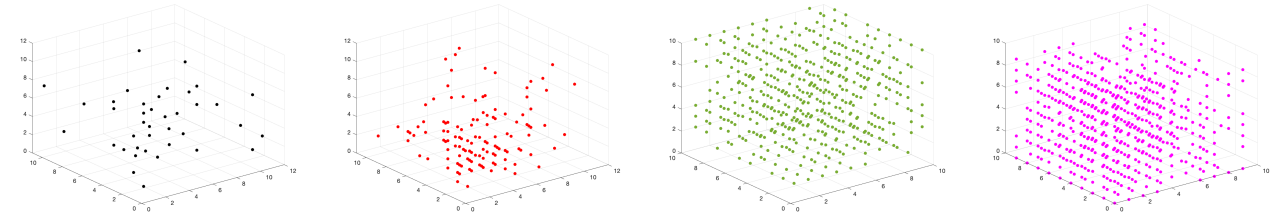
The results are reported in Fig. 1a and b. As we can see, both NEST-1 and NEST-2 generated very sparse tensors, where the number of sampled entries is way smaller than the tensor size. When the tensor size grows, the ratio between the entry number and tensor size keeps decreasing, showing a trend to converge to 0 at the limit. This is consistent with the sparse guarantee in Lemma 3.1 and Corollary 3.1.1. By contrast, both CP-Bayes and GPTF produced dense tensors where the ratio is almost constant, which implies the number of entries are (asymptotically) linear to the tensor size. We then showcase a $10 \times 10 \times 10$ tensor sampled by each model in Fig. 2. We can see that the tensors generated by CP-Bayes and GPTF are much denser, but more uniform. This is because their priors are symmetric (or exchangeable), and hence each entry has the same chance to be sampled (see Sec. 2).

6.2. Missing Entry Value Prediction

We next evaluated the predictive performance. To this end, we used three real-world datasets. (1) *Alog* (Zhe et al., 2016b), a three-mode tensor of size $200 \times 100 \times 200$. Each entry represents a three-way interaction (user, action, resource) in a file access log. The entry value is the logarithm of the access frequency. The dataset comprises $13,172$ entries, taking 0.3% of the tensor size. (2) *MovieLens* (https://grouplens.org/datasets/movielens/100k/), a three-mode tensor about movie ratings. The size is $1000 \times 1700 \times 31$. Each entry is indexed by a (user, movie, time) triple, and the entry value is the rating. There are 100K entries (0.19% of the tensor size). The entry values (ratings) were normalized in $[0, 1]$ (by dividing the highest rating 10). (3) *SG* (Li et al., 2015), a three-mode tensor extracted from data in Foursquare in Singapore. The size is $2321 \times 5596 \times 1600$. Each entry represents a (user, location, point-of-interest) check-in. The data was processed as by Liu et al. (2019). The entry value is the check-in frequency normalized in $[0, 1]$. There are $105,764$ entries, taking 0.0005% of the tensor size. As we can see, all these tensors are very sparse.

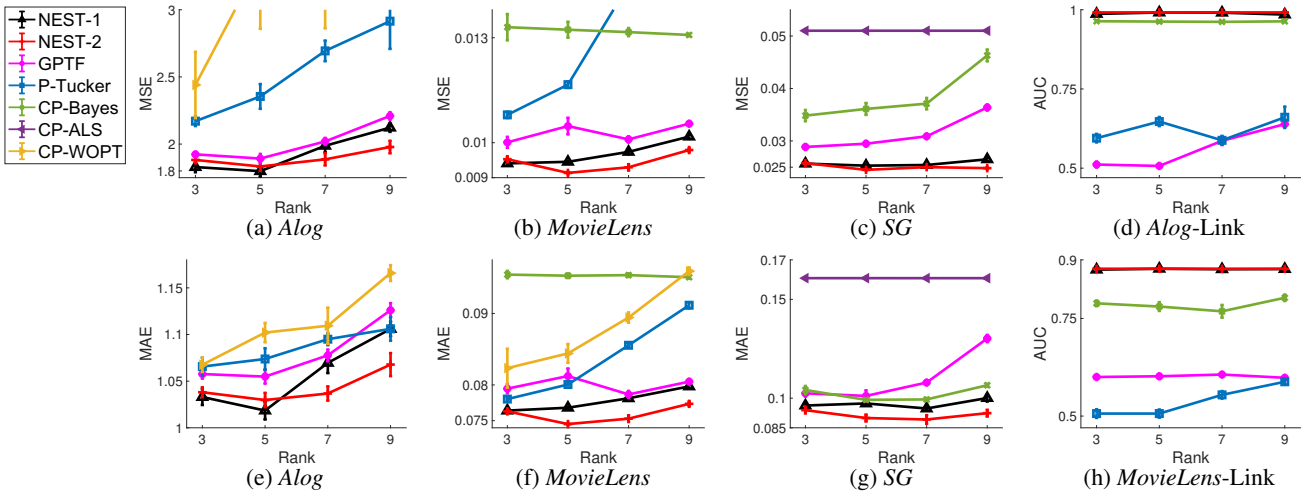
Competing Methods. We compared with the following state-of-the-art tensor decomposition methods. (1) CP-Bayes (Zhao et al., 2015; Du et al., 2018), a Bayesian version of CP decomposition that samples the embeddings from a standard Gaussian prior, and then samples each existing entry value from $p(y_\mathbf{i}|\mathcal{U}) = \mathcal{N}(y_\mathbf{i}|\sum_{k=1}^{K} \mathbf{1}^\top(\mathbf{u}_{i_1}^1 \circ \ldots \circ \mathbf{u}_{i_K}^K), \tau^{-1})$, where the mean is the element-wise CP form and τ the inverse noise variance. Note that the model is the same as the one mentioned in Sec. 6.1 except that we used a Gaussian likelihood for continuous entry values. (2) GPTF (Zhe et al., 2016b; Pan et al., 2020), the same as in the one used in Sec. 6.1 except that we used a Gaussian distribution to sample the value of each existing entry. Note that we used the same spectrum GP approximation (*i.e.,* random Fourier features) as in NEST for scalable inference. (3) CP-ALS (Bader et al., 2015), an efficient CP decomposition approach that updates the embeddings through alternating least squares. (4) P-Tucker (Oh et al., 2018), an efficient Tucker decomposition algorithm that conducts parallel row-wise updates of the embedding matrices, and (5) CP-WOPT (Acar et al., 2011), CP decomposition that uses conjugate gradient descent to optimize the embeddings.

Settings and Results. We implemented NEST-1, NEST-2, CP-Bayes and GPTF with TensorFlow (Abadi et al., 2016). All these methods used stochastic optimization, where the learning rate was chosen from $\{10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. The mini-batch size was set to 200 for *Alog* and *MovieLens*, and 512 for *SG*. To ensure convergence, we ran for 700 epochs on *Alog*, for 300 epochs on *MovieLens* and for 500 epochs on *SG* using ADAM (Kingma and Ba, 2014). For CP-ALS, P-Tucker and CP-WOPT, we used their original implementations and default settings. Note that while all the other methods decompose the existent entry values only, CP-ALS needs to fake zeros values for all the nonexistent entries so as to operate the entire tensor. For each dataset, we randomly split the existent entries into 80% for training and the remaining 20% for



(a) NEST-1 (b) NEST-2 (c) CP-Bayes (d) GPTF

Figure 2. A $10 \times 10 \times 10$ tensor generated by each method.



(a) *Alog* (b) *MovieLens* (c) *SG* (d) *Alog*-Link

(e) *Alog* (f) *MovieLens* (g) *SG* (h) *MovieLens*-Link

Figure 3. Prediction accuracy of entry values (a-c, e-g) and entry indices, *i.e.,* links (d, h). Note that in (a), part of the results of CP-WOPT was not shown because they were too large and were truncated. In (a-c, e-g), some methods performed much worse and their results were not included in the figures.

test. To evaluate the prediction accuracy, we calculated the mean-square-error (MSE) and mean-absolute-error (MAE). We varied the dimension of the embeddings, *i.e.,* rank, from $\{3, 5, 7, 9\}$. For each rank, we conducted the experiment for five times, and calculated the average MSE, average MAE and their standard deviations. We reported the results in Fig. 3 (a-c, e-g). As we can see, in all the cases, NEST-1 and NEST-2 outperform all the competing approaches, oftentimes by a large margin, demonstrating the advantage of our methods in prediction. GPTF also obtains much smaller error than the other methods, except in Fig. 3g, GPTF is worse than CP-Bayes. This might because both our method and GPTF use GPs to model the entry value as a latent function of the embeddings. The nonparametric nature of the GP prior allows us to flexibly estimate the function to capture the complex relationships in data. While compared with GPTF, NEST-1 only introduces one sociability into the embeddings, it consistently improves upon GPTF in all the case, and the improvement is often significant ($p < 0.05$). The results demonstrate that capturing and exploiting the sparse structure properties (reflected in the sociability of each tensor node) can indeed boost the embedding estimation and predictive performance. Finally, NEST-2 in general is even better than NEST-1, showing that using more sociabilities to capture richer structure information can further improve the embedding estimation, even when the internal features (like the DP locations in NEST-1) of the nodes are missing.

6.3. Missing Entry Prediction

Finally, we applied NEST to predict missing entries (*i.e.,* entry indices). This can be viewed as the link prediction task extended to tensors. We tested on *Alog* and *MovieLens* datasets. For each dataset, we randomly sampled 80% of the existent entry indices for training. We used the remaining existent entries and sampled ten times nonexistent entries for test. We expect that the prediction scores will be high for existent entries and low for nonexistent ones. We did not use all the nonexistent entries for test, because they are too many, and can dominate the test performance. We compared with CP-Bayes, GPTF and P-Tucker. To evaluate the prediction accuracy, we calculated the area under ROC curve (AUC) of predictions on the test data . For each method, we repeated

the experiment for five times, and calculated the average AUC and its standard deviation. As shown in Fig. 3d and h, NEST-1 and NEST-2 exhibit much better prediction accuracy than the competing approaches. While CP-Bayes is close to NEST on *Alog* (Fig. 3d), we found that the prediction scores of CP-Bayes have tiny differences between existing and nonexistent entries, typically 10^{-4}, showing that it actually does not capture the sparse structures and differentiate the existing entries from the nonexistent ones very well, despite the high AUC scores.

7. Conclusion

We have presented NEST, a first nonparametric decomposition approach for sparse tensors. The nonparametric prior of NEST guarantees to sample sparse tensors and addresses the model misspecification in many existing approaches. NEST can capture the valuable structure information from the sparse tensor entries, estimate the nonlinear relations between the tensor nodes, and assimilate both the structure properties and nonlinear relationships into the embeddings.

Acknowledgments

This work has been supported by NSF IIS-1910983

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283.

Acar, E., Dunlavy, D. M., Kolda, T. G., and M**o**rup, M. (2011). Scalable tensor factorizations for incomplete data. Chemometrics and Intelligent Laboratory Systems, 106(1):41–56.

Akhiezer, N. I. and Glazman, I. M. (2013). Theory of linear operators in Hilbert space. Courier Corporation.

Aldous, D. J. (1981). Representations for partially exchangeable arrays of random variables. Journal of Multivariate Analysis, 11(4):581–598.

Bader, B. W., Kolda, T. G., et al. (2015). Matlab tensor toolbox version 2.6. Available online.

Brix, A. (1999). Generalized gamma measures and shot-noise cox processes. Advances in Applied Probability, pages 929–953.

Cai, D., Campbell, T., and Broderick, T. (2016). Edge-exchangeable graphs and sparsity. In Advances in Neural Information Processing Systems, pages 4249–4257.

Caron, F. and Fox, E. B. (2014). Sparse graphs using exchangeable random measures. arXiv preprint arXiv:1401.1137.

Caron, F. and Fox, E. B. (2017). Sparse graphs using exchangeable random measures. Journal of the Royal Statistical Society. Series B, Statistical Methodology, 79(5):1295.

Choi, J. H. and Vishwanathan, S. (2014). Dfacto: Distributed factorization of tensors. In Advances in Neural Information Processing Systems, pages 1296–1304.

Chu, W. and Ghahramani, Z. (2009). Probabilistic models for incomplete multi-dimensional arrays. AISTATS.

Cinlar, E. and Agnew, R. (1968). On the superposition of point processes. Journal of the Royal Statistical Society. Series B (Methodological), pages 576–581.

Crane, H. and Dempsey, W. (2015). A framework for statistical network modeling. arXiv preprint arXiv:1509.08185.

Crane, H. and Dempsey, W. (2018). Edge exchangeable models for interaction networks. Journal of the American Statistical Association, 113(523):1311–1326.

Du, Y., Zheng, Y., Lee, K.-c., and Zhe, S. (2018). Probabilistic streaming tensor decomposition. In 2018 IEEE International Conference on Data Mining (ICDM), pages 99–108. IEEE.

Engen, S. (1975). A note on the geometric series as a species frequency model. Biometrika, 62(3):697–699.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. The annals of statistics, pages 209–230.

Griffiths, R. C. (1980). Lines of descent in the diffusion approximation of neutral wright-fisher models. Theoretical population biology, 17(1):37–50.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Model and conditions for an"explanatory"multimode factor analysis. UCLA Working Papers in Phonetics, 16:1–84.

Hensman, J., Durrande, N., and Solin, A. (2017). Variational fourier features for gaussian processes. The Journal of Machine Learning Research, 18(1):5537–5588.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, pages 282–290. AUAI Press.

Hoff, P. (2011). Hierarchical multilinear models for multiway data. Computational Statistics & Data Analysis, 55:530–543.

Hoover, D. N. (1979). Relations on probability spaces and arrays of random variables. Preprint, Institute for Advanced Study, Princeton, NJ, 2.

Hougaard, P. (1986). Survival models for heterogeneous populations derived from stable distributions. Biometrika, 73(2):387–396.

Hu, C., Rai, P., and Carin, L. (2015). Zero-truncated poisson tensor factorization for massive binary tensors. In UAI.

Kang, U., Papalexakis, E., Harpale, A., and Faloutsos, C. (2012). Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 316–324. ACM.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kingman, J. (1967). Completely random measures. Pacific Journal of Mathematics, 21(1):59–78.

Kingman, J. (1992). Poisson Processes, volume 3. Clarendon Press.

Kolda, T. G. (2006). Multilinear operators for higher-order decompositions, volume 2. United States. Department of Energy.

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

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. (2010). Sparse spectrum Gaussian process regression. Journal of Machine Learning Research, 11:1865–1881.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse spectrum gaussian process regression. The Journal of Machine Learning Research, 11:1865–1881.

Li, X., Cong, G., Li, X.-L., Pham, T.-A. N., and Krishnaswamy, S. (2015). Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, pages 433–442.

Lijoi, A., Prünster, I., et al. (2010). Models beyond the Dirichlet process. Bayesian nonparametrics, 28(80):342.

Liu, H., Li, Y., Tsang, M., and Liu, Y. (2019). CoSTCo: A Neural Tensor Completion Model for Sparse Tensors, page 324–334. Association for Computing Machinery, New York, NY, USA.

Lloyd, J. R., Orbanz, P., Ghahramani, Z., and Roy, D. M. (2012). Random function priors for exchangeable arrays with applications to graphs and relational data. In Advances in Neural Information Processing Systems 24, pages 1007–1015.

McCloskey, J. W. (1965). A model for the distribution of individuals by species in an environment. Michigan State University. Department of Statistics.

Oh, S., Park, N., Lee, S., and Kang, U. (2018). Scalable Tucker factorization for sparse tensors-algorithms and discoveries. In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pages 1120–1131. IEEE.

Pan, Z., Wang, Z., and Zhe, S. (2020). Streaming nonlinear bayesian tensor decomposition. In Conference on Uncertainty in Artificial Intelligence, pages 490–499. PMLR.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. The Journal of Machine Learning Research, 6:1939–1959.

Rahimi, A., Recht, B., et al. (2007). Random features for large-scale kernel machines. In NIPS, volume 3, page 5. Citeseer.

Rai, P., Hu, C., Harding, M., and Carin, L. (2015). Scalable probabilistic tensor factorization for binary and count data. In IJCAI.

Rai, P., Wang, Y., Guo, S., Chen, G., Dunson, D., and Carin, L. (2014). Scalable Bayesian low-rank decomposition of incomplete multiway tensors. In Proceedings of the 31th International Conference on Machine Learning (ICML).

Rasmussen, C. E. and Williams, C. K. I. (2006). Gaussian Processes for Machine Learning. MIT Press.

Schein, A., Paisley, J., Blei, D. M., and Wallach, H. (2015). Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1045–1054. ACM.

Schwaighofer, A. and Tresp, V. (2003). Transductive and inductive methods for approximate Gaussian process regression. In Advances in Neural Information Processing Systems 15, pages 953–960. MIT Press.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. Statistica sinica, pages 639–650.

Shashua, A. and Hazan, T. (2005). Non-negative tensor factorization with applications to statistics and computer vision. In Proceedings of the 22th International Conference on Machine Learning (ICML), pages 792–799.

Sutskever, I., Tenenbaum, J. B., and Salakhutdinov, R. R. (2009). Modelling relational data using bayesian clustered tensor factorization. In Advances in neural information processing systems, pages 1821–1828.

Tillinghast, C., Fang, S., Zhang, K., and Zhe, S. (2020). Probabilistic neural-kernel tensor decomposition. In 2020 IEEE International Conference on Data Mining (ICDM), pages 531–540. IEEE.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse gaussian processes. In International Conference on Artificial Intelligence and Statistics, pages 567–574.

Tucker, L. (1966). Some mathematical notes on three-mode factor analysis. Psychometrika, 31:279–311.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. Now Publishers Inc.

Williamson, S. A. (2016). Nonparametric network models for link prediction. The Journal of Machine Learning Research, 17(1):7102–7121.

Xu, Z., Yan, F., and Qi, Y. A. (2012). Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In ICML.

Yang, Y. and Dunson, D. (2013). Bayesian conditional tensor factorizations for high-dimensional classification. Journal of the Royal Statistical Society B, revision submitted.

Zhao, Q., Zhang, L., and Cichocki, A. (2015). Bayesian cp factorization of incomplete tensors with automatic rank determination. IEEE transactions on pattern analysis and machine intelligence, 37(9):1751–1763.

Zhe, S. and Du, Y. (2018). Stochastic nonparametric event-tensor decomposition. In Advances in Neural Information Processing Systems, pages 6856–6866.

Zhe, S., Qi, Y., Park, Y., Xu, Z., Molloy, I., and Chari, S. (2016a). Dintucker: Scaling up gaussian process models on large multidimensional arrays. In Thirtieth AAAI conference on artificial intelligence.

Zhe, S., Xu, Z., Chu, X., Qi, Y., and Park, Y. (2015). Scalable nonparametric multiway data analysis. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pages 1125–1134.

Zhe, S., Zhang, K., Wang, P., Lee, K.-c., Xu, Z., Qi, Y., and Ghahramani, Z. (2016b). Distributed flexible nonlinear tensor factorization. In Advances in Neural Information Processing Systems, pages 928–936.