Streaming Nonlinear Bayesian Tensor Decomposition

Zhimeng Pan, Zheng Wang, Shandian Zhe
School of Computing, University of Utah
Salt Lake City, UT 84112
z.pan@utah.edu, wzhut@cs.utah.edu, zhe@cs.utah.edu

Abstract

Despite the success of the recent nonlinear tensor decomposition models based on Gaussian processes (GPs), they lack an effective way to deal with streaming data, which are important for many applications. Using the standard streaming variational Bayes framework or the recent streaming sparse GP approximations will lead to intractable model evidence lower bounds; although we can use stochastic gradient descent for incremental updates, they are unreliable and often yield poor estimations. To address this problem, we propose Streaming Nonlinear Bayesian Tensor Decomposition (SNBTD) that can conduct high-quality, closed-form and iteration-free updates upon receiving new tensor entries. Specifically, we use random Fourier features to build a sparse spectrum GP decomposition model to dispense with complex kernel/matrix operations and to ease posterior inference. We then extend the assumed-density-filtering framework by approximating all the data likelihoods in a streaming batch with a single factor to perform one-shot updates. We use conditional moment matching and Taylor approximations to fulfill efficient, analytical factor calculation. We show the advantage of our method on four real-world applications.

1 Introduction

Tensor decomposition is a fundamental framework for multiway data analysis. While many decomposition methods have been proposed (Chu and Ghahramani, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014), they are mostly based on a multilinear form, and cannot estimate more complex, nonlinear relationships. To overcome this limitation, Zhe et al. (2016b) proposed a Bayesian nonlinear tensor decomposition model that uses nonparametric function learning, *i.e.,* Gaussian process (GP) (Rasmussen and Williams, 2006), to flexibly capture a variety of nonlinear relationships in data. On several benchmark datasets, the nonlinear model has shown a substantial improvement upon the multilinear methods in missing value prediction tasks (*i.e.,* tensor completion).

However, a critical bottleneck of the nonlinear decomposition is that it lacks an effective strategy to handle streaming data. Real-world applications are often saturated with high-velocity data streams (Du et al., 2018). It is extremely expensive or even infeasible to run the decomposition every time from scratch upon receiving a set of new entries. Therefore, we need an effective streaming estimation algorithm to update the model incrementally and responsively.

A powerful framework is streaming variational Bayes (SVB) (Broderick et al., 2013) that continuously integrates the current posterior (as a prior) with the incoming data to obtain the updated posterior in the variational approximation framework. While Du et al. (2018) have used SVB to successfully develop the state-of-the-art multilinear streaming decomposition algorithm, SVB does not work well for the nonlinear decomposition (Zhe et al., 2016b), which uses sparse variational GPs (Titsias, 2009) to estimate the latent embeddings and pseudo inputs in a variational model evidence lower bound (ELBO). To conduct SVB, we have to incorporate into the ELBO a variational posterior for the embeddings and pseudo inputs. Since they are coupled in the kernel function and the inverse and log determinant of the kernel matrices, the ELBO is not analytical and we do not have any closed-form update. Although we can use stochastic gradient descent (SGD) instead, due to its unstableness and the challenge of optimizing the intractable ELBO, SGD often fails to provide reliable, high-quality posterior updates, which in turn affects the updates for the subsequent data and ends up with a poor estimation of the embeddings. While the recent streaming sparse GP (SSGP) approxima-

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

tion (Bui et al., 2017) can avoid incrementally updating the pseudo inputs, introducing a variational posterior of the embeddings still leads to an intractable ELBO, for which we may still have to rely on SGD.

To address these issues, we propose SNBTD, a streaming nonlinear Bayesian tensor decomposition approach that performs high-quality, closed-form, and iteration-free posterior updates upon receiving new tensor entries, and hence is highly efficient to process data streams. Specifically, we first use random Fourier features to develop a nonlinear tensor decomposition model that can be viewed as a sparse spectral representation of the GP decomposition model in (Zhe et al., 2016b). We augment the model with feature weights to incorporate a linear structure so as to dispose of complex kernel computation and matrix operations. Thereby, we ease the posterior inference while preserving the nonlinear learning capability. Next, for efficient and reliable streaming inference, we extend the assumed-density-filtering (ADF) framework (Boyen and Koller, 2013) by using one single factor to approximate the likelihood of all the entries in each streaming batch. In this way, we avoid iteratively optimizing many approximation factors and instead perform real-time, one-shot update that is much faster. Finally, to bypass the intractable moment matching, we use conditional moment matching, quadrature and Taylor expansions to fulfill efficient, closed-form calculation of the approximate factor.

For evaluation, we examined SNBTD on four real-world, large-scale applications, including both binary and continuous tensors. We compared with POST (Du et al., 2018), the state of the art streaming tensor decomposition based on the multilinear, CP form (Harshman, 1970), and nonlinear streaming decomposition implemented with SVB and SSGP approximations. In both running and final predictive performance, our method consistently outperforms all the competing approaches, mostly by a large margin. Meanwhile, SNBTD spends running time much less than or comparable to POST, and is much faster than the one-by-one update as in ADF and the alternative nonlinear streaming decomposition methods.

2 Background

Tensor Decomposition. We denote a K-mode tensor by $\mathcal{Y} \in \mathbb{R}^{d_1 \times \ldots \times d_K}$. Each mode k consists of d_k entities or nodes. We index each entry with a tuple $\mathbf{i} = (i_1, \ldots, i_K)$ and denote the entry value by $y_{\mathbf{i}}$. For decomposition, we introduce K latent embedding matrices $\mathcal{U} = \{\mathbf{U}^1, \ldots, \mathbf{U}^K\}$ to represent the entities in all the K modes. Each $\mathbf{U}^k$ is $d_k \times r_k$ and the rows are the embedding vectors of the nodes in mode k. We aim to use $\mathcal{U}$ to reconstruct the observed entries in $\mathcal{Y}$. A classical method is Tucker decomposition (Tucker, 1966), which assumes $\mathcal{Y} = \mathcal{W} \times_1 \mathbf{U}^1 \times_2 \ldots \times_K \mathbf{U}^K$, where $\mathcal{W} \in \mathbb{R}^{r_1 \times \ldots \times r_K}$ is a parametric tenor and $\times_k$ is the mode-k tensor-matrix product (Kolda, 2006), which is very similar to matrix-matrix product. When we set all $r_k = r$ and $\mathcal{W}$ to be diagonal, Tucker decomposition becomes CANDECOMP/PARAFAC (CP) decomposition (Harshman, 1970).

While many tensor decomposition methods have been proposed, *e.g.,* (Chu and Ghahramani, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014), most of them are inherently based on the Tucker or CP decomposition forms. However, since both forms are mutilinear to the embeddings, they are incapable of capturing more complex, nonlinear relationships in data. To overcome this limitation, Zhe et al. (2016b) proposed a nonparametric tensor decomposition model, which learns the entry value as a (possible) nonlinear function of the embeddings, $y_{\mathbf{i}} = f(\mathbf{x}_{\mathbf{i}})$, where $\mathbf{x}_{\mathbf{i}} = [\mathbf{U}^1(i_1, :), \ldots, \mathbf{U}^K(i_K, :)]$ is the concatenation of the latent embeddings associated with each entry $\mathbf{i}$. To flexibly estimate $f(\cdot)$, Zhe et al. (2016b) assigned a Gaussian process (GP) (Rasmussen and Williams, 2006) prior. Therefore, for any collection of observed entries, $\mathcal{D} = \{\mathbf{i}_1, \ldots, \mathbf{i}_N\}$, the function values $\mathbf{f} = [f(\mathbf{x}_{\mathbf{i}_1}), \ldots, f(\mathbf{x}_{\mathbf{i}_N})]^\top$ follow a multivariate Gaussian distribution,

$$p(\mathbf{f}|\mathcal{U}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{NN}) \tag{1}$$

where $\mathbf{K}_{NN}$ is a kernel matrix on $\mathbf{X} = [\mathbf{x}_{\mathbf{i}_1}, \ldots \mathbf{x}_{\mathbf{i}_N}]^\top$ and each element is a covariance (kernel) function of the corresponding input vectors: $[\mathbf{K}_{NN}]_{n,t} = \kappa(\mathbf{x}_{\mathbf{i}_n}, \mathbf{x}_{\mathbf{i}_t})$. Given $\mathbf{f}$, we use a noise model $p(\mathbf{y}|\mathbf{f})$ to generate the observed entry values $\mathbf{y}$. For example, we can use a Gaussian noise model for continuous entries, $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \tau^{-1}\mathbf{I})$, where τ is the inverse noise variance. When the number of observed entries N is large, the computation of the $N \times N$ covariance matrix $\mathbf{K}$ (and its inverse/determinant) in (1) will be prohibitively costly. To scale up to large data, Zhe et al. (2016b) followed the sparse variational GP framework (Titsias, 2009; Hensman et al., 2013) to derive a tractable variational model evidence lower bound (ELBO), by introducing a small set of M pseudo inputs $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ ($M \ll N$). Take continuous tensors as an example. The ELBO is given by

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\log|\mathbf{K}_{MM}| - \frac{1}{2}\log|\mathbf{K}_{MM} + \tau\mathbf{K}_{MN}\mathbf{K}_{NM}| \\ &+ \frac{\tau}{2}\mathrm{tr}(\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}\mathbf{K}_{NM}) - \frac{\tau}{2}\sum_{t=1}^{N} k_{nn} + \frac{N}{2}\log(\tau) \\ &+ \frac{1}{2}\tau^2\mathbf{y}^\top\mathbf{K}_{NM}(\mathbf{K}_{MM} + \tau\mathbf{K}_{MN}\mathbf{K}_{NM})^{-1}\mathbf{K}_{MN}\mathbf{y} \\ &- \frac{\tau}{2}\sum_{n} y_n + \mathrm{const},\end{aligned} \tag{2}$$

where $\mathbf{K}_{MM}$ is the kernel matrix on the pseudo inputs $\mathbf{Z}$, $\mathbf{K}_{MN}$ is the cross kernel between $\mathbf{Z}$ and $\mathbf{X}$, each

$[\mathbf{K}_{MN}]_{mn} = \kappa(\mathbf{z}_m, \mathbf{x}_{\mathbf{i}_n})$, $\mathbf{K}_{NM} = \mathbf{K}_{MN}^\top$ and $k_{nn} = \kappa(\mathbf{x}_{\mathbf{i}_n}, \mathbf{x}_{\mathbf{i}_n})$. Now, all the inverse and (log) determinants are calculated on small matrices of size $M \times M$. The computation is scalable to large N. We maximize the ELBO (2) to estimate the embeddings $\mathcal{U}$.

Streaming Inference. Streaming variational Bayes (SVB) (Broderick et al., 2013) is a general posterior inference framework for streaming data. Denote by $\boldsymbol{\theta}$ all the latent random variables in the model, by $\mathcal{D}_t$ all the observed data before the next streaming batch $\mathcal{B}_t$ arrives, and by $q(\boldsymbol{\theta})$ the current variational posterior, *i.e.,* $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}_t)$. According to Bayes' rule, we have $p(\boldsymbol{\theta}|\mathcal{D}_t \cup \mathcal{B}_t) \propto p(\boldsymbol{\theta}|\mathcal{D}_t)p(\mathcal{B}_t|\boldsymbol{\theta})$. Therefore, to incrementally update the posterior after receiving $\mathcal{B}_t$, SVB uses $q(\boldsymbol{\theta})$ as the prior and finds $q^*(\boldsymbol{\theta}) = \operatorname{argmin}_{\hat{q}(\boldsymbol{\theta})} \mathrm{KL}\big(\hat{q}(\boldsymbol{\theta})\|q(\boldsymbol{\theta})p(\mathcal{B}_t|\boldsymbol{\theta})/C\big)$, where C is the normalization constant and KL is the Kullback-Leibler divergence. This is equivalent to maximizing a variational ELBO (Wainwright et al., 2008), $q^*(\boldsymbol{\theta}) = \operatorname{argmax}_{\widehat{q}} -\mathrm{KL}\big(\widehat{q}(\boldsymbol{\theta})\|q(\boldsymbol{\theta})\big) + \mathbb{E}_{\widehat{q}(\boldsymbol{\theta})}\log[p(\mathcal{B}_t|\boldsymbol{\theta})]$. Then SVB set $q^*(\boldsymbol{\theta})$ to $q(\boldsymbol{\theta})$ to finish the update. At the very beginning, we set $q(\boldsymbol{\theta})$ to be the original prior $p(\boldsymbol{\theta})$ in the model. Every time when a new batch of data points arrive, we repeat this procedure to update the variational posterior. For convenience and efficiency, we usually use a factorized posterior $q(\boldsymbol{\theta}) = \prod_i q(\boldsymbol{\theta}_i)$ and iteratively perform mean-field update for each $q(\boldsymbol{\theta}_i)$. Note that we never need to retrospectively access the previously seen data — they are summarized in the current posterior.

3 Model

While SVB is powerful, it does not support point estimations. To apply SVB for nonlinear decomposition (see (1) and (2)), we have to treat the embeddings $\mathcal{U}$ and pseudo inputs $\mathbf{Z}$ as random variables. That means, we need to introduce a variatonal posterior, $q(\mathcal{U}, \mathbf{Z})$, and take the expectation of $\mathcal{L}$ in (2) to construct a new ELBO for streaming updates, $\mathcal{L}^+ = -\mathrm{KL}\big(q(\mathcal{U},\mathbf{Z})\|p(\mathcal{U},\mathbf{Z})\big) + \mathbb{E}_{q(\mathcal{U},\mathbf{Z})}[\mathcal{L}]$, where $p(\mathcal{U}, \mathbf{Z})$ is the current variational posterior (now served as the prior). However, since $\mathcal{U}$ and $\mathbf{Z}$ are coupled in kernels and therefore in the matrix inverse and log determinant, the expectation $\mathbb{E}_{q(\mathcal{U},\mathbf{Z})}[\mathcal{L}]$ is not analytical. We do not have any efficient, closed-form updates to optimize $\mathcal{L}^+$. Although we can use stochastic gradient descent (SGD) (with the reparameterization trick (Kingma and Welling, 2013)), it often fails to provide a reliable, high-quality estimation of the variational posterior. First, the intractable $\mathcal{L}^+$ is complex and challenging to optimize. Second, without knowing the explicit form of $\mathcal{L}^+$, it is hard to diagnosis the convergence of the stochastic optimization. The algorithm may stop at a place far from local optimums. The inferior posterior estimation can further affect the incremental updates in the subsequent data stream, finally leading to very poor learning results. Recently, Bui et al. (2017) proposed a streaming sparse GP approximation that can avoid estimating the posterior of the pseudo inputs. However, for streaming tensor decomposition, we still need to plug in a variational posterior of the embeddings that are coupled in the kernel and complex matrix operations in the ELBO (which is even more complex than (2), see (8) in their paper). The new ELBO is therefore also intractable and we may still have to seek for stochastic optimization.

To address these issues, we propose a novel Bayesian nonlinear tensor decomposition model. Our model has the same nonlinear function learning capability, but the log joint probability is free of the complex kernel and matrix operations, and therefore the posterior inference is much easier. We then develop efficient, closed-form updates for streaming posterior inference. Our model is presented as follows.

3.1 Nonlinear Decomposition with Random Fourier Features

We first consider the spectral representation of a stationary kernel (or covariance) function $\kappa(\mathbf{a}_1, \mathbf{a}_2) = \kappa(\mathbf{a}_1 - \mathbf{a}_2)$. According to Bochner's theorem (Akhiezer and Glazman, 2013), $\kappa(\cdot)$ is the Fourier transform of a non-negative measure $\omega(\mathbf{s})$. When $\kappa(\cdot)$ is scaled properly, $\omega(\mathbf{s})$ is proportional to a probability density, $\omega(\mathbf{s}) = \kappa(\mathbf{0})p_\omega(\mathbf{s})$, which is called the spectral density of $\kappa(\cdot)$. Based on this fact, we have $\kappa(\mathbf{a}_1, \mathbf{a}_2) = \kappa(\mathbf{a}_1 - \mathbf{a}_2) = \int \omega(\mathbf{s})e^{i\mathbf{s}^\top(\mathbf{a}_1 - \mathbf{a}_2)}\mathrm{d}\mathbf{s} = \kappa(\mathbf{0})\mathbb{E}_{p_\omega(\mathbf{s})}[e^{i\mathbf{s}^\top\mathbf{a}_1}\big(e^{i\mathbf{s}^\top\mathbf{a}_2}\big)^\dagger]$, where $\dagger$ denotes the complex conjugate. Since the kernel function is essentially an expectation, we can draw M independent frequencies $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_M]^\top$ from $p_\omega(\cdot)$ and approximate $\kappa(\mathbf{a}_1, \mathbf{a}_2)$ as a Monte-Carlo summation,

$$\kappa(\mathbf{a}_1, \mathbf{a}_2) \approx \frac{\kappa(\mathbf{0})}{M}\sum_{m=1}^M e^{i\mathbf{s}_m^\top\mathbf{a}_1}\big(e^{i\mathbf{s}_m^\top\mathbf{a}_2}\big)^\dagger$$
$$= \frac{\kappa(\mathbf{0})}{M}\sum_{m=1}^M \big(\cos(\mathbf{s}_m^\top\mathbf{a}_1)\cos(\mathbf{s}_m^\top\mathbf{a}_2) + \sin(\mathbf{s}_m^\top\mathbf{a}_1)\sin(\mathbf{s}_m^\top\mathbf{a}_2)\big).$$

Therefore, we have

$$\kappa(\mathbf{a}_1, \mathbf{a}_2) \approx \frac{k(\mathbf{0})}{M}\boldsymbol{\phi}(\mathbf{a}_1)^\top\boldsymbol{\phi}(\mathbf{a}_2) \tag{3}$$

where $\boldsymbol{\phi}(\cdot)$ is a $2M$ dimensional, random Fourier feature vector. Given any input $\mathbf{a}$, we obtain $\boldsymbol{\phi}(\mathbf{a}) = [\cos(\mathbf{s}_1^\top\mathbf{a}), \sin(\mathbf{s}_1^\top\mathbf{a}), \ldots, \cos(\mathbf{s}_M^\top\mathbf{a}), \sin(\mathbf{s}_M^\top\mathbf{a})]^\top$.

Based on this kernel approximation, we propose our nonlinear tensor decomposition model. Specifically, given the collection of observed tensor entries $\mathcal{D} = \{\mathbf{i}_1, \ldots, \mathbf{i}_N\}$, we first sample M independent frequencies $\mathbf{S}$ as aforementioned. Next, we sample a feature weight vector $\mathbf{w}$ from $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \frac{k(\mathbf{0})}{M}\mathbf{I})$. Then we sample each observed entry $\mathbf{i}_n$ from a noise model with a linear structure, $p(y_{\mathbf{i}_n}|\mathcal{U}, \mathbf{w}, \mathbf{S}) = p\big(y_{\mathbf{i}_n}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})\big)$. For example, we use $p\big(y_{\mathbf{i}_n}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})\big) = \Phi\big((2y_{\mathbf{i}_n} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})\big)$ for binary tensors, where $\Phi(\cdot)$ is the cumulative density function (CDF) of the standard normal distribution, and for continuous tensors, $p(y_{\mathbf{i}_n}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})) = \mathcal{N}(y_{\mathbf{i}_n}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n}), \tau^{-1})$, where τ is the inverse noise variance. Here $\mathbf{x}_{\mathbf{i}_n}$ is the concatenation of the latent embeddings associated with entry $\mathbf{i}_n$.

In our paper, we will focus on the RBF kernel, $k(\mathbf{x}_{\mathbf{i}_m}, \mathbf{x}_{\mathbf{i}_n}) = \exp(-\frac{\|\mathbf{x}_{\mathbf{i}_m} - \mathbf{x}_{\mathbf{i}_n}\|^2}{\sigma^2})$. Since the inputs consist of the latent embeddings. We can let the length-scale σ be absorbed into the latent embeddings and only estimate the embeddings. The results are equivalent. In other words, we can simply set σ to 1. Then we have $\kappa(\mathbf{0}) = 1$ and $p_\omega(\cdot) = \mathcal{N}(\cdot|\mathbf{0}, \mathbf{I})$ (from the Fourier transform of the RBF kernel). We assign a standard Gaussian prior over the embeddings $\mathcal{U}$. For binary tensors, the joint probability of our model is

$$p(\mathcal{U}, \mathbf{S}, \mathbf{w}, \mathcal{D}) = \prod_{k=1}^{K} \prod_{j=1}^{d_k} \prod_{t=1}^{r_k} \mathcal{N}(u_{jt}^k|0, 1)$$
$$\cdot \prod_{m=1}^{M} \prod_{j=1}^{R} \mathcal{N}(s_{mj}|0, 1)\mathcal{N}(\mathbf{w}|0, M^{-1}\mathbf{I})$$
$$\cdot \prod_{\mathbf{i}\in\mathcal{D}} \Phi\big((2y_{\mathbf{i}} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}})\big), \quad (4)$$

where $s_{mj} = [\mathbf{S}]_{mj}$, $u_{jt}^k = [\mathbf{U}^k]_{jt}$ and $R = \sum_k r_k$. For continuous entries, we place a Gamma prior over the inverse noise variance τ. The joint probability is given by

$$p(\mathcal{U}, \mathbf{S}, \mathbf{w}, \tau, \mathcal{D}) = \prod_{k=1}^{K} \prod_{j=1}^{d_k} \prod_{t=1}^{r_k} \mathcal{N}(u_{jt}^k|0, 1)$$
$$\cdot \prod_{m=1}^{M} \prod_{j=1}^{R} \mathcal{N}\big(s_{mj}|0, 1\big)\mathcal{N}(\mathbf{w}|0, M^{-1}\mathbf{I})$$
$$\cdot \operatorname{Gamma}(\tau|c_0, d_0) \prod_{\mathbf{i}\in\mathcal{D}} \mathcal{N}(y_{\mathbf{i}}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}}), \tau^{-1}). \quad (5)$$

Note that if we marginalize out the feature weights $\mathbf{w}$, we will recover the kernel matrix and obtain a joint multivariate Gaussian likelihood of $\mathbf{y} = [y_{\mathbf{i}_1}, \ldots y_{\mathbf{i}_N}]$ similar to (1). Hence, our model is essentially a sparse spectrum approximation of the GP decomposition model in (1), where the kernel is approximated with the inner-product of the random Fourier features (3). Here, we keep $\mathbf{w}$ and the linear structure in the likelihood so that our joint probability will not include any complicated kernel computation and matrix operations, such as inverse and log determinant. Therefore, it is much simpler than the ELBO of the variational sparse GP (see (2)), and the posterior inference of the embeddings can be easier and more convenient.

4 Algorithm

We now present SNBTD, the streaming posterior inference algorithm of our nonlinear tensor decomposition model. In general, we assume the observed tensor entries are streamed in a series of small batches, $\{\mathcal{B}_1, \mathcal{B}_2, \ldots\}$. Different batches do not necessarily include the same number of entries. We aim to update the posterior of the latent embeddings $\mathcal{U}$, the weights $\mathbf{w}$ and the frequencies $\mathbf{S}$ upon receiving each batch $\mathcal{B}_t$, without using the previously accessed batches $\{\mathcal{B}_1, \ldots, \mathcal{B}_{t-1}\}$.

4.1 One-Shot Posterior Update

First, to avoid optimizing intractable ELBOs as in SVB, we explore assumed-density-filtering (ADF) (Boyen and Koller, 1998), which processes data points one by one and upon each data point applies expectation propagation (EP) (Minka, 2001) to update the posterior (analytically). Specifically, let us consider binary tensors as an example. When a new data point $(\mathbf{i}, y_{\mathbf{i}})$ arrives, we first replace the prior in our model (4) by the current posteriors $q(\mathcal{U}, \mathbf{w}, \mathbf{S})$. We then combine with the likelihood of the data point to construct a blending distribution, $p_b(\mathcal{U}, \mathbf{S}, \mathbf{w}) \propto q(\mathcal{U}, \mathbf{S}, \mathbf{w})\Phi\big((2y_{\mathbf{i}} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}})\big)$. Since the blending distribution is intractable to compute, we project it to the exponential family to obtain the updated posterior (with a nice and easy form). The projection is done by moment matching, which is the key step of EP and essentially minimizes the KL divergence between p_b and the approximate posterior. To illustrate it, suppose $q(\mathcal{U}, \mathbf{w}, \mathbf{S})$ includes a fully factorized Gaussian distribution for $\mathcal{U}$. To update each $q(u_{jt}^k) = \mathcal{N}(u_{jt}^k|\mu_{jt}^k, v_{jt}^k)$, we need to compute the first and second order moments, $\mathbb{E}_{p_b}[u_{jt}^k]$ and $\mathbb{E}_{p_b}[(u_{jt}^k)^2]$, and find a Gaussian distribution with the same moments. The matched Gaussian is the updated posterior. Obviously, this is can be done by setting $\mu_{jt}^k = \mathbb{E}_{p_b}[u_{jt}^k]$ and $v_{jt}^k = \mathbb{E}_{p_b}[(u_{jt}^k)^2] - \mathbb{E}_{p_b}[u_{jt}^k]^2$.

Although ADF is efficient, it only handles one data point at a time. When we receive a batch of entries $\mathcal{B}_t = \{\mathbf{i}_1, \ldots, \mathbf{i}_N\}$, ADF cannot jointly integrate all the data points to improve the robustness/quality of the posterior update. To address this problem, one can return to the standard EP, where we introduce a factor $\hat{f}_n(\mathcal{U}, \mathbf{S}, \mathbf{w})$ in the exponential family to approximate the likelihood of each data point n,

$$p_b(\mathcal{U}, \mathbf{S}, \mathbf{w}) \propto q(\mathcal{U}, \mathbf{S}, \mathbf{w}) \prod_{\mathbf{i}_n\in\mathcal{B}_t} \Phi((2y_{\mathbf{i}_n} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n}))$$
$$\approx \widehat{q}(\mathcal{U}, \mathbf{S}, \mathbf{w}) \propto q(\mathcal{U}, \mathbf{S}, \mathbf{w}) \prod_{\mathbf{i}_n\in\mathcal{B}_t} \hat{f}_n(\mathcal{U}, \mathbf{S}, \mathbf{w}). \quad (6)$$

We then iteratively update each approximate factor $\hat{f}_n$ with three steps: (1) computing the calibrating distribution, $q^{\backslash n} \propto \widehat{q}/\hat{f}_n$, (2) constructing a tilted distribution $\propto q^{\backslash n}(\mathcal{U}, \mathbf{S}, \mathbf{w})\Phi\big((2y_{\mathbf{i}} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}})\big)$, then projecting it back to the exponential family to obtain $q^*(\mathcal{U}, \mathbf{S}, \mathbf{w})$ via moment matching, and (3) updating the approximate factor, $\hat{f}_n \propto q^*/q^{\backslash n}$.

While the standard EP can jointly exploit the information of the data batch, it needs extra storage for the approximate factor of each data point and iteratively updates all the factors. Therefore, it is much more costly than ADF. In order to unify the benefits of both ADF and EP, we use one single factor to approximate the product of all the likelihoods,

$$\hat{f}(\mathbf{U}, \mathbf{S}, \mathbf{w}) \approx \prod_{\mathbf{i}_n \in \mathcal{B}_t} \Phi((2y_{\mathbf{i}_n} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})).$$

We then have

$$p_b(\mathcal{U}, \mathbf{S}, \mathbf{w}) \approx \widehat{q}(\mathcal{U}, \mathbf{S}, \mathbf{w}) \propto q(\mathcal{U}, \mathbf{S}, \mathbf{w})\hat{f}(\mathbf{U}, \mathbf{S}, \mathbf{w}).$$

Now the calibrating distribution to update $\hat{f}$ is simply the prior (*i.e.,* the current posterior) $q(\mathcal{U}, \mathbf{S}, \mathbf{w})$, and so the tilted distribution is $p_b(\mathcal{U}, \mathbf{S}, \mathbf{w})$. We then use moment matching to obtain $q^*(\mathcal{U}, \mathbf{S}, \mathbf{w})$. Since there is only one approximate factor and we do not need to update other factors (iteratively), the optimal $\hat{f}$ is obtained by matching the moments just once. Furthermore, because there are no iterations, we even do not need to explicitly update $\hat{f}$; instead, we can directly set the current posterior $q(\mathcal{U}, \mathbf{S}, \mathbf{w})$ to $q^*(\mathcal{U}, \mathbf{S}, \mathbf{w})$ and prepare for the next batch. In this way, we fulfill a one-shot update as in ADF, but we jointly integrate all the data points to improve the quality/robustness of the posterior update; our method is even more efficient, because we do not go through each data point and match the moments one by one. Similarly, for continuous data, we use the current posterior $q(\mathcal{U}, \mathbf{S}, \mathbf{w}, \tau)$ as the prior and construct the blending distribution, $p_b(\mathcal{U}, \mathbf{S}, \mathbf{w}, \tau) \propto q(\mathcal{U}, \mathbf{S}, \mathbf{w}, \tau) \prod_{\mathbf{i} \in \mathcal{B}_t} \mathcal{N}(y_{\mathbf{i}} | \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}}), \tau^{-1})$ (see (5)). We then introduce one single approximate factor $\hat{f}(\mathbf{U}, \mathbf{S}, \mathbf{w}, \tau) \approx \prod_{\mathbf{i} \in \mathcal{B}_t} \mathcal{N}(y_{\mathbf{i}} | \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}}), \tau^{-1})$ to perform the one-shot update.

4.2 Closed-Form Moment Approximation

The next step is to compute the moments from the blending distribution ($p_b(\mathcal{U}, \mathbf{w}, \mathbf{S})$ for binary and $p_b(\mathcal{U}, \mathbf{w}, \mathbf{S}, \tau)$ for continuous tensors), to update the posterior upon receiving new entry batch $\mathcal{B}_t$. However, the exact moments are infeasible to compute, because the embeddings $\mathcal{U}$ and frequencies $\mathbf{S}$ are coupled in the nonlinear random Fourier features in the data likelihood (see (4) and (5)). To address this issue, we use the conditional moment matching (Wang and Zhe, 2019), Gauss-Hermite quadrature and Taylor expansion to fulfill an effective, closed-form moment approximation, based on which we can conduct highly-efficient and reliable posterior updates.

Specifically, let us first consider binary tensors. We will use a factorized posterior,

$$q(\mathcal{U}, \mathbf{S}, \mathbf{w}) = \prod_{k=1}^{K} \prod_{j=1}^{d_k} \prod_{t=1}^{r_k} q(u_{jt}^k) \prod_{m=1}^{M} \prod_{j=1}^{R} q(s_{mj}) \cdot q(\mathbf{w})$$

where each $q(u_{jt}^k) = \mathcal{N}(u_{jt}^k | \mu_{jt}^k, v_{jt}^k)$, $q(s_{mj}) = \mathcal{N}(s_{mj} | \alpha_{mj}, \rho_{mj})$ and $q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \boldsymbol{\eta}, \boldsymbol{\Sigma})$. We first look into the update of each frequency's posterior $q(s_{mj})$. For convenience, we denote $\boldsymbol{\theta} = \{\mathcal{U}, \mathbf{w}, \mathbf{S}\}$. To update $q(s_{mj})$ upon receiving $\mathcal{B}_t$, we observe that the blending distribution (see (6)) can be decomposed as

$$p_b(\boldsymbol{\theta}) = p_b(\boldsymbol{\theta}_{\backslash s_{mj}}) p_b(s_{mj} | \boldsymbol{\theta}_{\backslash s_{mj}})$$

where $\boldsymbol{\theta}_{\backslash s_{mj}} = \boldsymbol{\theta} \setminus \{s_{mj}\}$ and $p_b(s_{mj} | \boldsymbol{\theta}_{\backslash s_{mj}}) \propto q(s_{mj}) \prod_{\mathbf{i}_n \in \mathcal{B}_t} \Phi\big((2y_{\mathbf{i}_n} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n})\big)$. Therefore, for the first order moment (*i.e.,* mean), we have

$$\mathbb{E}_{p_b}[s_{mj}] = \mathbb{E}_{p_b(\boldsymbol{\theta}_{\backslash s_{mj}})} \mathbb{E}_{p_b(s_{mj} | \boldsymbol{\theta}_{\backslash s_{mj}})}[s_{mj}]. \tag{7}$$

To approximate this intractable moment, we first calculate the conditional moment $\mathbb{E}_{p_b(s_{mj}|\boldsymbol{\theta}_{\backslash})}[s_{mj}]$. Although it still is intractable, because the expectation is only taken over a scalar and all the other random variables (*i.e.,* $\boldsymbol{\theta}_{\backslash s_{mj}}$) are fixed, we can analytically represent the conditional moment via Gauss-Hermite quadrature,

$$\mathbb{E}_{p_b(s_{mj} | \boldsymbol{\theta}_{\backslash s_{mj}})}[s_{mj}] = \frac{\sum_j \beta_j \gamma_j g(\gamma_j, \boldsymbol{\theta}_{\backslash s_{mj}})}{\sum_j \beta_j g(\gamma_j, \boldsymbol{\theta}_{\backslash s_{mj}})} = h(\boldsymbol{\theta}_{\backslash s_{mj}}),$$

where $\{\gamma_j, \beta_j\}$ are quadrature nodes and weights, $g(\cdot) = \prod_{\mathbf{i}_n \in \mathcal{B}_t} \Phi\big((2y_{\mathbf{i}} - 1)\mathbf{w}^\top \phi(\mathbf{x}_{\mathbf{i}_n})\big)$, and the denominator is the quadrature for the normalizer. Note that the quadrature nodes are determined by the current posterior $q(s_{mj})$.

The next step is to compute the expectation of the conditional moment $\mathbb{E}_{p_b(\boldsymbol{\theta}_{\backslash s_{mj}})}[h(\boldsymbol{\theta}_{\backslash s_{mj}})]$, which, however, is infeasible as well, because the marginal blending distribution $p_b(\boldsymbol{\theta}_{\backslash s_{mj}})$ is intractable. To address this problem, we observe that due to the factorized posterior structure, the moment matching is also maintained between $p_b(\boldsymbol{\theta}_{\backslash s_{mj}})$ and $q(\boldsymbol{\theta}_{\backslash s_{mj}})$. In other words, we can assume they are close in high density regions. Hence, we can use the current posterior $q(\boldsymbol{\theta}_{\backslash s_{mj}})$ as a surrogate of $p_b(\boldsymbol{\theta}_{\backslash s_{mj}})$, and compute $\mathbb{E}_q[h(\boldsymbol{\theta}_{\backslash s_{mj}})]$ instead. Now, with the nice form of q (in the exponential family), even the expectation is still intractable, we can apply the first-order Taylor approximation at the mean,

$$\begin{aligned} \hat{h}(\boldsymbol{\theta}_{\backslash s_{mj}}) &= h(\mathbb{E}_q[\boldsymbol{\theta}_{\backslash s_{mj}}]) \\ &+ \nabla h(\mathbb{E}_q[\boldsymbol{\theta}_{\backslash s_{mj}}])^\top (\boldsymbol{\theta}_{\backslash s_{mj}} - \mathbb{E}_q[\boldsymbol{\theta}_{\backslash s_{mj}}]). \end{aligned}$$

Then we have

$$\mathbb{E}_q[h(\boldsymbol{\theta}_{\backslash s_{mj}})] \approx \mathbb{E}_q[\hat{h}(\boldsymbol{\theta}_{\backslash s_{mj}})] = h(\mathbb{E}_q[\boldsymbol{\theta}_{\backslash s_{mj}}]). \tag{8}$$

We can use the same approach to compute the second order moment. Therefore, to approximate the moments of s_{mj}, we only need to represent the conditional moment by quadrature formulas, and replace all the other random variables inside the quadrature by the mean of their current posterior. This is analytical and straightforward. Note that although we can also use the second-order Taylor expansion, we did not find improvement in the experiments. Similarly, we can compute the first/second moments of all elements $\mathcal{U}$ and update each $\{q(u_{jt}^k)\}$ accordingly.

Now we look at $q(\mathbf{w})$. If we choose a fully factorized form for $q(\mathbf{w})$, we can apply the same approach to update each $q(w_j)$. Here we use a joint Gaussian $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\eta}, \boldsymbol{\Sigma})$ to preserve the posterior correlations and to further enhance the inference quality. To achieve a closed-form update, we observe that the likelihood of each data point, $\Phi((2y_{\mathbf{i}_n} - 1)\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n}) = \int \mathcal{N}(z_{\mathbf{i}}|\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_n}), 1)\mathbb{1}(z_{\mathbf{i}}(2y_{\mathbf{i}} - 1) \geq 0)\mathrm{d}z_{\mathbf{i}}$ where $\mathbb{1}(\cdot)$ is the indicator function. Therefore, we can augment the blending distribution and obtain $p_b(\mathcal{U}, \mathcal{S}, \mathbf{w}, \mathbf{z}) = q(\mathcal{U}, \mathcal{S}, \mathbf{w})\mathcal{N}(\mathbf{z}|\boldsymbol{\Phi}\mathbf{w}, \mathbf{I}) \prod_{\mathbf{i}\in\mathcal{B}_t} \mathbb{1}(z_{\mathbf{i}}(2y_{\mathbf{i}}-1) \geq 0)$ where $\mathbf{z} = [z_{\mathbf{i}_1}, \ldots, z_{\mathbf{i}_N}]$ and $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_1}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}_N})]^\top$. We then derive the conditional moments of $\mathbf{w}$ based this augmented distribution, which is analytically tractable,

$$\mathbb{E}(\mathbf{w}|\boldsymbol{\theta}_{\backslash\mathbf{w}}) = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}(\boldsymbol{\Phi}\mathbf{z} + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}),$$
$$\mathbb{E}(\mathbf{w}\mathbf{w}^\top|\boldsymbol{\theta}_{\backslash\mathbf{w}}) = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1} + \mathbb{E}(\mathbf{w}|\boldsymbol{\theta}_{\backslash\mathbf{w}})\mathbb{E}(\mathbf{w}|\boldsymbol{\theta}_{\backslash\mathbf{w}})^\top. \quad (9)$$

Note that now $\boldsymbol{\theta}_{\backslash\mathbf{w}} = \{\mathcal{S}, \mathcal{U}, \mathbf{z}\}$. To compute the moments, we first calculate $\mathbb{E}_{p_b}(\mathbf{z})$ via conditional moment matching again. We derive each $\mathbb{E}_{p_b}(z_{\mathbf{i}}|\mathcal{U}, \mathcal{S}, \mathbf{w})$ under $p_b(\mathbf{z}_i|\mathbf{w}, \mathcal{U}, \mathbf{S}) \propto \mathcal{N}(z_{\mathbf{i}}|\boldsymbol{\phi}(\mathbf{x}_{\mathbf{i}})^\top \mathbf{w}, 1)\mathbb{1}\left(z_{\mathbf{i}}(2y_{\mathbf{i}}-1) \geq 0\right)$ and then replace each frequency s_{mj}, each latent factor u_{jt}^k and $\mathbf{w}$ with their current posterior mean. This is equivalent to taking the expectation of the first Taylor approximation of the conditional mean (see (8)). Next, we substitute $\mathbb{E}_{p_b}(\mathbf{z})$ and $\mathbb{E}_q(\mathcal{U})$ and $\mathbb{E}_q(\mathcal{S})$ for $\mathbf{z}$, $\mathcal{S}$ and $\mathcal{U}$, respectively into the conditional moments of $\mathbf{w}$ in (9) to obtain the approximate moments and to update $q(\mathbf{w})$.

For efficiency, we compute all the required moments in parallel and obtain the new posteriors $\{q(u_{jt}^k), q(s_{mj}), q(\mathbf{w})\}$ simultaneously.

The posterior update for continuous data are similar. We use the following factorized posterior, $q(\mathcal{U}, \mathbf{S}, \mathbf{w}, \tau) = \prod_{k=1}^K \prod_{j=1}^{d_k} \prod_{t=1}^{r_k} q(u_{jt}^k) \prod_{m=1}^M \prod_{j=1}^R q(s_{mj})q(\mathbf{w})q(\tau)$, where each $q(u_{jt}^k)$ and $q(s_{mj})$ is a scalar Gaussian, $q(\tau) = \mathrm{Gamma}(\tau|c, d)$ and $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\eta}, \boldsymbol{\Sigma})$. The conditional moments for $\mathbf{w}$ are the same as (9) except that we replace $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ by $\tau\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ and $\boldsymbol{\Phi}\mathbf{z}$ by $\tau\boldsymbol{\Phi}\mathbf{y}$. The conditional moments of τ are $\mathbb{E}(\tau|\boldsymbol{\theta}_{\backslash\tau}) = (c + \frac{N}{2})/(d + \frac{1}{2}\|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|^2)$ and $\mathrm{Var}(\tau|\boldsymbol{\theta}_{\backslash\tau}) = (c + \frac{N}{2})/(d + \frac{1}{2}\|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|^2)^2$. We then substitute $\mathbb{E}_q(\boldsymbol{\theta}_{\backslash\mathbf{w}})$ and $\mathbb{E}_q(\boldsymbol{\theta}_{\backslash\tau})$ for $\boldsymbol{\theta}_{\backslash\mathbf{w}}$ and $\boldsymbol{\theta}_{\backslash\tau}$ respectively, to obtain the approximation of the moments, and update $q(\mathbf{w})$ and $q(\tau)$ accordingly. The updates for each $q(u_{jt}^k)$ and $q(s_{mj})$ are similar to that in the binary case. Finally, our streaming nonlinear decomposition is summarized in Algorithm 1 in the appendix.

4.3 Algorithm Complexity

The time complexity of our algorithm in processing one streaming batch is $\mathcal{O}\left(N(\sum_k d_k r_k + MR + 4M^2)\right)$ for continuous data. Therefore, the time complexity is proportional to N, the size of the streaming batch. The space complexity is $\mathcal{O}(\sum_k d_k r_k + MR + M^2)$ for continuous data, which is to store the posteriors for the embeddings, frequencies and feature weights.

5 Related Work

Classical Tucker (Tucker, 1966) and CP (Harshman, 1970) decomposition are multilinear and therefore cannot estimate complex, nonlinear relationships in data. While many other approaches have also been proposed (Shashua and Hazan, 2005; Chu and Ghahramani, 2009; Acar et al., 2011; Hoff, 2011; Yang and Dunson, 2013; Rai et al., 2014; Choi and Vishwanathan, 2014; Hu et al., 2015), they are mostly based on Tucker or CP forms. To overcome their limitation, several Bayesian nonparametric decomposition models (Xu et al., 2012; Zhe et al., 2015, 2016a,b) were recently proposed. These methods use GPs to flexibly estimate a variety of nonlinear relationships in tensors. Our approach adopts the same GP modeling choice as in (Zhe et al., 2016b). The standard GP is known to be prohibitively costly for large data. Accordingly, many sparse GP approximations have been proposed, *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 approximation (Titsias, 2009; Hensman et al., 2013) to develop a distributed estimation algorithm.

Expectation propagation (Minka, 2001) is a powerful approximate posterior inference algorithm that generalizes the assumed-density-filtering (ADF) (Boyen and Koller, 1998) and (loopy) belief propagation (Murphy et al., 1999). EP uses an exponential-family factor to approximate the likelihood of each data point (and also the prior) in a probabilistic model, and iteratively update each approximate factor by moment matching, which is essentially to minimize a local KL divergence. EP updates are fixed-point iterations to find a stationary point of the corresponding energy function. The updates in ADF can be viewed as EP inference for a model including only one data point. While often being fast and accurate, EP can be troublesome when the moment matching is intractable. While in such cases, importance sampling is a straightforward solution, it is very inefficient and often unreliable. Recently, Wang and Zhe (2019) proposed conditional EP that uses conditional moment matching and Taylor approximations to provide a high-quality, analytical solution. In our work, to improve the quality and efficiency for streaming inference, we extend ADF by approximating the product of all the data likelihoods in each streaming batch with one single factor, so that we only need to perform a one-shot update. To fulfill reliable, closed-form one-shot updates, we then combine conditional moment matching, Gauss-Hermite quadrature and Taylor approximations.

6 Experiment

6.1 Predictive Performance

Datasets. For evaluation, we examined SNBTD on four real-world, large-scale datasets. (1) *DBLP* (Du et al., 2018), a binary tensor depicting three way bibliography relationships *(author conference, keyword)*. The tensor is $10,000 \times 200 \times 10,000$, including 0.001% nonzero entries. (2) *ACC* (Du et al., 2018), a continue tensor of size $3,000 \times 150 \times 30,000$, which was extracted from the log of a code repository and records the three-way interactions *(user, action, resource)*. The entry values are the logarithm of the resource access frequencies. The tensor contains 0.9% nonzero entries. (3) *Anime*(`https://www.kaggle.com/CooperUnion/anime-recommendations-database`), a two-mode binary tensor describing *(user, anime)* preferences (each entry value is whether a user likes an anime or not). The tensor is $25,838 \times 4,066$ and includes $1,300,160$ observed entries. (4) *MovieLen1M* (`https://grouplens.org/datasets/movielens/`), a two-mode continuous tensor of size $6,040 \times 3,706$, recording *(user, movie)* ratings. There are $1,000,209$ observed entries.

Competing methods. We compared with the following baselines. (1) POST (Du et al., 2018), the state-of-the-art streaming tensor decomposition algorithm based on a probabilistic CP model. It uses streaming variational Bayes (SVB) (Broderick et al., 2013) to perform mean-field posterior updates upon receiving newly observed entries. (2) SVB-NTD, SVB based nonlinear tensor decomposition. It introduces a variational posterior (*i.e.,* fully factorized Gaussian) of the latent embeddings and pseudo inputs in the sparse variational GP framework (Hensman et al., 2013) to construct an (intractable) ELBO for streaming inference. It uses the reparameterization trick (Kingma and Welling, 2013) and stochastic gradient descent (SGD) to optimize the ELBO. (3) SSGP-NTD, nonlinear tensor decomposition based on the recent streaming sparse GP approximations (Bui et al., 2017). Again, it uses SGD to optimize the ELBO. (4) CP-WOPT (Acar et al., 2011), a scalable static CP decomposition algorithm implemented with gradient-based optimization. Finally, we also tested our method to process streaming entries one by one. We denote this variant by (5) SNBTD-1.

Parameter Settings. We implemented our methods (SNBTD and SNBTD-1) with Python + Numpy/Scipy library, and SVB-NTD and SSGP-NTD with TensorFlow. For POST, we used the original MATLAB implementation (`https://github.com/yishuaidu/POST`). We set the number of pseudo inputs in SVB/SSGP-NTD and frequencies in our methods to 128. We used Adam (Kingma and Ba, 2014) for the stochastic optimization in SVB/SSGP-NTD, for which we set the number of epochs to 100 in processing each streaming batch and chose the learning rate from $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$ on extra validation datasets. Like our method, SVB/SSGP-NTD used RBF kernel with the length-scale set to 1, which is equivalent to letting any free length-scale parameter be absorbed into the latent embeddings. To compare the running speed, we ran all the methods on a server with 64GB memory and a single Intel i7-9700K CPU.

We first examined the predictive performance of each method after processing all the (accessible) tensor entries. To this end, we sequentially fed a collection of training entries into every streaming decomposition method, each time with a small batch. Then we evaluated the prediction accuracy on the test entries. We fixed the batch size to 256. We used the mean-squared-error (MSE) and area under ROC curves (AUC) for continuous and binary data, respectively. For the static decomposition algorithm CP-WOPT, we ran it on all the training entries. On *DBLP* and *ACC*, we used the same set of training and test entries as in (Du et al., 2018), including 320K and 1M training entries for *DBLP* and *ACC* respectively, and 100K test entries for both. We randomly split the observed entries with the training/test ratio being 1/1 and 9/1 on *Anime* and *MovieLen1M*, respectively. For each streaming decomposition algorithm, we randomly shuffled the training entries and then partitioned them into a stream of entry batches. On each dataset, we repeated the test for 5 times and computed the average of MSEs/AUCs and standard deviations. For CP-WOPT, we used a different, random initialization in each test. We varied the rank of the embeddings, namely the dimension of each embedding vector, denoted by r, from $\{3, 5, 8, 10\}$. The performance of all the methods is reported in Fig. 1a-d.

As we can see, our methods (including both SNBTD and SNBTD-1) outperform all the competing baselines in all the cases and mostly by a large margin. In particular, SNBTD significantly improves upon POST and CP-WOPT — the streaming and static multilinear de-

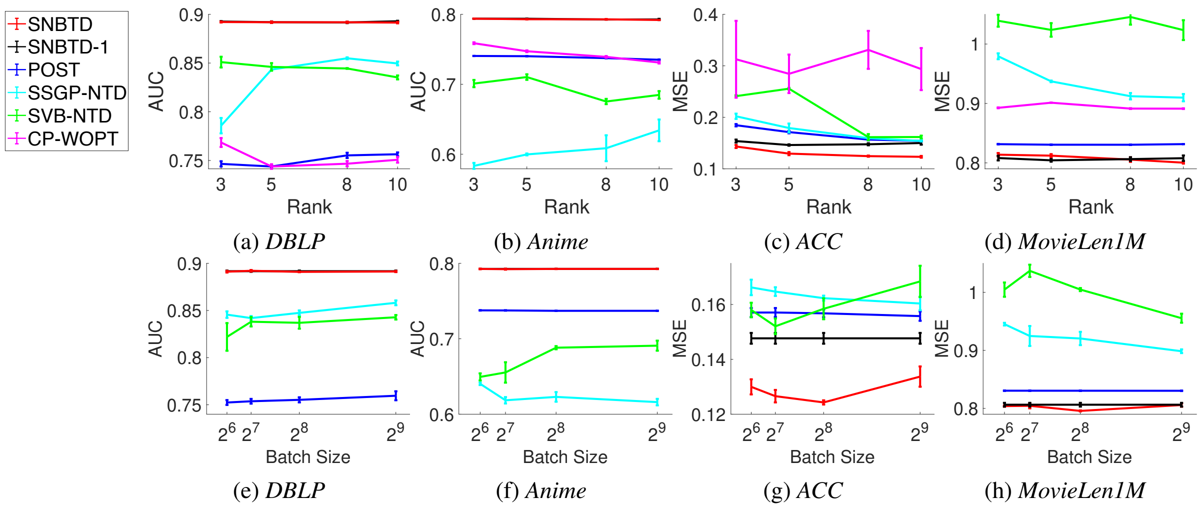

Figure 1: Predictive performance with different ranks (top row) and streaming batch sizes (bottom row). In the top row, the streaming bath size is fixed to 256; in the bottom row, the rank is fixed to 8. The results are averaged over 5 runs.

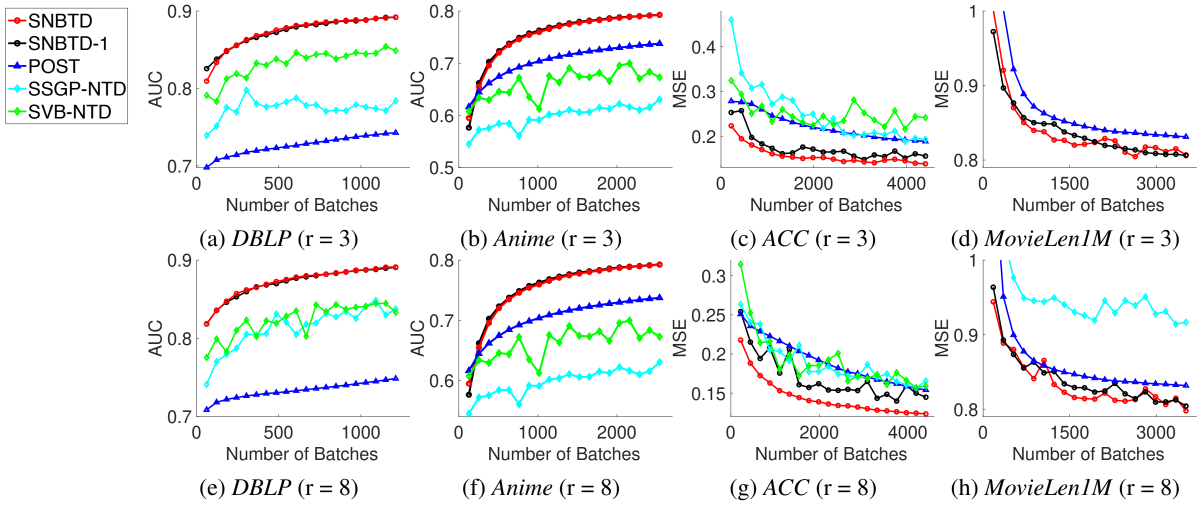

Figure 2: Running prediction accuracy along with the number of processed streaming batches. The batch size was fixed to 256.

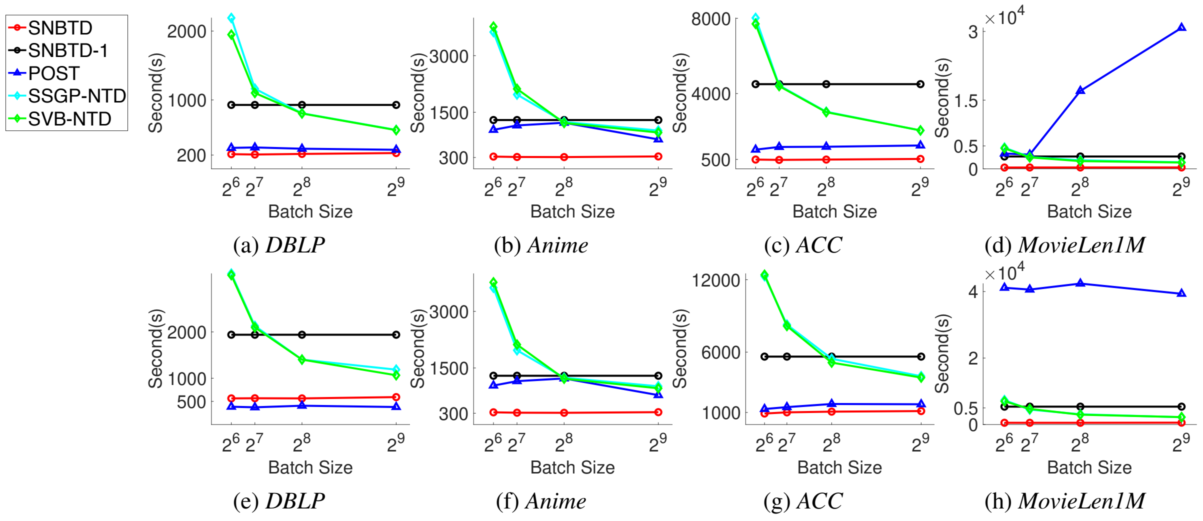

Figure 3: Running time with different streaming batch sizes. The rank was set to 3 and 8 in the top and bottom rows, respectively.

compositions, confirming the advantages of the nonlinear decomposition. It is interesting to see that CP-WOPT is much worse than POST on *ACC* and *MovieLen1M*. The reason might be CP-WOPT converged to poor local optimums. Generally, SVB/SSGP-NTD are far worse than our methods, and in many cases even worse than POST (see Fig. 1b,c and d). This might be due to the inferior/unreliable stochastic posterior updates. Lastly, SNBTD and SNBTD-1 achieved almost the same prediction accuracy on binary tensors (see Fig. 1a and b; their curves overlap) and close MSEs on *MovieLen1M*. But on *ACC*, SNBTD achieved significantly smaller MSEs than SNBTD-1 (see Fig. 1c, $p < 0.05$). This demonstrates the benefit of our one-shot update (Sec. 4.1) that jointly integrates all the entries in each streaming batch, rather than one by one, to further improve the quality of posterior updates.

Second, we examined how the predictive performance of the streaming methods varies along with the size of the streaming batches. To this end, we set the rank of the embeddings to 8, and tested with the batch size from $\{2^6$, 2^7, 2^8, $2^9\}$. Similarly as above, we randomly shuffled the training entries to generate the sequence of streaming batches. On each dataset, we repeated the test for 5 times, and calculated the average MSE/AUC and their standard deviations. We show the results in Fig. 1e-h. It can be seen that SNBTD and SNBTD-1 consistently outperform all the competing methods by a large margin. The prediction accuracy of SNBTD-1 is close to SNBTD on *DBLP* and *Anime* (Fig. 1e and f), slightly worse than SNBTD on *MovieLen1M* (Fig. 1h), and significantly worse on *ACC* (Fig. 1g). Therefore, the results further confirm the advantage of our nonlinear streaming decomposition and one-shot posterior updates.

6.2 Prediction Accuracy On the Fly

Next, we evaluated the dynamic performance of the streaming decomposition. To do so, on each dataset, we randomly generated a stream of training batches, upon which we ran each streaming decomposition approach and tested the prediction accuracy after each batch was processed. We set the batch size to 256 and tested with the rank $r = 3$ and $r = 8$. The running MSE/AUC of each method is shown in Fig. 2. Note that in Fig. 2d and h, we did not show the results of SVB/SSGP-NTD and SVB-NTD respectively, because their performance is much worse than all the other methods. In general, all the methods improved their prediction accuracy with more and more batches, showing increasingly better estimations of the embeddings. However, our methods (SNBTD and SNBTD-1) always obtained the best AUC/MSE throughout the running, except at the very beginning stage on *Anime*. SNBTD is in general better than or almost the same as SNBTD-1. In addition, we can see that the trend of our methods is much smoother than that of SVB/SSGP-NTF; so are POST that employs analytical, mean-field updates. This might be because the stochastic updates in SVB/SSGP-NTF are unstable and unreliable. Note that while the curves of SNBTD-1 and SNBTD are equally smooth on the other datasets, on *ACC* (Fig. 2c and g), the curve of SNBTD-1 vibrates much more and the trend of SNBTD is still quite smooth. It implies that our one-shot update can be more robust/stable than the one by one fashion in streaming inference.

6.3 Running Speed

Finally, we examined the efficiency of our methods in terms of running speed. To this end, on each dataset, we varied the size of streaming batches from $\{2^6, 2^7, 2^8, 2^9\}$ and reported the running time of each method to finish the streaming decomposition. We tested with the rank from $\{3, 8\}$. As shown in Fig. 3, SNBTD nearly always spent the least time, except that on *DBLP*, SNBTD took a little bit more than POST. Note that the mean-field updates of POST for multilinear decomposition are much simpler. We can see that due to the iteration-free nature, the running time (or the speed) of our methods is relatively constant to the batch size. By contrast, the competing approaches require iterative updates toward convergence and their speeds can vary much. For instance, on *DBLP*, *Anime* and *ACC*, POST converges fast with each rank and batch size setting, but on *MovieLen1M*, POST converges much slower in larger batches when $r = 3$ (see Fig. 3d). The running time of SSGP/SVB-NTD decreases with larger streaming batches. The reason is that smaller batches need more streaming updates and incurs more frequent execution of the computation graphs in TensorFlow. The loading and running of the graph dominated the cost while the kernel/matrix computation on different sizes of small batches turned out to make little difference. Lastly, while often achieving close predictive performance to SNBTD, SNBTD-1 is much slower. Overall, SNBTD gains 3-8x speed up as compared with SNBTD-1. Therefore, it demonstrates that our one-shot update not only improves the quality of the streaming posterior inference, but also saves the computation (*i.e.,* moment matching) and greatly accelerates the decomposition.

7 Conclusion

We have presented SNBTD, a streaming nonlinear Bayesian tensor decomposition method. Based on a spectrum GP decomposition model, SNBTD can integrate all the entries in each streaming batch to perform one-shot, closed-form posterior updates. Experiments on real-world applications have demonstrated the advantage of SNBTD in both prediction accuracy and running efficiency.

Acknowledgement

This work was supported by the NSF IIS-1910983.

References

Acar, E., Dunlavy, D. M., Kolda, T. G., and Morup, 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.

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes. In UAI, pages 33–42. Morgan Kaufmann Publishers Inc.

Boyen, X. and Koller, D. (2013). Tractable inference for complex stochastic processes. arXiv preprint arXiv:1301.7362.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational bayes. In NIPS, pages 1727–1735.

Bui, T. D., Nguyen, C., and Turner, R. E. (2017). Streaming sparse gaussian process approximations. In Advances in Neural Information Processing Systems, pages 3299–3307.

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.

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.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Model and conditions for an"explanatory"multi-mode 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 UAI, pages 282–290. AUAI Press.

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

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 KDD, pages 316–324. ACM.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

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

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.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In UAI, pages 362–369.

Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pages 467–475. Morgan Kaufmann Publishers Inc.

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.

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.

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.

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.

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., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning, 1(1–2):1–305.

Wang, Z. and Zhe, S. (2019). Conditional expectation propagation. In UAI, page 6.

Xu, Z., Yan, F., and Qi, Y. (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.

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 AISTATS, 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 NIPS, pages 928–936.