TR 11-026

Probabilistic Tensor Factorization for Tensor Completion

Hanhuai Shan, Arindam Banerjee, and Ramesh Natarajan

October 28, 2011

Probabilistic Tensor Factorization for Tensor Completion

Hanhuai Shan* Arindam Banerjee* Ramesh Natarajan†

Abstract

Multi-way tensor datasets emerge naturally in a variety of domains, such as recommendation systems, bioinformatics, and retail data analysis. The data in these domains usually contains a large number of missing entries. Therefore, many applications in those domains aim at missing value prediction, which boils down to a tensor completion problem. While tensor factorization algorithms can be a potentially powerful approach to tensor completion, most existing methods have the following limitations: First, some tensor factorization algorithms are unsuitable for tensor completion since they cannot work with incomplete tensors. Second, deterministic tensor factorization algorithms can only generate point estimates for the missing entries, while in some cases, it is desirable to obtain multiple-imputation datasets which are more representative of the joint variability for the predicted missing values. Therefore, we propose probabilistic tensor factorization algorithms, which are naturally applicable to incomplete tensors to provide both point estimate and multiple imputation for the missing entries. In this paper, we mainly focus on the applications to retail sales datasets, but the framework and algorithms are applicable to other domains as well. Through extensive experiments on real-world retail sales data, we show that our models are competitive with state-of-the-art algorithms, both in prediction accuracy and running time.

1 Introduction

Multi-way datasets containing data measured across multiple related dimensions are increasingly important in a variety of domains, such as recommendation systems, bioinformatics, and retail systems. An important special case that is widely encountered is the three-way tensor dataset. For example, in a recommendation system, the user rates a certain product of a certain brand, which yields a "rating tensor," whose three dimensions are user, product and brand, and each entry is the rating value. In retail sales data, a product has a certain price in some store at some time, so we have a "price tensor" whose three dimensions are product, store and time, and each entry is the price. If the prices are replaced by the unit sales, we would have a similar "unit-sales" tensor. In this paper, the exposition is targeted towards applications to tensors in retail sales data, however, the general framework is applicable to a variety of other domains as well.

The retail sales datasets mentioned above, may have missing entries due to various reasons associated with process errors in the data logging, reporting or integration steps of the data collection process. However, many important decision-support applications, such as inventory optimization, product demand forecasting, strategic product pricing etc., typically require a complete dataset with no missing entries. Therefore, an important initial task in retail sales data analysis is missing value prediction or imputation, in order to generate the required complete dataset for the relevant decision-support applications. In this context, the task of missing value prediction or imputation is two-fold: The first, which is relevant in most circumstances, is to obtain a point estimate for each missing value, thereby leading to a single complete data set. This is the most common task for which matrix and tensor completions [9, 1] are used. The second is to obtain multiple imputation (MI) datasets [14], i.e., multiple separate datasets containing different realizations of the relevant missing entries, which are then used in the subsequent statistical or decision-support applications from which the individual results are appropriately combined. The objective of doing MI is to capture the natural joint variability in missing data for the subsequent statistical analysis, which would be lost by only using point estimates.

A naive way to do tensor completion is to ignore the tensor structure and just use matrix completion on the data. In particular, we can perform matrix completion either on each slice of a tensor, or on a big matrix converted from a tensor (e.g., by concatenating the slices of a tensor to form a matrix). However, this naive approach has several limitations: First, it ignores the correlation that exists along certain dimensions of the dataset. Second, the number of parameters needed increases dramatically, which not only leads to potential overfitting, but also increases time and space complexity. Therefore, we work on tensors directly for tensor completion without converting them to matrices, and in fact, our results in Section 5 clearly illustrate that the matrix-based approaches take much longer

*University of Minnesota, Twin Cities
†IBM T.J. Watson Research Center

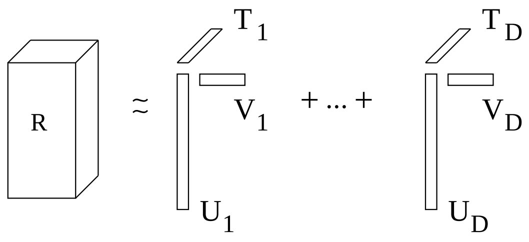


Figure 1: CP decomposition of a tensor R.

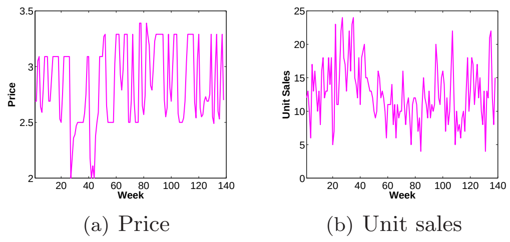


Figure 2: Price and unit sales changes of a certain product in a certain store over a period of time.

computational time than tensor based approaches, and produce poorer quality results, particularly when the fraction of missing entries is large.

Just as matrix factorization algorithms are widely used for matrix completion problems [16, 9], tensor factorizations can also be used for tensor completion. Recent years have seen considerable progresses in development of tensor factorization algorithms [8, 1, 20, 6]. However, some of these algorithms cannot work on incomplete tensors, hence are not applicable to missing value prediction [20, 6], and some can only generate point estimate for the missing entries [1]. Therefore, in this paper, we propose probabilistic tensor factorizations, which can be used for both point estimate and multiple imputation in missing value prediction. The main idea is to capture the latent structure of a tensor through a probabilistic factorization framework, and the latent structure is used for prediction. In particular, we propose two models—parametric probabilistic tensor factorization (PPTF) and Bayesian probabilistic tensor factorization (BPTF). Both of them are instances of CANDECOMP/PARAFAC (CP) tensor decomposition [8], which is a commonly used tensor model for factorization. As shown in Figure 1, CP decomposition factorizes a tensor into a summation of rank-one tensors, where U, V and T are the latent factors. In PPTF, the latent factors are assumed to be generated from Gaussian distributions, and each entry in the tensor is also generated from a Gaussian whose mean is determined by the latent factors. BPTF generalizes PPTF to a full Bayesian model. It provides the posterior distribution on missing entries, so multiple samples can be obtained from this posterior distribution for the missing entries.

In retail sales data, one dimension of the tensor is time, but plotting out the price and unit sales, as shown for a certain product over a period of time in Figure 2, we can see that the price and unit sales can change abruptly over time. Furthermore, the price is more strongly related to the corresponding price for the same product in other stores at the same time, rather than to the lagged values of the price in the same store itself. Therefore, we ignore the time dependency in our algorithm, which makes the model simple and efficient, and potentially applicable to other general tensor data which do not have a time-like dimension. In the experiment, we show that our algorithms achieve competitive performance on retail sales data compared to other algorithms in terms of prediction accuracy and running time, including a recent model explicitly capturing the temporal dependency [23].

The rest of the paper is organized as follows: Section 2 has a brief discussion on related work. In Section 3 and 4, we propose PPTF and BPTF respectively. We show experimental results in Section 5 and conclude in Section 6.

2 Related Work

We briefly review matrix and tensor factorization algorithms for missing value prediction, and provide some background for retail sales data analysis.

Matrix-factorization based algorithms have been used for matrix completion problem, such as [12] which uses regularized singular value decomposition for missing value prediction and [15] which describes a probabilistic framework for matrix factorization. Tensor factorization algorithms are usually generalizations of matrix factorization. In particular, PPTF and BPTF models proposed in this paper are generalizations of parametric probabilistic matrix factorizations (PPMF) in [19], and Bayesian probabilistic matrix factorizations (BPMF) in [16] respectively. However, PPTF and BPTF are able to capture the correlation along the additional dimension. Also, they are substantially faster than performing matrix factorization algorithms on flattened matrices converted from the tensor.

There has been some work on tensor factorization with missing data: [21] and [1] formulate the CP model as a weighted least-squares problem with a weight 1 on non-missing entries and 0 for the missing entries, so that the reduced models only minimize the tensor approximation error on the non-missing entries. [10] proposes a tensor completion algorithm based on tensor trace norm, as a generalization of matrix trace norm. These algorithms only generate point estimates for the missing entries upon reconstruction, and therefore cannot be directly used for multiple imputation. In terms of probabilistic methods, [13] proposes a multi-HDP model which combines multiple latent Dirichlet allocation [3] to represent a multi-way tensor, however, that model in spirit is closer to a clustering algorithm than a factorization algorithm. [5] proposes a method which works on tensors containing discrete ordinal values. Finally, [23] proposes a probabilistic factorization model using a Bayesian framework, which is similar to our BPTF model except that it also models the temporal behavior along the time dimension.

One simple strategy for predicting missing values in multi-dimensional retail datasets is to replace these missing entries with the corresponding mean values over the non-missing entries along one of the tensor dimensions. This strategy can be highly effective if the tensor entries are highly correlated along this dimension. However, the main disadvantage of this simple "mean imputation" strategy is that the completed tensor has a deflated variance along this dimension, and the correlation along the remaining dimensions is also ignored. A more sophisticated method that captures the variability in missing entries is "multiple imputation" [18], where multiple datasets are created, with each providing one realization of predicted missing entries. Although we are unaware of tensor-based methods for multiple imputation, [7] describes a technique based on using the correlation structure in cross-sectional time-series data for multiple imputation; in comparison, the approaches described here make use of the tensor structure along multiple dimensions simultaneously, not just the correlations in the cross-sectional dimension as in [7].

3 Parametric Probabilistic Tensor Factorization

In this section, we propose parametric probabilistic tensor factorization (PPTF), as well as a variational approximation based algorithm to learn the model.

3.1 PPTF Let R be a three-dimensional tensor of size $I \times J \times K$, where each entry is indexed as (i, j, k), and assume there is a D-dimensional latent factor $\mathbf{u}_i$, $\mathbf{v}_j$ and $\mathbf{t}_k$ corresponding to each i, j and k respectively. In other words, for each dimension of the tensor, we have a latent factor matrix $U_{I\times D}$, $V_{J\times D}$, and $T_{K\times D}$ respectively. In PPTF, the latent factors are generated from multivariate Gaussian distributions, and each entry (i, j, k) is generated from a univariate Gaussian, whose mean is determined by the corresponding latent factors on i, j, and k. The plate diagram of PPTF is given in Figure 3, and the generative process is given as follows:

1. For each i, $[i]_1^I$ ($[i]_1^I$ is defined as $i = 1 \dots I$), generate $\mathbf{u}_i \sim \mathcal{N}(\boldsymbol{\mu}_u, \Sigma_u)$.

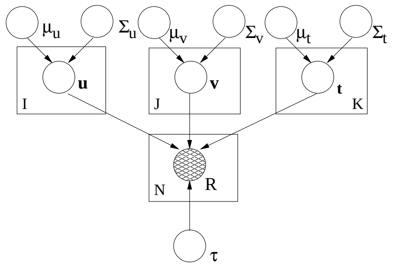


Figure 3: Graphical model for PPTF.

2. For each j, $[j]_1^J$, generate $\mathbf{v}_j \sim \mathcal{N}(\boldsymbol{\mu}_v, \Sigma_v)$.

3. For each k, $[k]_1^K$, generate $\mathbf{t}_k \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$.

4. For each of N non-missing entries (i, j, k), generate $r_{ijk} \sim \mathcal{N}(\mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{t}_k, \tau^2)$, where $\mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{t}_k = \sum_{d=1}^D u_{id} v_{jd} t_{kd}$.

Given the generative model, the likelihood function of PPTF is as follows:

(3.1)

$$p(R|\Theta) = \int\int\int \prod_{i=1}^{I} p(\mathbf{u}_i|\boldsymbol{\mu}_u, \Sigma_u) \prod_{j=1}^{J} p(\mathbf{v}_j|\boldsymbol{\mu}_v, \Sigma_v)$$

$$\prod_{k=1}^{K} p(\mathbf{t}_k|\boldsymbol{\mu}_t, \Sigma_t) \prod_{i=1}^{I}\prod_{j=1}^{J}\prod_{k=1}^{K} p(r_{ijk}|U, V, T, \tau^2)^{\delta_{ijk}} d\{U, V, T\} \ ,$$

where $\Theta = \{\boldsymbol{\mu}_u, \Sigma_u, \boldsymbol{\mu}_v, \Sigma_v, \boldsymbol{\mu}_t, \Sigma_t, \tau^2\}$ denotes the model parameters., and δ_{ijk} is an indicator which takes value 0 if r_{ijk} is missing and 1 otherwise.

3.2 Variational Inference for PPTF Given R, the learning task is to learn model parameters Θ such that $p(R|\Theta)$ in (3.1) is maximized. A general approach is to use the expectation maximization (EM) algorithm [11], where we calculate the posterior over latent variables $p(U, V, T|R, \Theta)$ in the E-step and estimate model parameters Θ in the M-step. However, the calculation of posterior for PPTF is intractable, implying that a direct application of EM is not feasible. Therefore, we propose a variational EM algorithm by introducing a fully factorized distribution $q(U, V, T|\Theta')$ as an approximation to the true posterior $p(U, V, T|R, \Theta)$. In particular,

$$q(U,V,T|\Theta') = \prod_{i=1}^{I} q(\mathbf{u}_i|\mathbf{m}_{ui}, \mathrm{diag}(\mathbf{w}_{ui}))$$

$$\prod_{j=1}^{J} p(\mathbf{v}_j|\mathbf{m}_{vj}, \mathrm{diag}(\mathbf{w}_{vj})) \prod_{k=1}^{K} p(\mathbf{t}_k|\mathbf{m}_{tk}, \mathrm{diag}(\mathbf{w}_{tk})) \ ,$$

where $\Theta' = \{\mathbf{m}_{ui}, \mathbf{m}_{vj}, \mathbf{m}_{tk}, \mathbf{w}_{ui}, \mathbf{w}_{vj}, \mathbf{w}_{tk}, [i]_1^I, [j]_1^J, [k]_1^K\}$ are variational parameters. All variational parameters are D-dimensional vectors, and $\text{diag}(\mathbf{w}_{ui})$ denotes a square matrix with $\mathbf{w}_{ui}$ on the diagonal.

Given $q(U,V,T|\Theta')$, applying Jensen's inequality [22] yields a lower bound to the original log-likelihood of R:

$$\log p(R|\Theta) \geq E_q[\log p(U,V,T,R|\Theta)] - E_q[\log q(U,V,T|\Theta')] \,.$$

Denoting the lower bound using $L(\Theta,\Theta')$, it could be expanded as:

$$L(\Theta,\Theta') = E_q\left[\log\frac{p(U|\Theta)}{q(U|\Theta')}\right] + E_q\left[\log\frac{p(V|\Theta)}{q(V|\Theta')}\right] + E_q\left[\log\frac{p(T|\Theta)}{q(T|\Theta')}\right] + E_q\left[\log p(R|\Theta,U,V,T)\right] . \tag{3.2}$$

The first term is given by

$$E_q\left[\log\frac{p(U|\Theta)}{q(U|\Theta')}\right] = -\frac{1}{2}\sum_{i=1}^{I}\Big\{\text{Tr}(\Sigma_u^{-1}\text{diag}(\mathbf{w}_{ui})) + (\mathbf{m}_{ui}-\boldsymbol{\mu}_u)^T\Sigma_u^{-1}(\mathbf{m}_{ui}-\boldsymbol{\mu}_u)\Big\} + \frac{I}{2}\log|\Sigma_u^{-1}| + \frac{ID}{2} + \frac{1}{2}\sum_{i=1}^{I}\sum_{d=1}^{D}\log w_{uid},$$

and the terms $E_q\left[\log\frac{p(V|\Theta)}{q(V|\Theta')}\right]$ and $E_q\left[\log\frac{p(T|\Theta)}{q(T|\Theta')}\right]$ have a similar form. For $E_q[\log p(R|\Theta,U,V,T)]$, we have

$$E_q[\log p(R|U,V,T,\tau^2)] = -\frac{1}{2\tau^2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\Big\{r_{ijk}^2 - 2r_{ijk}\sum_{d=1}^{D}E_q[u_{id}v_{jd}t_{kd}] + E_q\Big[(\sum_{d=1}^{D}u_{id}v_{jd}t_{kd})^2\Big]\Big\} - \frac{N}{2}\log 2\pi\tau^2 \ ,$$

where

$$E_q[u_{id}v_{jd}t_{kd}] = m_{uid}m_{vjd}m_{tkd} \ , \tag{3.3}$$

and

$$E_q\left[\left(\sum_{d=1}^{D}u_{id}v_{jd}t_{kd}\right)^2\right] = \left(\sum_{d=1}^{D}m_{uid}m_{vjd}m_{tkd}\right)^2 + \sum_{d=1}^{D}\Big(m_{uid}^2m_{vjd}^2w_{tkd} + m_{uid}^2w_{vjd}m_{tkd}^2 + w_{uid}m_{vjd}^2m_{tkd}^2 + m_{uid}^2w_{vjd}w_{tkd} + w_{uid}m_{vjd}^2w_{tkd} + w_{uid}w_{vjd}m_{tkd}^2 + w_{uid}w_{vjd}w_{tkd}\Big) \ . \tag{3.4}$$

The variational EM algorithm is in Algorithm 1. In the E-step, if we keep the model parameters Θ fixed, the best lower bound function could be found by maximizing $L(\Theta,\Theta')$ w.r.t. Θ'. In particular, we have

Algorithm 1 Variational inference for PPTF

```
Initialize the model parameters $\{\Theta^{(0)}\}$.
for g=1...$G$ do
    Variational E-step: Maximize the lower bound $L(\Theta^{(g)},\Theta')$ in (3.2) w.r.t. $\Theta'$: Iterate through the variational parameters $\{\mathbf{m}_u,\mathbf{m}_v,\mathbf{m}_t,\mathbf{w}_u,\mathbf{w}_v,\mathbf{w}_t\}$ (as in (3.5) and (3.6)) to get $\Theta'^{(g)}$.
    Variational M-step: Maximize the lower bound $L(\Theta,\Theta'^{(g)})$ in (3.2) w.r.t. $\Theta$. Update model parameters $\Theta^{(g)}$ following (3.7)-(3.9).
end for
```

$$\mathbf{m}_{ui}^* = \Big\{\Sigma_u^{-1} + \frac{1}{\tau^2}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}[\tilde{\mathbf{m}}_{jk}\tilde{\mathbf{m}}_{jk}^T + \text{diag}(\mathbf{m}_{vj}^2\circ\mathbf{w}_{tk} + \mathbf{m}_{tk}^2\circ\mathbf{w}_{vj} + \mathbf{w}_{vj}\circ\mathbf{w}_{tk})]\Big\}^{-1}\Big(\Sigma_u^{-1}\boldsymbol{\mu}_u + \frac{1}{\tau^2}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}r_{ijk}\tilde{\mathbf{m}}_{jk}\Big) \tag{3.5}$$

$$w_{uid}^* = 1/\Big(\Sigma_{u,dd}^{-1} + \frac{1}{\tau^2}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\{m_{vjd}^2m_{tkd}^2 + m_{vjd}^2w_{tkd} + w_{vjd}m_{tkd}^2 + w_{vjd}w_{tkd}\Big) \ , \tag{3.6}$$

where $\mathbf{m}_{vj}^2$ and $\mathbf{m}_{tk}^2$ are elementwise square, $\circ$ is elementwise product, $\tilde{\mathbf{m}}_{jk} = \mathbf{m}_{vj}\circ\mathbf{m}_{tk}$, and $\Sigma_{u,dd}^{-1}$ is the d^{th} element on Σ_u^{-1}'s diagonal. The update equation for other variational parameters—$\mathbf{m}_{vj}$, $\mathbf{w}_{vj}$ and $\mathbf{m}_{tk}$, $\mathbf{w}_{tk}$—are of a similar form. The variational parameters has a dependency on each other, the algorithm thus iterates through the variational parameters several times inside the E-step.

Variational parameters Θ'^* from the E-step gives the best lower bound function $L(\Theta,\Theta'^*)$. In the M-step, maximizing $L(\Theta,\Theta'^*)$ w.r.t. Θ yields the estimation of the model parameters:

$$\boldsymbol{\mu}_u^* = \frac{1}{I}\sum_{i=1}^{I}\mathbf{m}_{ui} \tag{3.7}$$

$$\Sigma_u^* = \frac{1}{I}\sum_{i=1}^{I}\Big\{\text{diag}(\mathbf{w}_{ui}) + (\mathbf{m}_{ui}-\boldsymbol{\mu}_u)(\mathbf{m}_{ui}-\boldsymbol{\mu}_u)^T\Big\} \tag{3.8}$$

$$\tau^{2*} = \frac{1}{N}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\Big\{r_{ijk}^2 - 2r_{ijk}\sum_{d=1}^{D}E_q[u_{id}v_{jd}t_{kd}] \tag{3.9}$$

$$+ E_q \left[(\sum_{d=1}^{D} u_{id} v_{jd} t_{kd})^2 \right] \Bigg\} ,$$

where $E_q[u_{id}v_{jd}t_{kd}]$ and $E_q[(\sum_d u_{id}v_{jd}t_{kd})^2]$ are given in (3.3) and (3.4). The update equation for other model parameters, $\boldsymbol{\mu}_v^*$, Σ_v^* and $\boldsymbol{\mu}_t^*$, Σ_t^*, are of a similar form.

Overall, the algorithm iterates through the E-step and M-step. After g iterations, the lower bound function becomes $L(\Theta^{(g)}, \Theta'^{(g)})$. We have

$$L(\Theta^{(g)}, \Theta'^{(g)}) \leq L(\Theta^{(g)}, \Theta'^{(g+1)}) \leq L(\Theta^{(g+1)}, \Theta'^{(g+1)}) .$$

The first inequality holds because in the E-step, $\Theta'^{(g+1)}$ maximizes $L(\Theta^{(g)}, \Theta')$, and the second inequality holds because in the M-step $\Theta^{(g+1)}$ maximizes $L(\Theta, \Theta'^{(g+1)})$. Therefore, the lower bound function $L(\Theta, \Theta')$ keeps on increasing during the iteration. In Algorithm 1, we set a fixed number of iterations G, but in practice, we can stop the iteration when the log-likelihood stops changing or we can use an early stopping strategy which is given in the experiment section.

For time complexity, in each variational EM step, the time takes by each iteration of E-step is $O(D(JK + IK + IJ) + D^2N + D^3(I + J + K))$, and the time takes by the M-step is $O(D^2(I + J + K) + DN)$. Therefore, the E-step dominates the computation. Usually, when D is small, the size of the tensor is large, and the tensor is not extremely sparse, the time complexity of each iteration in variational E-step is roughly $O(D^2N)$, which is linear in terms of the total number of non-missing entries N.

3.3 Prediction We use a MAP estimate for prediction. In particular, to predict the entry (i, j, k), we use

$$\hat{r}_{ijk} = \hat{\mathbf{u}}_i \cdot \hat{\mathbf{v}}_j \cdot \hat{\mathbf{t}}_k = \sum_{d=1}^{D} \hat{u}_{id} \hat{v}_{jd} \hat{t}_{kd} ,$$

where

$$\begin{aligned} \{\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_j, \hat{\mathbf{t}}_k\} &= \operatorname*{argmax}_{\mathbf{u}_i,\mathbf{v}_j,\mathbf{t}_k} p(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k | R, \Theta) \\ &\approx \operatorname*{argmax}_{\mathbf{u}_i,\mathbf{v}_j,\mathbf{t}_k} q(\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k | \Theta') \\ &= \{\mathbf{m}_{ui}, \mathbf{m}_{vj}, \mathbf{m}_{tk}\} . \end{aligned}$$

4 Bayesian Probabilistic Tensor Factorization

In this section, we propose Bayesian probabilistic tensor factorization (BPTF). In PPTF, the model parameters are learnt from the data through a maximum likelihood estimate strategy. In BPTF, instead of picking one best set of model parameters, it maintains a distribution over all possible parameters by putting a prior on top. Therefore, it effectively gets the benefit from averaging all possible models.

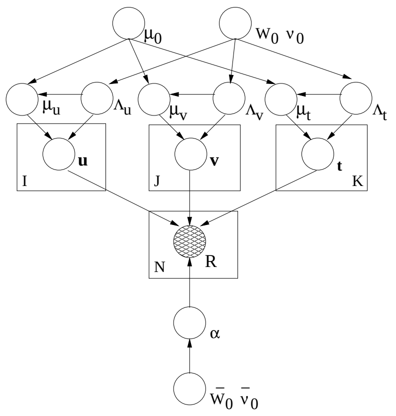


Figure 4: Graphical model for BPTF.

4.1 BPTF BPTF puts Gaussian-Wishart priors on top of the model parameters of PPTF. It is hence a full Bayesian extension of PPTF. The plate diagram of BPTF is given in Figure 4, and the generative process of BPTF is as follows:

1. Generate $\Lambda_u, \Lambda_v, \Lambda_t \sim \mathcal{W}(W_0, \nu_0)$. $\mathcal{W}(W_0, \nu_0)$ is the Wishart distribution with ν_0 degrees of freedom and a $D \times D$ scale matrix W_0. In particular,

   $$\mathcal{W}(\Lambda | W_0, \nu_0) = \frac{1}{C} |\Lambda|^{(\nu_0 - D - 1)} \exp(-\frac{1}{2} \text{Tr}(W_0^{-1} \Lambda)) ,$$

   where C is the constant for normalization.

2. Generate $\boldsymbol{\mu}_u \sim \mathcal{N}(\boldsymbol{\mu}_0, c_0\Lambda_u^{-1})$, $\boldsymbol{\mu}_v \sim \mathcal{N}(\boldsymbol{\mu}_0, c_0\Lambda_v^{-1})$, $\boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\mu}_0, c_0\Lambda_t^{-1})$, where Λ_u, Λ_v and Λ_t are used as the precision matrices for Gaussians.

3. Generate $\alpha \sim \mathcal{W}(\bar{W}_0, \bar{\nu}_0)$.

4. For each i, $[i]_1^I$, generate $\mathbf{u}_i \sim \mathcal{N}(\boldsymbol{\mu}_u, \Lambda_u^{-1})$.

5. For each j, $[j]_1^J$, generate $\mathbf{v}_j \sim \mathcal{N}(\boldsymbol{\mu}_v, \Lambda_v^{-1})$.

6. For each k, $[k]_1^K$, generate $\mathbf{t}_k \sim \mathcal{N}(\boldsymbol{\mu}_t, \Lambda_t^{-1})$.

7. For each non-missing entry (i, j, k), generate $r_{ijk} \sim \mathcal{N}(\mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{t}_k, \alpha^{-1})$.

4.2 Inference and Prediction From BPTF, the distribution of an unknown entry r_{ijk} given the observable tensor R is as follows:

(4.10)
$$p(\hat{r}_{ijk}|R, \Theta_0) = \int p(\hat{r}_{ijk}|\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k, \alpha)$$
$$p(U, V, T, \Theta_u, \Theta_v, \Theta_t, \alpha | R, \Theta_0) d\{U, V, T, \Theta_u, \Theta_v, \Theta_t, \alpha\},$$

where $\Theta_u = \{\boldsymbol{\mu}_u, \Lambda_u\}$, $\Theta_v = \{\boldsymbol{\mu}_v, \Lambda_v\}$, $\Theta_t = \{\boldsymbol{\mu}_t, \Lambda_t\}$, and $\Theta_0 = \{\boldsymbol{\mu}_0, W_0, \nu_0, \bar{W}_0, \bar{\nu}_0\}$.

Algorithm 2 Gibbs sampling for BPTF

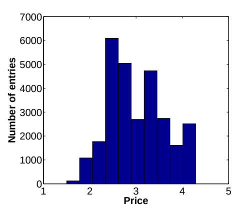
```
Initialize the latent factors $\{U^{(1)}, V^{(1)}, T^{(1)}\}$.
for $g = 1 \dots G$ do
    Sample $\Theta_u$, $\Theta_v$, $\Theta_t$ according to (4.12) and similar equations on $V$ and $T$.
        $\Theta_u^{(g)} \sim p(\Theta_u | U^{(g)}, \Theta_0)$.
        $\Theta_v^{(g)} \sim p(\Theta_v | V^{(g)}, \Theta_0)$.
        $\Theta_t^{(g)} \sim p(\Theta_t | T^{(g)}, \Theta_0)$.
    Sample $\alpha$ according to (4.13).
        $\alpha^{(g)} \sim p(\alpha | R, U^{(g)}, V^{(g)}, T^{(g)}, \bar{W}_0, \bar{\nu}_0)$.
    for $i = 1 \dots I$ do
        Sample latent factors $\mathbf{u}_i$ according to (4.14).
            $\mathbf{u}_i^{(g+1)} \sim p(\mathbf{u}_i | R, V^{(g)}, T^{(g)}, \Theta_u^{(g)}, \alpha^{(g)})$
    end for
    for $j = 1 \dots J$ do
        Sample latent factors $\mathbf{v}_j$
            $\mathbf{v}_j^{(g+1)} \sim p(\mathbf{v}_j | R, U^{(g+1)}, T^{(g)}, \Theta_v^{(g)}, \alpha^{(g)})$
    end for
    for $k = 1 \dots K$ do
        Sample latent factors $\mathbf{t}_k$
            $\mathbf{t}_k^{(g+1)} \sim p(\mathbf{t}_k | R, U^{(g+1)}, V^{(g+1)}, \Theta_t^{(g)}, \alpha^{(g)})$
    end for
end for
```

$p(\hat{r}_{ijk}|R, \Theta_0)$ in (4.10) could be considered as an expectation of $p(\hat{r}_{ijk}|\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k, \alpha)$ over the posterior distribution $p(U, V, T, \Theta_u, \Theta_v, \Theta_t, \alpha|R, \Theta_0)$. Since the calculation of (4.10) is intractable, we use Markov chain Monte Carlo methods (MCMC) [2] to do inference. The MCMC method draws from some proposal distribution a sequence of samples, which form a Markov chain. After a large number of steps, the chain converges to the target distribution. We can collect a number of samples and use their empirical mean to approximate the expectation in (4.10), i.e.,

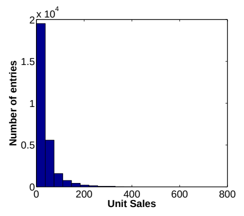
(4.11)
$$p(\hat{r}_{ijk}|R, \Theta_0) \approx \frac{1}{G} \sum_{g=1}^{G} p(\hat{r}_{ijk}|\mathbf{u}_i^{(g)}, \mathbf{v}_j^{(g)}, \mathbf{t}_k^{(g)}, \alpha^{(g)}) ,$$

where G is the total number of samples we draw for approximation, and $\mathbf{u}_i^{(g)}$, $\mathbf{v}_j^{(g)}$, $\mathbf{t}_k^{(g)}$ and $\alpha^{(g)}$ are from the g^{th} sample.

Given (4.11), the inference and prediction for BPTF boils down to drawing samples of $\mathbf{u}_i^{(g)}$, $\mathbf{v}_j^{(g)}$, $\mathbf{t}_k^{(g)}$ and $\alpha^{(g)}$. In this paper, we use Gibbs sampling [4] to draw such samples, as shown in Algorithm 2. In particular, we iterate through the model parameters and latent variables, by drawing each random variable conditioned on the current values of the rest of them. The total number of samples we draw in each iteration



(a) On Price



(b) On Unit sales

Figure 5: The histogram of the entry values in price and unit-sales tensor.

is $O(D^3(I + J + K) + D^2N)$, which usually reduces to $O(D^2N)$.

In Algorithm 2, the conditional distribution for Θ_u where $\Theta_u^{(g)}$ is sampled from is given by Gaussian-Wishart distribution as follows:

(4.12)
$$p(\boldsymbol{\mu}_u, \Lambda_u|U, \Theta_0) = N(\boldsymbol{\mu}_u|\boldsymbol{\mu}_0^*, (c_0^*\Lambda_u)^{-1})\mathcal{W}(\Lambda_u|W_0^*, \nu_0^*) ,$$

where

$$\begin{aligned}
\boldsymbol{\mu}_0^* &= \frac{c_0\boldsymbol{\mu}_0 + I\bar{\mathbf{u}}}{c_0 + I}, \quad c_0^* = c_0 + I, \quad \nu_0^* = \nu_0 + I \\
W_0^* &= \left(W_0^{-1} + S + \frac{c_0 I}{c_0 + I}(\bar{\mathbf{u}} - \boldsymbol{\mu}_0)(\bar{\mathbf{u}} - \boldsymbol{\mu}_0)^T)\right)^{-1} \\
S &= \sum_{i=1}^{I}(\mathbf{u}_i - \bar{\mathbf{u}})(\mathbf{u}_i - \bar{\mathbf{u}})^T, \quad \bar{\mathbf{u}} = \frac{1}{I}\sum_{i=1}^{I}\mathbf{u}_i \ .
\end{aligned}$$

The conditional distributions on Θ_v and Θ_t are of a similar form.

The conditional distribution to sample $\alpha^{(g)}$ is given by

$$p(\alpha|R, U, V, T, \bar{W}_0, \bar{\nu}_0) = \mathcal{W}(\alpha|\bar{W}_0^*, \bar{\nu}_0^*) , \tag{4.13}$$

with

$$\begin{aligned}
\bar{\nu}_0^* &= \bar{\nu}_0 + N \\
(\bar{W}_0^*)^{-1} &= \bar{W}_0^{-1} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}(r_{ijk} - \mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{t}_k)^2 \ .
\end{aligned}$$

The conditional distribution for $\mathbf{u}_i$ is given by

$$p(\mathbf{u}_i|R, V, T, \boldsymbol{\mu}_u, \Lambda_u, \alpha) = \mathcal{N}(\boldsymbol{\mu}_i^*, [\Lambda_i^*]^{-1}) , \tag{4.14}$$

with

$$\begin{aligned}
\Lambda_i^* &= \Lambda_u + \alpha\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}\mathbf{q}_{jk}\mathbf{q}_{jk}^T \\
\boldsymbol{\mu}_i^* &= (\Lambda_i^*)^{-1}\left(\alpha\sum_{j=1}^{J}\sum_{k=1}^{K}\delta_{ijk}r_{ijk}\mathbf{q}_{jk} + \Lambda_u\boldsymbol{\mu}_u\right) ,
\end{aligned}$$

where $\mathbf{q}_{jk} = \mathbf{v}_j \circ \mathbf{t}_k$. The conditional distribution on the latent matrix U factorizes over all $p(\mathbf{u}_i|R, V, T, \boldsymbol{\mu}_u, \Lambda_u, \alpha)$:

$$p(U|R, V, T, \boldsymbol{\mu}_u, \Lambda_u, \alpha) = \prod_{i=1}^{I} p(\mathbf{u}_i|R, V, T, \boldsymbol{\mu}_u, \Lambda_u, \alpha) \ .$$

The conditional distribution on V and T are of a similar form.

We sample $\{\Theta_u, \Theta_v, \Theta_t\}$ and $\{U, V, T, \alpha\}$ following Algorithm 2. After we have collected enough samples, we get the estimation for any missing entry (i, j, k) based on (4.11). In addition, if we do not take the average over all samples of $\hat{r}_{ijk}$ as in (4.11), we could keep multiple samples which leads to multiple imputation. We illustrate this point in the experiment (Section 5.3).

5 Experimental Results

In this section, we present experimental result on synthetic and real retail sales datasets. We compare PPTF and BPTF on point estimate with other tensor factorization algorithms, as well as matrix factorization algorithms. In addition, we also describe some results obtained for multiple imputation, although a detailed analysis of this will be presented elsewhere.

5.1 Dataset We use both synthetic data and real retail sales data for experiments. For synthetic data, we first generate a tensor $X_{I\times J\times K}$ given the latent factor matrices U, V, and T. In particular, $x_{ijk} = \mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{t}_k$. We then generate another tensor $E_{I\times J\times K}$ from a standard Gaussian distribution. E is of the same size as X and it is used to add noise to X following [1]:

$$R = X + (100/\eta - 1)^{-1/2} \frac{\| X \|}{\| E \|} E \ , \tag{5.15}$$

where η is a parameter to control the noise level. A larger η indicates a larger noise.

We use two patterns to hold out missing data: randomly missing entries and randomly missing fibers. For randomly missing entries, we randomly hold out p of all entries in the tensor. For randomly missing fibers, among totally $I \times J$ fibers (A fiber in a tensor is obtain by fixing every index but one [8].), we randomly hold out p of them as missing data. In retail sales data, missing fibers corresponds to the case when data from a whole store at a certain time are not collected, which is common in reality.

The real retail sales data contain price and unit-sales records for 19 products in 10 stores of each week during a three-year period (August 2006 to August 2009). Therefore, this dataset contains two tensors for price and unit-sales respectively, whose three dimensions corresponding to store, product, and time. The dataset contains some missing entries, which are in the same location of the price and unit-sales tensors, i.e., these entries are missing together, if at all, for a given product, store, time combination. While a real-world task would require predicting these missing data values, for evaluation purposes, we use the non-missing entries in these tensors as training and test partitions. In summary, the tensor datasets are of size $19 \times 10 \times 156$, and contain 28406 non-mssing entries. Figure 5 shows histograms of all entry values in the price and unit-sales tensors. The price appears to be normally distributed with values in the range of 1.5 to 4.5, while the distribution of unit sales is highly skewed with values in a wide range between 0 and 800. In this regard, we expect that estimation of the missing values for the unit-sales tensor may be more challenging than that for the price tensor.

5.2 Point Estimate Result In point estimate experiment, we first run PPTF and BPTF on synthetic data. We then compare them with other matrix and tensor factorization methods on real retail sales data.

For evaluation, we use root mean square error (RMSE), which is defined as

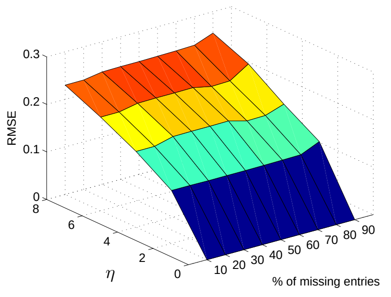
$$RMSE = \sqrt{\frac{\sum_{n=1}^{\tilde{N}} (x_n - \hat{x}_n)^2}{\tilde{N}}} \ ,$$

where x_n and $\hat{x}_n$ are the real value and predicted value for the n^{th} entry respectively, with n running over all $\tilde{N}$ entries in the tensor to be predicted.

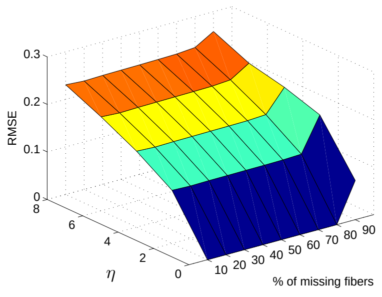
For PPTF, we observed that after a certain number of iterations, its performance decreases on the test set, implying the overfitting, so we hold out a validation set, and stop the algorithm if the prediction accuracy decreases on the validation set. For BPTF, such "early stopping" strategy does not give good results, so we let it run for 200 iterations, where empirically we observe that the results become stable when we keep on sampling.

5.2.1 Results on synthetic data On synthetic data, we run PPTF and BPTF to show how RMSE changes with increasing missing entries and noises. In particular, on a $50 \times 50 \times 50$ tensor, we increase the missing percentage p from 10% to 90%, in both missing entries and missing fibers cases. We also increase the noise by setting the parameter η in (5.15) from 0 to 8. The result for PPTF and BPTF are presented in Figure 6 and 7 respectively, where we set the latent dimension $D = 5$. The main observations are as follows:

1. For both PPTF and BPTF, RMSE decreases mono-

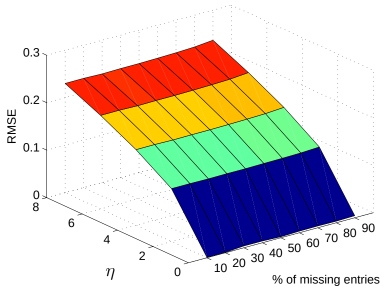


(a) Missing entries

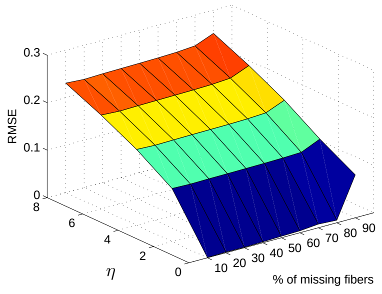


(b) Missing fibers

Figure 6: RMSE of PPTF on synthetic data of missing entries and missing fibers.



(a) Missing entries



(b) Missing fibers

Figure 7: RMSE of BPTF on synthetic data of missing entries and missing fibers.

tonically when the noise parameter η gets smaller. When $\eta = 0$, the RMSE is almost 0, meaning that the algorithms are able to recover the original tensor almost perfectly.

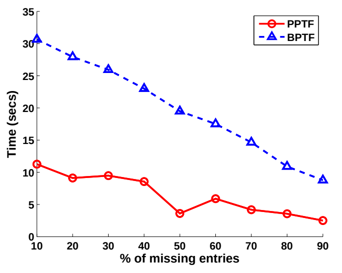
2. The increasing percentage of missing data does not affect the result much. Even with up to 80% of data missing, the RMSE is still close to the result with 10% of data missing. In particular, RMSE on 80% of missing data is almost 0 when $\eta = 0$. According to [1], the reason is that there are still much more data than variables even with a large percentage of missing data. In particular, the total number of variables in latent factor matrices is $D(I + J + K)$, and the total number of non-missing entries in the tensor is $(1 - p)IJK$, which is usually greater than $D(I + J + K)$. A clear deterioration in RMSE is only observed when 90% of fibers are missing.

We also show how the running time changes with percentage of missing data p, latent factor dimensions D, and the size of tensor, in the missing entry case. The results are shown in Figure 8. In (a), we increase p from 10% to 90%. The running time decreases when p goes up, indicating that both PPTF and BPTF can take advantage of the sparsity in the tensor. In (b), the running time increases with D, meaning that the model runs slower when its complexity increases. From (c), we see that the running time has a high order relationship with the size of the tensor, so it increases fast when the size increases. Overall, in all three cases, PPTF runs faster than BPTF.

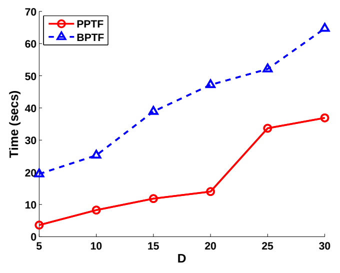
5.2.2 Comparison with matrix factorizations

We compare PPTF and BPTF with other matrix and tensor factorization algorithms on real retail sales data. For easy reference, we give the acronyms of the algorithms we compared to in Table 1. We run 10-fold cross validation on the available entries. In particular, we divide all non-missing entries evenly into 10 parts. Each part is used as the test set in turn, and the rest are used as the training set, i.e., 90% data for training. In order to see how the algorithms behave on sparse data, we then run 10-fold cross validation with the training set and test set swapped, i.e., 10% data for training.

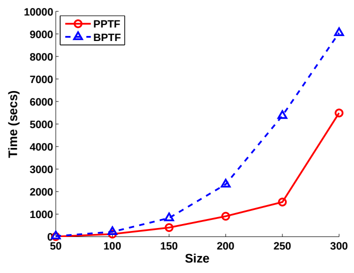
In this subsection, we present the result on comparison with Bayesian probabilistic matrix factorization (BPMF) [17]. BPMF is the state-of-the-art algorithm used for missing value prediction in matrix. We try two ways to run matrix factorization on a tensor: one is to run on each slice [8] of tensor, and the other is to run



(a) Increasing p



(b) Increasing D



(c) Increasing size

Figure 8: The running time on synthetic data with increasing missing data, increasing latent dimension, and increasing tensor size ($I = J = K$). (a) D=5 and I=J=K=50. (b) p=50% and I=J=K=50. (c) p=50% and D=5. The noise parameter $\eta = 2$.

Table 1: Acronyms for the compared algorithms.

BPMF1	Split the tensor along the first dimension (store). Perform BPMF [17] on each slice.
BPMF2	Split the tensor along the second dimension (product). Perform BPMF on each slice.
BPMF3	Split the tensor along the third dimension (time). Perform BPMF on each slice.
BPMFC	Split the tensor along the first dimension (store). Concatenate the slices to form a big matrix. Perform BPMF on the matrix.
WCP	Weighted CANDECOMP/PARAFAC [1].
BPTFT	Bayesian tensor factorization capturing temporal information [23].

on the concatenate slices as a big matrix. In the first method, we take the slices along the first (store), second (product) and third (time) dimension of the tensor and run BPMF, which we refer to as BPMF1, BPMF2 and BPMF3 respectively. For the second method, there are totally 6 ways to get a large matrix with concatenate slices of the tensor, we only pick one of them and refer to it as BPMFC. The results of RMSE using 90% and 10% of training data are presented in Table 2 and 3 respectively, where we increase the number of latent dimensions D from 5 to 30 in steps of 5. In each table, the top part shows the result from PPTF and BPTF, and the middle part shows the result from BPMF.

On price prediction, PPTF and BPTF perform much better than BPMF with a suitable choice of D. The best result (among using different D) from PPTF and BPTF is apparently much better than the best result from BPMF. The advantage of PPTF and BPTF is more distinct when only 10% of data are used for training. On unit sales prediction, PPTF and BPTF also perform better when only 10% of training data are used. The only case where BPMF has competitive performance is on unit-sales prediction with 90% of training data, where BPMF2 mostly performs better but the performance of other BPMF algorithms is still not good. The possible reason is as follows: The correlation (price or unit sales) among different products is not very strong, compared to the correlation of a same product in different stores or a same product over different time, so breaking the weaker correlation still yields good results. In terms of running time, although we do not show the numbers in the paper, BPMF is at least an order of magnitude slower than PPTF and BPTF. Taking BPMF1 for example, the number of its latent factors to infer is $I(J + K)$, which is usually much larger than $I + J + K$ in tensor factorization. Therefore, it is much slower than tensor factorization algorithms.

5.2.3 Comparison with tensor factorizations

We also compare our algorithms with two other tensor factorization methods. The first one is a weighted least square formulation of CP decomposition [1] (WCP), where the approximation error is taken only on the non-missing entries. The second one is similar with BPTF except that it captures the temporal dependencies [23], which we refer to as BPTFT. We again run 10-fold cross validation and the RMSE of WCP and BPTFT are shown at the bottom of Table 2 and 3.

As a full Bayesian model, BPTF performs better than PPTF in most of the cases. Also, from Table 2 and 3, we can see the RMSE of BPTF decreases almost monotonically when the latent factor dimension D increases. It means that when the model becomes more complicated, it achieves better performance. On unit-sales prediction, there is a slight increase of RMSE when D reaches 30, which might be an indication of overfitting.

PPTF and BPTF outperform WCP significantly in all cases. Between BPTF and BPTFT, when 90% of training data are used, they have very similar accuracy on price prediction, and BPTFT performs better on unit-sales prediction; when 10% of training data are used, BPTFT outperforms BPTF on price prediction, and BPTF outperforms BPTFT on unit-sales prediction, which is the most difficult task among four due to the high variance of unit-sales data and the sparsity

Table 2: RMSE on retail sales data with 90% training data. Top 3 results for each choice of D are bolded. On price data, BPTF and PPTF are much better than BPMF and WCP, and competitive with BPTFT. On unit-sales data, BPTF and PPTF are better than WCP, but BPTFT and BPMF2 perform better than BPTF and PPTF.

(a) On price

D	5	10	15	20	25	30
PPTF	0.284 ± 0.005	**0.170** ± **0.005**	**0.089** ± **0.006**	**0.055** ± **0.005**	**0.053** ± **0.004**	**0.053** ± **0.004**
BPTF	0.284 ± 0.004	**0.169** ± **0.004**	**0.087** ± **0.004**	**0.054** ± **0.004**	**0.054** ± **0.004**	**0.053** ± **0.003**
BPMF1	0.430 ± 0.004	0.428 ± 0.004	0.426 ± 0.004	0.425 ± 0.004	0.425 ± 0.004	0.424 ± 0.004
BPMF2	**0.181** ± **0.002**	0.182 ± 0.001	0.183 ± 0.001	0.183 ± 0.001	0.182 ± 0.001	0.181 ± 0.002
BPMF3	**0.180** ± **0.002**	0.188 ± 0.002	0.183 ± 0.002	0.183 ± 0.002	0.175 ± 0.003	0.186 ± 0.003
BPMFC	**0.1334** ±**0.0021**	**0.1323** ±**0.0020**	0.1325 ±0.0018	0.1323 ±0.0019	0.1330 ±0.0021	0.1330 ±0.0021
WCP	0.349 ± 0.029	0.316 ± 0.044	0.260 ± 0.041	0.246 ± 0.033	0.191 ± 0.028	0.203 ± 0.044
BPTFT	0.285 ± 0.005	0.171 ± 0.004	**0.087** ± **0.002**	**0.052** ± **0.004**	**0.053** ± **0.003**	**0.053** ± **0.004**

(b) On unit sales

D	5	10	15	20	25	30
PPTF	**15.867** ± **0.572**	13.988 ± 0.393	13.053 ± 0.588	12.789 ± 0.573	12.594 ± 0.598	**12.239** ± **0.502**
BPTF	**15.837** ± **0.540**	**13.696** ± **0.493**	**12.836** ± **0.677**	**12.365** ± **0.546**	**12.209** ± **0.725**	12.470 ± 1.360
BPMF1	316.884 ±171.038	259.632 ±73.202	63.425 ±10.288	27.035 ± 1.216	25.218 ± 1.267	25.187 ± 0.960
BPMF2	193.094 ±151.512	**12.170** ± **0.676**	**12.100** ± **0.639**	**12.159** ± **0.611**	**12.094** ± **0.595**	**12.286** ± **0.620**
BPMF3	235.870 ±157.520	29.555 ± 1.421	29.617 ± 1.888	28.736 ± 1.500	29.755 ± 1.309	28.876 ± 1.908
BPMFC	63.259 ±21.721	14.661 ±0.712	14.648 ±0.711	14.655 ±0.728	14.655 ±0.711	14.663 ±0.715
WCP	29.422 ± 5.528	21.628 ± 1.109	21.617 ± 1.978	21.070 ± 1.336	20.947 ± 1.920	20.676 ± 1.806
BPTFT	**15.302** ± **0.520**	**13.054** ± **0.394**	**12.197** ± **0.516**	**11.770** ± **0.460**	**11.588** ± **0.502**	**11.611** ± **0.625**

Table 3: RMSE on retail sales data with 10% training data. Top 3 results for each choice of D are bolded. BPTF outperforms BPMF and WCP. On price data, BPTFT performs better than BPTF and PPTF. On unit-sales data, BPTF perform better than BPTFT.

(a) On price

D	5	10	15	20	25	30
PPTF	**0.369** ± **0.009**	**0.360** ± **0.015**	**0.364** ± **0.010**	**0.363** ± **0.010**	**0.371** ± **0.011**	**0.374** ± **0.007**
BPTF	**0.352** ± **0.007**	**0.355** ± **0.015**	**0.313** ± **0.011**	**0.295** ± **0.010**	**0.291** ± **0.008**	**0.285** ± **0.008**
BPMF1	0.476 ± 0.003	0.476 ± 0.003	0.476 ± 0.003	0.474 ± 0.003	0.472 ± 0.003	0.476 ± 0.003
BPMF2	0.425 ± 0.004	0.419 ± 0.004	0.418 ± 0.004	0.416 ± 0.004	0.414 ± 0.004	0.412 ± 0.004
BPMF3	0.522 ± 0.006	0.519 ± 0.006	0.509 ± 0.007	0.512 ± 0.005	0.513 ± 0.005	0.527 ± 0.006
BPMFC	0.568 ±0.006	0.571 ±0.006	0.572 ±0.006	0.571 ±0.005	0.571 ±0.006	0.572 ±0.006
WCP	0.510 ± 0.043	0.566 ± 0.052	0.615 ± 0.057	0.661 ± 0.061	0.655 ± 0.041	0.683 ± 0.045
BPTFT	**0.347** ± **0.006**	**0.348** ± **0.016**	**0.263** ± **0.011**	**0.235** ± **0.009**	**0.235** ± **0.012**	**0.230** ± **0.013**

(b) On unit sales

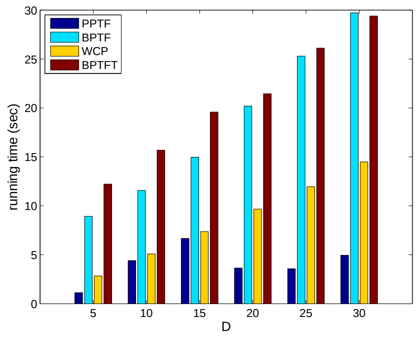
D	5	10	15	20	25	30
PPTF	**26.292** ± **1.215**	25.735 ± 1.579	24.337 ± 0.887	23.802 ± 0.710	**23.510** ± **0.770**	**23.230** ± **1.037**
BPTF	**26.755** ± **2.372**	**23.240** ± **1.041**	**23.257** ± **0.824**	**23.041** ± **0.604**	**22.769** ± **0.905**	**22.969** ± **0.616**
BPMF1	32.763 ± 1.212	33.228 ± 1.537	30.885 ± 0.983	30.422 ± 0.884	30.448 ± 0.907	29.676 ± 0.936
BPMF2	27.527 ± 2.508	**23.518** ± **1.738**	**23.326** ± **1.714**	**23.156** ± **0.9676**	**21.035** ± **0.847**	**23.423** ± **1.729**
BPMF3	41.668 ± 1.552	42.891 ± 2.493	39.489 ± 1.648	39.884 ± 2.254	40.806 ± 2.175	39.624 ± 1.986
BPMFC	37.962 ±2.591	36.341 ±1.407	37.457 ±1.423	36.815 ±0.988	36.839 ±1.471	37.586 ±2.138
WCP	44.321 ± 6.683	43.852 ± 3.519	42.641 ± 4.132	44.352 ± 4.010	45.898 ± 2.545	45.463 ± 2.043
BPTFT	**25.761** ± **5.568**	**23.437** ± **1.586**	**23.659** ± **0.649**	**23.596** ± **1.187**	23.633 ± 0.962	24.126 ± 0.944

of the training set. Overall, there is no clear winner between BPTF and BPTFT. Recall that BPTF and BPTFT are very similar models, except that BPTFT captures temporal correlation while BPTF does not. The RMSE results show that in retail data, incorporating temporal dependency does not necessarily yields better results.

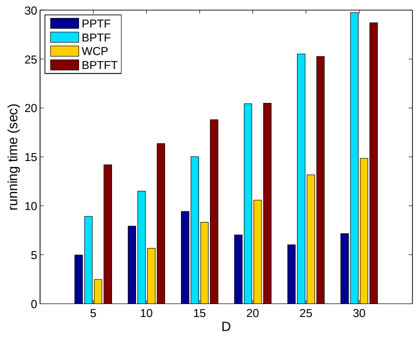
In terms of running time, the results for four algorithms are presented in Figure 9 and 10. BPTF is slower than WCP, but in general slightly faster than BPTFT since it is inherently a simpler model. For PPTF, it is the fastest algorithm among four in most of the cases, showing its high efficiency.

5.3 Result on multiple imputation In BPTF, instead of giving one point estimate by averaging over all samples, we can also keep multiple samples, which are much richer information than point estimate alone, since the variance of multiple samples tells us the model's confidence on a particular prediction. If the variance is small, it means that all samples are in a small range of values, indicating that the model is quite confident about the prediction. If the variance is large, the samples spreads over a wide range of values, indicating low confidence of the model.

In this experiment, we show the relationship between BPTF's confidence and prediction accuracy. The key question is: Are the predictions with high confidence more accurate than those with low confidence? Given multiple samples of some test entries, we can sort the test entries based on the standard deviation of the samples for each entry. We then divided the test entries into five parts. The first part contains the first 20% of entries with the lowest standard deviation on samples, the second part contains another 20% of entries with the second lowest standard deviations on samples, etc.. We

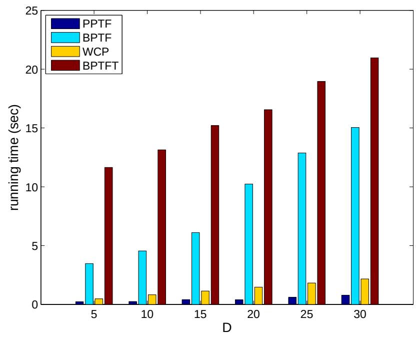


(a) On Price

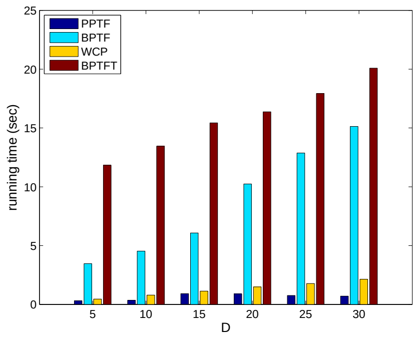


(b) On Unit sales

Figure 9: Running time of four tensor factorization algorithms on retail sales data with 90% of training data. PPTF is the fastest algorithm mostly. BPTF is slower than WCP, but slightly faster than BPTFT in most of the cases. We do not show the result for BPMF since it is at least an order of magnitude slower than tensor factorization algorithms (Best viewed in color).



(a) On Price



(b) On Unit sales

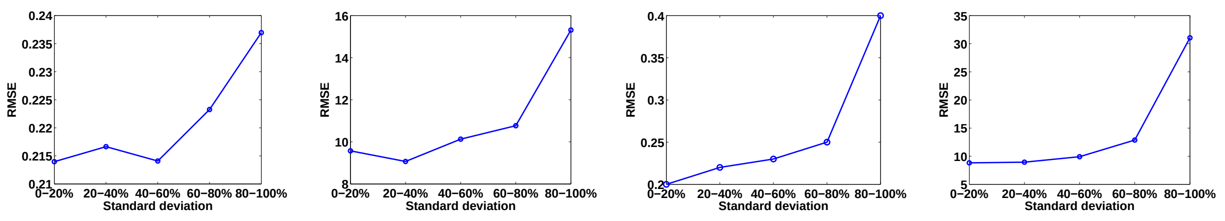
Figure 10: Running time of different algorithms on retail sales data with 10% training data. PPTF is the fastest among four algorithms. BPTF is slower than WCP, but faster than BPTFT. We do not show the result for BPMF since it is at least an order of magnitude slower than tensor factorization algorithms (Best viewed in color).

then compute RMSE on each part and Figure 11 shows how the RMSE increases with standard deviation. Interestingly, we found that when the samples' standard deviation increases, i.e., when the model's confidence decreases, the model's accuracy also decreases, almost monotonically. It is a nice and interesting property of BPTF, since it gives us a sense of its accuracy by just looking at its confidence, i.e., sample variance. It also shows the value of the multiple imputation, which can not be done by deterministic algorithms such as WCP.

6 Conclusion

In this paper, we have proposed methods for missing value prediction and imputation in tensors through tensor factorization. In particular, we propose parametric probabilistic tensor factorization (PPTF) and its full Bayesian extension—Bayesian probabilistic tensor factorization (BPTF), both of which are instances of CP tensor decompositions. The experiments on real-world retail sales datasets show that these methods outperform matrix-factorization based algorithms, and are competitive with other state-of-the-art tensor-factorization algorithms in terms of predictive performance and computational time. In addition, our approach can be used to generate multiple imputation data sets for further decision-support applications.

The future work includes the following two aspects. First, in the retail sales dataset considered in the paper, the price and unit-sales data are treated as separate tensor objects, although there is a strong negative correlation and similar missing pattern between the corresponding price and unit-sales data values. These cross-tensor relationships are often modeled separately using the completed datasets, however, their joint modeling within the current tensor factorization framework is of



(a) On price, 90% training (b) On unit sales, 90% training (c) On price, 10% training (d) On unit sales, 10% training

Figure 11: RMSE on entries with ascending standard deviation of samples. The model's accuracy decreases when its confidence goes down.

considerable interest. Second, there is usually a large amount of side information available in retail sales data, which include product and store meta data. It would be beneficial to to incorporate this side information into tensor factorization approaches.

References

[1] E. Acar, D. Dunlavy, T. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *Proceedings of SIAM International Conference on Data Mining*, 2010.

[2] C. Andrieu, N. Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[3] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] G. Casella and E. George. Explaining the Gibbs sampler. *The American Statistician*, 46:167–174, 1992.

[5] Y. Chi, S. Zhu, Y. Gong, and Y. Zhang. Probabilistic polyadic factorization and its application to personalized recommendation. In *Proceedings of International Conference on Information and Knowledge Management*, 2008.

[6] A. Cichocki, R. Zdunek, A. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Multi-way Exploratory Analysis and Blind Source Separation*. Wiley, 2009.

[7] J. Honaker and G. King. What to do about missing values in times-series cross-section data. *American Journal of Political Science*, 54(2):561–581, 2010.

[8] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), 2009.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 2009.

[10] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. In *Proceedings of IEEE International Conference on Computer Vision*, 2009.

[11] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[12] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup*, 2007.

[13] I. Porteous, E. Bart, and M. Welling. Multi-hdp: A non parametric Bayesian model for tensor factorization. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2008.

[14] D. Rubin. Multiple imputation after 18+ years. *Journal of American Statistical Association*, 91:473–489, 1996.

[15] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2007.

[16] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of International Conference on Machine Learning*, 2008.

[17] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of International Conference on Machine Learning*, 2008.

[18] J. Schafer. Multiple imputation: a primer. *Stat Methods Med Res*, 8(1):3–15, 1999.

[19] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of IEEE International Conference on Data Mining*, 2010.

[20] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of International Conference on Machine Learning*, 2005.

[21] G. Tomasi and R. Bro. Parafac and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):5–43, 2003.

[22] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[23] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor facotorization. In *Proceedings of SIAM International Conference on Data Mining*, 2010.