Interpretable Bayesian Tensor Network Kernel Machines with Automatic Rank and Feature Selection

Afra Kilic H.A.KILIC@TUDELFT.NL
Delft Center for Systems and Control
Delft University of Technology
Delft, 2628 CN, The Netherlands

Kim Batselier K.BATSELIER@TUDELFT.NL
Delft Center for Systems and Control
Delft University of Technology
Delft, 2628 CN, The Netherlands

Abstract

Tensor Network (TN) Kernel Machines speed up model learning by representing parameters as low-rank TNs, reducing computation and memory use. However, most TN-based Kernel methods are deterministic and ignore parameter uncertainty. Further, they require manual tuning of model complexity hyperparameters like tensor rank and feature dimensions, often through trial-and-error or computationally costly methods like cross-validation. We propose Bayesian Tensor Network Kernel Machines, a fully probabilistic framework that uses sparsity-inducing hierarchical priors on TN factors to automatically infer model complexity. This enables automatic inference of tensor rank and feature dimensions, while also identifying the most relevant features for prediction, thereby enhancing model interpretability. All the model parameters and hyperparameters are treated as latent variables with corresponding priors. Given the Bayesian approach and latent variable dependencies, we apply a mean-field variational inference to approximate their posteriors. We show that applying a mean-field approximation to TN factors yields a Bayesian ALS algorithm with the same computational complexity as its deterministic counterpart, enabling uncertainty quantification at no extra computational cost. Experiments on synthetic and real-world datasets demonstrate the superior performance of our model in prediction accuracy, uncertainty quantification, interpretability, and scalability.

Keywords: tensor network kernel machines, tensor decompositions, variational inference, uncertainty quantification, automatic model selection

1 Introduction

Kernel methods, such as Support Vector Machines (SVM) and Gaussian Processes (GP), are powerful tools for nonlinear learning, capable of approximating arbitrary functions given enough data and often matching or outperforming neural networks (Hammer and Gersmann, 2003; Garriga-Alonso et al., 2018; Lee et al., 2018; Novak et al., 2018). They use a kernel function $k(x, x') = \langle \boldsymbol{\varphi}(x), \boldsymbol{\varphi}(x') \rangle$ to implicitly map inputs into a Reproducing Kernel Hilbert Space, where nonlinear problems become linear. This kernel trick avoids explicit computations in high-dimensional spaces but scales poorly with data size, conventional methods require $\mathcal{O}(N^2)$ memory and $\mathcal{O}(N^3)$ flops for N training points (Rasmussen and Williams, 2006; Suykens and Vandewalle, 1999; Suykens et al., 2002). Kernel approxima-

tion methods reduce complexity to $\mathcal{O}(NM^2)$ by projecting data onto M basis function where $M \ll N$: data-dependent (Williams and Seeger, 2001; Drineas and Mahoney, 2005) and data-independent (Rahimi and Recht, 2007) methods both converge at rate $\mathcal{O}(1/\sqrt{M})$. Deterministic approaches (Dao et al., 2017; Mutný and Krause, 2018; Hensman et al., 2017; Solin and Särkkä, 2020) can achieve faster convergence, but their reliance on tensor products causes M to grow exponentially with input dimension D, limiting their use to low-dimensional settings.

Tensor Network (TN) Kernel machines deal with high-dimensional feature spaces and large-scale data sets by considering the model

$$f(\mathbf{x}_n) = \boldsymbol{\varphi}(\mathbf{x}_n)^T \mathbf{w}, \tag{1}$$

with model weights $\mathbf{w}$ and a feature map $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^{M_1 M_2 \cdots M_D}$ that is restricted to a Kronecker product form

$$\boldsymbol{\varphi}(\mathbf{x}_n) = \boldsymbol{\varphi}^{(1)}(x_n^{(1)}) \otimes \boldsymbol{\varphi}^{(2)}(x_n^{(2)}) \otimes \cdots \otimes \boldsymbol{\varphi}^{(D)}(x_n^{(D)}), \tag{2}$$

where each factor is a feature vector $\boldsymbol{\varphi}^{(d)} \in \mathbb{R}^{M_d}$ for all $d \in [1, D]$. As a result only product kernels are considered, which includes common choices such as the polynomial (Batselier et al., 2017; Novikov et al., 2018) and Gaussian kernel (Wesel and Batselier, 2021, 2024). Assume now without loss of generality that $M_1 = M_2 = \cdots = M_D = M$ such that $\mathbf{w} \in \mathbb{R}^{M^D}$. Tensor Network Kernel Machines impose an additional low-rank TN constraint on the weights $\mathbf{w}$. In this way the exponential storage complexity of the parameters to-be-learned is reduced from $O(M^D)$ to $O(DMR)$ or $O(DMR^2)$, depending on whether a canonical polyadic decomposition (CPD) (Harshman, 1970; Carroll and Chang, 1970) or tensor train (TT) (Oseledets, 2011) is used, respectively. Likewise, the Kronecker product structures in both the feature map $\boldsymbol{\varphi}(\cdot)$ and low-rank TN $\mathbf{w}$ can be exploited during training via the alternating least squares (ALS) algorithm with a computational complexity of $O(N(MR)^2 + D(MR)^3)$ when using CPD and $O(N(MR^2)^2 + D(MR^2)^3)$ when using TT. Furthermore, the low-rank structure captures redundancies in $\mathbf{w}$, resulting in a parsimonious representation, while also imposing implicit regularization by constraining the model's degrees of freedom.

Despite the widespread use of TNs for complexity reduction, most existing techniques operate in deterministic settings and do not account for parameter uncertainty. Uncertainty is a key challenge in modern learning due to noisy and limited data. Bayesian inference provides a principled way to handle this uncertainty. Beyond quantifying uncertainty, it also helps control model complexity, choose model size, enforce sparsity, and include prior knowledge. However, the application of TN Kernel machines to reduce model complexity in probabilistic frameworks remains largely underexplored.

Using TNs requires specifying model complexity in advance by tuning hyperparameters like tensor rank R and feature dimension M_d. Choosing the right complexity is essential but challenging. Typically, this is done by trial and error, which can be time-consuming and imprecise. More principled methods like maximum likelihood might cause overfitting if only the training data is used. With ample data, multiple models can be trained on subsets, and the best is chosen based on validation performance. However, when data is limited, using most for training leaves only small validation sets, which may give unreliable results.

Cross-validation improves data use by rotating validation sets but is computationally costly, especially with multiple hyperparameters. Thus, a more efficient and sophisticated method to determine model complexity is needed.

In this paper, to address the gap in probabilistic kernel methods with low-rank TNs and to enable more principled and efficient hyperparameter tuning, we introduce a fully Bayesian tensor network kernel method (BTN-Kernel machines) that automatically infers both the tensor rank R and feature dimension M_d for all $d \in [1, D]$. To achieve this, we specify a sparsity-inducing hierarchical prior over the TN components with individual parameters associated to each feature dimension M_d and rank R component, promoting minimal tensor rank and feature dimension of each TN component. This penalizes the irrelevant components to shrink towards zero, allowing automatic inference of the tensor rank and feature dimensions during training. Additionally, the sparsity parameters placed on the feature dimensions highlights the most relevant features for prediction, enhancing model interpretability. All model parameters are treated as random variables with corresponding priors. Due to the complex dependencies and the fully Bayesian formulation, exact inference is intractable. We therefore employ a mean field variational Bayesian inference to obtain a deterministic approximation of the posterior distributions over all parameters and hyperparameters. We show that employing a mean field approximation on the factors of the TN results in a Bayesian ALS algorithm with an identical computational complexity as the conventional ALS algorithm. In other words, the uncertainty quantification of BTN-Kernel machines can be achieved at zero additional computational cost! We demonstrate the superior performance of our proposed Bayesian model in terms of prediction accuracy, uncertainty quantification, scalability and interpretability through numerous experiments.

1.1 Related Work

Probabilistic models for tensor decompositions have gained attention in collaborative filtering, tensor factorization and completion. Gibbs sampling with Gaussian priors on CP factors is used in (Rai et al., 2015, 2014; Xiong et al., 2010), while variational Bayes (VB) is employed in (Zhao et al., 2015a, 2016). Orthogonal factor recovery via Stiefel manifold optimization with VB is explored in (Cheng et al., 2017). Bayesian low-rank Tucker models are studied using VB (Chu and Ghahramani, 2009; Zhao et al., 2015b) and Gibbs sampling (Hoff, 2016). An infinite Tucker model based on a t-process is proposed in (Xu et al., 2015). A VB Tensor Train (TT) model with von Mises–Fisher priors appears in (Hinrich and Morup, 2019). More recently, a Bayesian TT model has been introduced that sequentially infers each component's posterior, proposing a probabilistic interpretation of the alternating linear scheme (ALS). A MATLAB toolbox supporting both VB and Gibbs sampling has also been developed (Hinrich et al., 2020).

Tensor regression and classification have been explored (Hoff, 2015; Yang and Dunson, 2016; Guhaniyogi et al., 2017), including connections to Gaussian processes (Yu et al., 2018). VB PARAFAC2 models analyze multiple matrices with a varying mode (Jørgensen et al., 2018, 2019), while approaches for multiple 3-way tensors with varying modes appear in (Khan and Kaski, 2014; Khan et al., 2016). While tensor networks are widely used to reduce model complexity in deterministic kernel methods, their use in probabilistic models

remains limited. Existing approaches, such as the TT-GP model, apply variational inference by representing the posterior mean with a TT and using a Kronecker-structured covariance matrix (Izmailov et al., 2018). More recently, the Structured Posterior Bayesian Tensor Network (SP-BTN) further reduces parameter complexity by imposing a low-rank structure on the mean of a CP-decomposed weight tensor, combined with Kronecker-structured local covariances (Konstantinidis et al., 2022). Although both methods perform well in terms of predictive mean, their ability to capture uncertainty is limited by the diagonal structure of the covariance matrices. In fact, uncertainty quantification in the existing models remains largely unexplored. In addition, these methods still require manually tuned hyperparameters, often relying on costly or imprecise optimization procedures.

2 Mathematical Background

2.1 Preliminaries and Notation

The order of a tensor is the number of dimensions, also known as ways or modes. Scalars are denoted by lowercase letters e.g., a. Vectors (first-order tensors) are denoted by boldface lowercase letters, e.g., $\mathbf{a}$. Matrices (second-order tensors) are denoted by boldface capital letters, e.g., $\mathbf{A}$. Higher-order tensors (order ≥ 3) are denoted by boldface calligraphic letters, e.g., $\boldsymbol{\mathcal{A}}$. Given an Dth-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$, the $(i_1, i_2, \ldots, i_D)$-th entry is denoted by $a_{i_1 i_2 \cdots i_D}$, where the indices range from 1 to their capital version, e.g., $i_d = 1, 2, \ldots, I_d, \forall d \in [1, D]$. Often it is easier to avoid working with the tensors directly, thus, we will consider their vectorization. The vectorization $\text{vec}(\boldsymbol{\mathcal{A}}) \in \mathbb{R}^{I_1 I_2 \ldots I_D}$ of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is a vector such that $\text{vec}(\boldsymbol{\mathcal{A}})_i = a_{i_1 i_2 \ldots i_D}$, where the relationship between the linear index i and the multi-index $(i_1, i_2, \ldots, i_D)$ is given by

$$i = i_1 + \sum_{d=2}^{D} (i_d - 1) \prod_{k=1}^{d-1} I_k.$$

When applied to a matrix, the operator $\text{vec}(\cdot)$ performs column-wise vectorization. The reshape operator $\mathcal{R}\{\cdot\}_{I \times J}$ reorganizes the entries of its input into a matrix of size $I \times J$, preserving the column-wise order consistent with the $\text{vec}(\cdot)$ operator. The operator $\text{diag}(\cdot)$ returns a diagonal matrix when applied to a vector, and extracts the main diagonal as a vector when applied to a matrix. The inner product of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^I$ is $c = \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{I} a_i b_i$. The Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{KI \times LJ}$. The Khatri-Rao product $\mathbf{A} \odot \mathbf{B}$ of the matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times J}$ is an $IK \times J$ matrix obtained by taking the column-wise Kronecker product. The Hadamard (element-wise) product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{I \times J}$ is denoted by $\mathbf{A} \circledast \mathbf{B} \in \mathbb{R}^{I \times J}$. The identity matrix is conventionally denoted by $\mathbf{I}$, with its dimensions either inferred from the context or explicitly stated as a subscript.

2.2 Tensor Decompositions

Tensor decompositions, also known as tensor networks, generalize the Singular Value Decomposition (SVD) from matrices to higher-order tensors (Kolda and Bader, 2009). Three widely used decompositions are the Tucker decomposition (Tucker, 1966; Kolda and Bader, 2009), the Tensor Train (TT) decomposition (Oseledets, 2011), and the Canonical Polyadic

Decomposition (CPD) (Harshman, 1970; Carroll and Chang, 1970). Each of these extends different properties of the SVD to tensors. In this subsection, we briefly introduce these decompositions and discuss their advantages and limitations for modeling $\mathbf{w}$ in (2). For a more detailed overview, we refer to (Kolda and Bader, 2009) and references therein. We omit Tucker decomposition due to its non-uniqueness and high storage complexity $\mathcal{O}(R^D)$, which makes it impractical for high-dimensional data. Therefore, since CPD and TT decompositions have storage complexities of $O(DMR)$ and $O(DMR^2)$, respectively, they are both suitable options for representing $\mathbf{w}$ in (1).

Definition 1 (Canonical Polyadic Decomposition) *A rank-R Canonical Polyadic Decomposition of* $\mathbf{w} = \text{vec}(\boldsymbol{\mathcal{W}}) \in \mathbb{R}^{M^D}$ *consists of D factor matrices* $\mathbf{W}^{(d)} \in \mathbb{R}^{M_d \times R}$*, such that*

$$\mathbf{w} = (\mathbf{W}^{(1)} \odot \mathbf{W}^{(2)} \odot \ldots \odot \mathbf{W}^{(D)})\mathbf{1}_R = \sum_{r=1}^{R} \mathbf{w}_r^{(1)} \otimes \mathbf{w}_r^{(2)} \otimes \ldots \otimes \mathbf{w}_r^{(D)}. \tag{3}$$

Unlike matrix factorizations, CPD is unique under mild conditions (Kruskal, 1977). The primary storage cost arises from the D factor matrices, leading to a complexity of $\mathcal{O}(RMD)$.

Definition 2 (Tensor Train Decomposition) *Tensor Train decomposition represents* $\mathbf{w} \in \mathbb{R}^{M^D}$ *using D third-order tensors* $\boldsymbol{\mathcal{W}}^{(d)} \in \mathbb{R}^{R_d \times M_d \times R_{d+1}}$ *such that*

$$w_{i_1 i_2 \ldots i_D} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{D+1}=1}^{R_{D+1}} w_{r_1 i_1 r_2}^{(1)} \ldots w_{r_D i_D r_{D+1}}^{(D)}. \tag{4}$$

The auxiliary dimensions $R_1, R_2, \ldots, R_{D+1}$ *are called TT-ranks. To ensure that the right-hand side of* (4) *is a scalar, the boundary condition* $R_1 = R_{D+1} = 1$ *is required. TT decomposition is non-unique, with a storage complexity of* $\mathcal{O}(R^2MD)$ *due to the D tensors* $\boldsymbol{\mathcal{W}}^{(d)}$.

The CP-rank R and TT-ranks $R_2, \ldots, R_D$ serve as additional hyperparameters, often making CPD preferable in practice. For ease of the discussion, we focus on the learning algorithm for the CPD case. In section 3.5, we briefly discuss what changes to our probabilistic model when using a TT decomposition.

3 Probabilistic Tensor Network Kernel Machines

3.1 Probabilistic Model and Priors

When N inputs $\{\mathbf{x}_n\}_{n=1}^N$ and outputs $\{y_n\}_{n=1}^N$ are available for learning then model (1) can be expressed as a linear matrix equation

$$\mathbf{y} = \boldsymbol{\Phi}^T \mathbf{w} + \mathbf{e}. \tag{5}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ and $\mathbf{e} \in \mathbb{R}^N$, with $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \tau^{-1}\mathbf{I}_N)$. The matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M^D \times N}$ is constructed such that its nth column contains $\boldsymbol{\varphi}(\mathbf{x}_n)$. Due to the tensor-product structure of the feature map in (2), $\boldsymbol{\Phi}$ can be expressed as a Khatri-Rao product of component matrices $\boldsymbol{\Phi} = \boldsymbol{\Phi}^{(1)} \odot \boldsymbol{\Phi}^{(2)} \odot \ldots \odot \boldsymbol{\Phi}^{(D)}$, where the columns of $\boldsymbol{\Phi}^{(d)}$ are the feature vectors $\boldsymbol{\varphi}^{(d)}$, for

all $d \in [1, D]$. Consequently, assuming a CP-decomposed weight vector $\mathbf{w}$ along with the Gaussian noise model, the likelihood of the observed outputs is given by

$$p(\mathbf{y} \mid \{\mathbf{W}^{(d)}\}_{d=1}^{D}, \tau) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{\varphi}(x_n)^T \mathbf{w}, \tau^{-1}\right), \tag{6}$$

where the parameter τ denotes the noise precision. Factor matrices $\mathbf{W}^{(d)} \in \mathbb{R}^{M_d \times R}$ can be represented either row-wise or column-wise as

$$\mathbf{W}^{(d)} = \begin{bmatrix} \mathbf{w}_1^{(d)} & \cdots & \mathbf{w}_{m_d}^{(d)} & \cdots & \mathbf{w}_{M_d}^{(d)} \end{bmatrix}^T = \begin{bmatrix} \mathbf{w}_1^{(d)} & \cdots & \mathbf{w}_r^{(d)} & \cdots & \mathbf{w}_R^{(d)} \end{bmatrix}. \tag{7}$$

A subscript m_d will always refer to a row of the factor matrix $\mathbf{W}^{(d)}$, while a subscript r to a column. The tensor rank R and feature dimensions $\{M_d\}_{d=1}^{D}$ are tuning parameters whose selection is challenging and computationally costly. Therefore, we seek a principled and efficient model selection method that not only infers the model complexity via R and $\{M_d\}_{d=1}^{D}$ but also effectively avoids overfitting. To achieve this, we specify a sparsity-inducing hierarchical prior over the factor matrices $\{\mathbf{W}^{(d)}\}_{d=1}^{D}$ with continuous hyperparameters that control the variance related to their rows and the columns. This promotes simpler models by pushing unnecessary components toward zero, allowing automatic model selection during training. This form of prior is motivated by the framework of automatic relevance determination (ARD) introduced in the context of neural networks by Neal (1996) and MacKay (1994). A similar prior has also been applied in a Bayesian CP tensor completion method, where it enables automatic determination of the CP rank (Zhao et al., 2015a).

More specifically, we define the sparsity parameters as $\boldsymbol{\lambda}_R := [\lambda_1, \lambda_2, \ldots, \lambda_R]$, where each λ_r regulates the strength of the rth **column** $\mathbf{w}_r^{(d)}$ of $\mathbf{W}^{(d)}$, for all $d \in [1, D]$. The parameter set $\boldsymbol{\lambda}_R$ is shared across all factor matrices, allowing uniform regularization across different modes. Similarly, we define $\boldsymbol{\lambda}_{M_d} := [\lambda_{1_d}, \lambda_{2_d}, \ldots, \lambda_{M_d}]$, where each λ_{m_d} controls the regularization of the m_dth **row** $\mathbf{w}_{m_d}^{(d)}$ of $\mathbf{W}^{(d)}$. Unlike $\boldsymbol{\lambda}_R$, the vector of precisions $\boldsymbol{\lambda}_{M_d}$ is specific to each factor matrix $\mathbf{W}^{(d)}$, for all $d \in [1, D]$, allowing for feature-dependent regularization. The prior distribution for the vectorized factor matrices is a zero mean Gaussian prior

$$p(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \boldsymbol{\lambda}_R, \boldsymbol{\lambda}_{M_d}) = \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \mathbf{0}, \boldsymbol{\Lambda}_R^{-1} \otimes \boldsymbol{\Lambda}_{M_d}^{-1}\right), \quad \forall d \in [1, D], \tag{8}$$

where $\boldsymbol{\Lambda}_R \otimes \boldsymbol{\Lambda}_{M_d} = \mathrm{diag}(\boldsymbol{\lambda}_R) \otimes \mathrm{diag}(\boldsymbol{\lambda}_{M_d})$ represents the inverse covariance matrix , also known as the precision matrix. The factor $\boldsymbol{\Lambda}_R$ is shared by all factor matrices and determines the **CP rank**. The factor $\boldsymbol{\Lambda}_{M_d}$ is specific to each mode and determines feature dimension M_d used for each factor matrix. To allow the CP rank and feature dimensions to be inferred from data, we place Gamma hyperpriors over the sparsity parameters $\boldsymbol{\lambda}_R$ and $\boldsymbol{\lambda}_{M_d}$. The hyperprior over $\boldsymbol{\lambda}_R$ is factorized across rank components

$$p(\boldsymbol{\lambda}_R) = \prod_{r=1}^{R} \mathrm{Ga}(\lambda_r \mid c_0, d_0), \tag{9}$$

while the hyperprior over $\boldsymbol{\lambda}_{M_d}$ is factorized over their individual feature dimensions

$$p(\boldsymbol{\lambda}_{M_d}) = \prod_{m_d=1}^{M_d} \mathrm{Ga}(\lambda_{m_d} \mid g_0, h_0), \tag{10}$$

where $\mathrm{Ga}(x \mid a, b) = b^a x^{a-1} e^{-bx}/\Gamma(a)$ denotes a Gamma distribution and $\Gamma(a)$ is the Gamma function. To complete our model with a fully Bayesian treatment, we also place a hyperprior over the noise precision τ , that is,

$$p(\tau) = \mathrm{Ga}(\tau \mid a_0, b_0). \tag{11}$$

For simplicity of notation, all unknowns including latent variables and hyperparameters are collected and denoted together by $\Theta = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(D)}, \boldsymbol{\lambda}_{M_1,\ldots\boldsymbol{\lambda}_{M_D}}, \boldsymbol{\lambda}_R, \tau\}$. The probabilistic graph model is illustrated in Figure 1, from which we can express the joint distribution as

$$p(\mathbf{y}, \Theta) = p(\mathbf{y} \mid \{\mathbf{W}^{(d)}\}_{d=1}^{D}, \tau) \prod_{d=1}^{D} \Big\{ p(\mathbf{W}^{(d)} \mid \boldsymbol{\lambda}_R, \boldsymbol{\lambda}_{M_d})\, p(\boldsymbol{\lambda}_{M_d}) \Big\}\; p(\boldsymbol{\lambda}_R)\, p(\tau). \tag{12}$$

Figure 1 illustrates the probabilistic graphical model corresponding to the joint distribution in (12). For simplicity, the deterministic design matrices $\{\boldsymbol{\Phi}^{(d)}\}_{d=1}^{D}$ are omitted, as they are not treated as random variables. The weight vector $\boldsymbol{w}$ is represented using a CPD decomposition with factor matrices $\{\boldsymbol{W}^{(d)}\}_{d=1}^{D}$. These matrices are governed by a shared sparsity term $\boldsymbol{\lambda}_R$ and individual sparsity terms $\boldsymbol{\lambda}_{M_d}$ for each $\boldsymbol{W}^{(d)}$. The shared term $\boldsymbol{\lambda}_R$ determines the CP rank by penalizing entire columns across all factor matrices, while the individual terms $\mathrm{diag}(\boldsymbol{\lambda}_{M_d})$ control the number of active feature dimensions M_d in each $\boldsymbol{W}^{(d)}$ by penalizing its rows. Like the observation noise precision τ, both $\boldsymbol{\lambda}_R$ and $\boldsymbol{\lambda}_{M_d}$ are precision parameters and are assigned Gamma priors. The shape and scale parameters of these Gamma distributions, shown above the corresponding nodes in the figure, are hyperparameters that must be initialized before training. Hyperparameter initialization is discussed in Section 3.2.5.

By combining the likelihood in (6), the priors of model parameters in (8) and the hyperpriors in (9), (10) and (11), the logarithm of the joint distribution of the model is given by

$$\begin{aligned} l(\Theta) = &-\frac{\tau}{2}\|\mathbf{y} - \langle \boldsymbol{\Phi}, \mathbf{w} \rangle\|_F^2 - \frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d} w_{m_d r}^{(d)}\, \lambda_r\, \lambda_{m_d}\, w_{m_d r}^{(d)} + \left(\frac{N}{2} + a_0 - 1\right)\ln\tau \\ &+ \sum_{r}\left(\frac{\sum_d M_d}{2} + (c_0^r - 1)\right)\ln\lambda_r + \sum_{d}^{D}\sum_{m_d}\left(\frac{R}{2} + (g_0^{dm_d} - 1)\right)\ln\lambda_{m_d}^d \\ &- \sum_{d}^{D}\sum_{m_d} h_0^{dm_d}\lambda_{m_d}^d - \sum_{r} d_0^r\lambda_r - b_0\tau + \mathrm{const}, \end{aligned} \tag{13}$$

where N denotes the total number of observations. See Section 2 of the Appendix for a detailed derivation. Without loss of generality, we can perform maximum a posteriori (MAP)

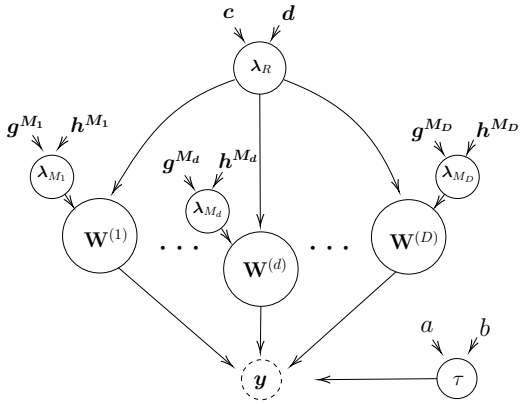

Figure 1: Representation of BTN-Kernel machines with the CPD-decomposed weight vector $\mathbf{w}$ as a probabilistic graphical model showing the hierarchical sparsity inducing priors over the factor matrices $\{\boldsymbol{W}^{(d)}\}_{d=1}^{D}$ by the sparsity parameters $\boldsymbol{\lambda}_R$ and $\{\boldsymbol{\lambda}_{M_d}\}_{d=1}^{D}$. The dashed node denotes the observed data $\mathbf{y}$, while the solid nodes represent random variables. Shape and scale hyperparameters of the Gamma priors placed on $\boldsymbol{\lambda}_R$, $\{\boldsymbol{\lambda}_{M_d}\}_{d=1}^{D}$ and τ are shown as unbounded nodes.

estimation of Θ by maximizing (13), which is, to some extent, equivalent to optimizing a squared error function with regularizations imposed on the factor matrices and additional constraints imposed on the regularization parameters. However, our goal is to develop a method that, instead of relying on point estimates, computes the full posterior distribution of all variables in Θ given the observed data, that is,

$$p(\Theta|\mathbf{y}) = \frac{p(\mathbf{y}, \Theta)}{\int p(\mathbf{y}, \Theta) d\Theta}. \tag{14}$$

Based on the posterior distribution of Θ, the predictive distribution over unseen data points, denoted $\tilde{y_i}$ can be inferred by

$$p(\tilde{y_i} \mid \mathbf{y}) = \int p\left(y_i \mid \Theta\right) p\left(\Theta \mid \mathbf{y}\right) d\Theta. \tag{15}$$

3.2 Model Learning

An exact Bayesian inference in (14) and (15) requires integrating over all latent variables and hyperparameters, making it analytically intractable. In this section, we present the development of a deterministic approximate inference method within the variational Bayesian (VB) framework (Winn and Bishop, 2005) to learn the probabilistic CP model. In this approach, we seek a variational distribution $q(\Theta)$ that approximates the true posterior distribution $p(\Theta \mid \mathbf{y})$ by minimizing the Kullback–Leibler (KL) divergence, that is,

$$\begin{aligned}\mathrm{KL}\left(q(\Theta) \,\|\, p(\Theta \mid \mathbf{y})\right) &= \int q(\Theta) \ln \frac{q(\Theta)}{p(\Theta \mid \mathbf{y})}\, d\Theta \\ &= \underbrace{\ln p(\mathbf{y})}_{\text{evidence}} - \underbrace{\int q(\Theta) \ln \frac{p(\mathbf{y}, \Theta)}{q(\Theta)}\, d\Theta}_{\mathcal{L}(q)},\end{aligned} \tag{16}$$

where $\ln p(\mathbf{y})$ represents the model evidence, and its lower bound is defined by $\mathcal{L}(q) = \mathbb{E}_{q(\Theta)}[\ln p(\mathbf{y}, \Theta)]$. Since the model evidence is a constant, the maximum of the lower bound occurs when the KL divergence vanishes, implying $q(\Theta) = p(\Theta \mid \mathbf{y})$. We assume a mean field approximation that factorizes the variational distribution over each variable $\theta_j \in \Theta$, allowing it to be expressed as

$$q(\Theta) = \prod_{d=1}^{D} \left\{ q_{\mathbf{W}^{(d)}}(\mathbf{W}^{(d)})\, q_{\boldsymbol{\lambda}_{M_d}}(\boldsymbol{\lambda}_{M_d}) \right\} q_{\lambda_R}(\boldsymbol{\lambda}_R) q_\tau(\tau). \tag{17}$$

It is important to note that this factorization assumption is the only assumption imposed on the distribution and the specific functional forms of the individual factors $q_j(\theta_j)$ can be explicitly derived in turn. Consider the jth variable θ_j. The optimal solution for θ_j, obtained by maximizing $\mathcal{L}(q)$ and the maximum occurs when

$$\ln q_j(\theta_j) = \mathbb{E}_{q(\Theta \setminus \theta_j)}[\ln p(\mathbf{y}, \Theta)] + \text{const}, \tag{18}$$

where $\mathbb{E}_{q(\Theta \setminus \theta_j)}[\cdot]$ denotes the expectation taken with respect to the variational distributions of all variables except θ_j. For a detailed derivation and proof, see Bishop (2006). Since all parameter distributions belong to the exponential family and are conjugate to their corresponding prior distributions, we can derive closed-form posterior update rules for each parameter in Θ using (18). Learning the BTN-Kernel machines can then be done by initializing the distributions $q_j(\theta_j)$ appropriately and replacing each in turn with a revised estimate given by the update rule.

3.2.1 Posterior distribution of factor matrices

To derive the update rule for the dth factor matrix $\mathbf{W}^{(d)}$, we first need to introduce the following theorem.

Theorem 3 *Given a set of matrices $\mathbf{W}^{(d)}$ for all $d \in [1, D]$, the following linear relation holds:*

$$\boldsymbol{\Phi}^T \mathbf{w} = \mathrm{vec}(\mathbf{W}^{(d)})^T \boldsymbol{G}^{(d)}, \tag{19}$$

where

$$\mathbf{G}^{(d)} := \boldsymbol{\Phi}^{(d)} \odot \left(\circledast_{k \neq d} \mathbf{W}^{(k)^T} \boldsymbol{\Phi}^{(k)} \right) \in \mathbb{R}^{M_d R \times N},$$

and $\mathrm{vec}(\mathbf{W}^{(d)}) \in \mathbb{R}^{M_d R}$ is the column-wise vectorization of the dth factor matrix. The matrix $\mathbf{G}^{(d)}$ can be interpreted as the design matrix corresponding to $\mathbf{W}^{(d)}$.

Proof See Section 1 of the Appendix. ■

The update for the dth factor matrix $\mathbf{W}^{(d)}$ is based on two main sources of information,

as shown in Figure 1. The first source comes from the observed data and related variables, including the other factor matrices $\mathbf{W}^{(k)}$ for $k \neq d$ and the hyperparameter τ, which are included in the likelihood term (6). The second source comes from sparsity parameters $\mathbf{\Lambda}_R$ and $\mathbf{\Lambda}_{M_d}$, which contribute through the prior term (8). By applying (18), the posterior mean $\text{vec}(\tilde{\mathbf{W}}^{(d)})$ and covariance matrix $\mathbf{\Sigma}^{(d)}$ of

$$q_{\mathbf{W}^{(d)}}(\text{vec}(\mathbf{W}^{(d)})) = \mathcal{N}\left(\text{vec}(\mathbf{W}^{(d)}) \mid \text{vec}(\tilde{\mathbf{W}}^{(d)}), \mathbf{\Sigma}^{(d)}\right), \quad \forall d \in [1, D] \tag{20}$$

are updated by

$$\begin{aligned} \text{vec}(\tilde{\mathbf{W}}^{(d)}) &= \mathbb{E}_q[\tau]\, \mathbf{\Sigma}^{(d)}\, \mathbb{E}_q\left[\mathbf{G}^{(d)}\right]\ \mathbf{y}, \\ \mathbf{\Sigma}^{(d)} &= \left[\mathbb{E}_q[\tau]\, \mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right] + \mathbb{E}_q[\mathbf{\Lambda}_R] \otimes \mathbb{E}_q[\mathbf{\Lambda}_{M_d}]\right]^{-1}. \end{aligned} \tag{21}$$

See Section 3 of the Appendix for a detailed derivation of (21). The matrix $\mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right]$ represents the posterior covariance of the model fitting term $\mathbf{\Phi}^T\mathbf{w}$ in (19), excluding the dth factor matrix $\mathbf{W}^{(d)}$. In other words, it corresponds to the posterior covariance of the design matrix $\mathbf{G}^{(d)}$ which combines the features $\mathbf{\Phi}^{(d)}$ with all factor matrices $\mathbf{W}^{(k)}$ for all $k \neq d$, leaving out the dth factor matrix. This term cannot be computed straightforwardly, and therefore, we first need to introduce the following results. In order to express $\mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right]$ in terms of the posterior parameters of the factor matrices in (21), we reformulate it by isolating the random variables in the expression as in the following theorem.

Theorem 4 *For any fixed $d \in [1, D]$, and assuming that the random matrices $\mathbf{W}^{(k)}$ are independent for all $k \neq d$, the following linear relation holds:*

$$\mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right] = \mathcal{R}\left\{\left(\mathbf{\Phi}^{(d)} \odot \mathbf{\Phi}^{(d)}\right) \overset{D}{\underset{k\neq d}{\circledast}} \left(\mathbf{\Phi}^{(k)} \odot \mathbf{\Phi}^{(k)}\right)^T \mathbb{E}_q\left[\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)}\right]\right\}_{M_dR \times M_dR},$$

where

$$\mathbb{E}_q\left[\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)}\right] = \mathbb{E}_q\left[\mathbf{W}^{(k)}\right] \otimes \mathbb{E}_q\left[\mathbf{W}^{(k)}\right] + \text{Var}\left[\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)}\right], \tag{22}$$

and $\mathcal{R}\{\cdot\}_{M_dR\times M_dR}$ is the operator that reshapes its $M^2 \times R^2$ argument into a matrix of size $M_dR \times M_dR$.

Proof See Section 4 of the Appendix. ■

We require Theorem 4 to evaluate the variance term $\text{Var}[\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)}]$ that appears in $\mathbb{E}_q[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}]$. Notice that when this variance is zero, we recover the standard ALS update equation for $\mathbf{W}^{(d)}$. Now, in order to evaluate (22) in terms of the posterior parameters of the factor matrices from (21), we need to introduce the following Lemma.

Lemma 5 *Let $\mathbf{W}^{(k)} \in \mathbb{R}^{M_k \times R}$ and let $\text{vec}(\mathbf{W}^{(k)}) \in \mathbb{R}^{M_kR}$ be its column-wise vectorization. Then,*

$$\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)} = \mathcal{R}\left\{\text{vec}(\mathbf{W}^{(k)})\, \text{vec}(\mathbf{W}^{(k)})^T\right\}_{M_k^2 \times R^2} \tag{23}$$

where $\mathcal{R}\{.\}_{M_k^2\times R^2}$ reshapes the outer product $\text{vec}(\mathbf{W}^{(k)})\, \text{vec}(\mathbf{W}^{(k)})^T$ from dimensions $M_kR \times M_kR$ to $M_k^2 \times R^2$.

The intuitive interpretation of Lemma 5 is that the Kronecker product of $\mathbf{W}^{(k)}$ with itself is the same as the outer product of $\text{vec}(\mathbf{W}^{(k)})$ with itself, but reshaped into a structured block form. Using this identity, the variance term in (22) can be computed by reshaping the covariance matrix $\mathbf{\Sigma}^{(k)}$ of $\text{vec}(\mathbf{W}^{(k)})$ as

$$\text{Var}\left[\mathbf{W}^{(k)} \otimes \mathbf{W}^{(k)}\right] = \mathcal{R}\left\{\mathbf{\Sigma}^{(k)}\right\}_{M_k^2 \times R^2}. \tag{24}$$

By combining (24) with $\mathbb{E}_q\left[\mathbf{W}^{(k)}\right] = \tilde{\mathbf{W}}^{(k)}$ from (21), the term $\mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right]$ in (22) can be computed explicitly as

$$\mathcal{R}\left\{\left(\mathbf{\Phi}^{(d)} \odot \mathbf{\Phi}^{(d)}\right) \overset{D}{\underset{k\neq d}{\circledast}} \left(\mathbf{\Phi}^{(k)} \odot \mathbf{\Phi}^{(k)}\right)^T \left(\tilde{\mathbf{W}}^{(k)} \otimes \tilde{\mathbf{W}}^{(k)} + \mathcal{R}\left\{\mathbf{\Sigma}^{(k)}\right\}_{M_k^2 \times R^2}\right)\right\}_{M_d R \times M_d R}. \tag{25}$$

An intuitive interpretation of Equation (21) is as follows. The posterior covariance $\mathbf{\Sigma}^{(d)}$ is updated by combining prior information, $\mathbb{E}_q[\mathbf{\Lambda}_R]$ and $\mathbb{E}_q[\mathbf{\Lambda}_{M_d}]$ with contributions from other factor matrices $\mathbb{E}_q\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right]$. These contributions are scaled by the corresponding feature matrix, as shown in (22), and represent the data-dependent part of the update. The impact of this data-dependent term is further weighted by $\mathbb{E}_q[\tau]$, which reflects the model's fit to the data. Hence, better model fit leads to greater reliance on information from the other factors rather than the prior.

The posterior mean $\text{vec}(\tilde{\mathbf{W}}^{(d)})$ is computed by projecting the outcome variable $\mathbf{y}$ onto the expected design matrix $\mathbb{E}_q[\mathbf{G}^{(d)}]$, which captures interactions between the features and the other factor matrices except the dth one. This projection is then scaled by the posterior covariance $\mathbf{\Sigma}^{(d)}$. Finally, the result is scaled by the expected noise precision $\mathbb{E}_q[\tau]$, amplifying the influence of the data when the model fit is good.

3.2.2 Posterior distribution of $\mathbf{\Lambda}_R$ and $\mathbf{\Lambda}_{M_d}$

Instead of point estimation through optimization, learning the posterior of $\boldsymbol{\lambda}_R$ is crucial for automatic rank inference. As seen in Figure 1, the inference of $\boldsymbol{\lambda}_R$ can be done by receiving messages from all the factor matrices and incorporating the messages from its hyperprior. By applying (18), we can identify the posteriors of λ_r, $\forall r \in [1, R]$ as independent Gamma distributions,

$$q_{\boldsymbol{\lambda}_R}(\boldsymbol{\lambda}_R) = \prod_{r=1}^{R} \text{Ga}(\lambda_r \mid c_N^r, d_N^r), \tag{26}$$

where c_N^r, d_N^r denote the posterior parameters learned from N observations and are updated by

$$
\begin{aligned}
c_N^r &= c_0^r + \frac{1}{2}\sum_{d=1}^{D} M_d, \\
d_N^r &= d_o^r + \frac{1}{2}\sum_{d=1}^{D} \mathbb{E}_q\left[\mathbf{w}_r^{(d)T}\mathbf{\Lambda}_{M_d}\mathbf{w}_r^{(d)}\right].
\end{aligned} \tag{27}
$$

See Section 5 of the Appendix for a detailed derivation of (27). The expectation of the inner product of the rth column in the dth factor matrix $\mathbf{w}_r^{(d)}$ with respect to the q-distribution can be computed using the posterior parameters for $\mathbf{W}^{(d)}$ in equation (21) as

$$
\mathbb{E}_q\left[\mathbf{w}_r^{(d)T}\mathbf{\Lambda}_{M_d}\mathbf{w}_r^{(d)}\right] = \tilde{\mathbf{w}}_r^{(d)T}\mathbf{\Lambda}_{M_d}\tilde{\mathbf{w}}_r^{(d)} + \boldsymbol{\lambda}_{M_d}^T \operatorname{Var}\left(\mathbf{w}_r^{(d)}\right), \tag{28}
$$

where $\tilde{\mathbf{w}}_r^{(d)}$ denotes the rth column of $\tilde{\mathbf{W}}^{(d)}$. The second term accounts for the uncertainty in the posterior distribution, where the variance of each element in $\mathbf{w}_r^{(d)}$ is weighted by the corresponding λ_{m_d} in $\boldsymbol{\lambda}_{M_d}$. Notice that the diagonal elements of the posterior covariance matrix $\mathbf{\Sigma}^{(d)}$ corresponds to the variance of every element in $\mathbf{W}^{(d)}$, i.e., $\operatorname{Var}(w_{m_d r}^{(d)})$. Hence, the variance term $\operatorname{Var}(\mathbf{w}_r^{(d)})$ in (28) is also contained in $\operatorname{diag}(\mathbf{\Sigma}^{(d)})$. To efficiently compute this variance $\forall r \in [1, R]$ at once, we reshape the diagonal elements of $\mathbf{\Sigma}^{(d)}$ into a matrix $\mathbf{V}^{(d)} \in \mathbb{R}^{M_d \times R}$, defined as

$$
\mathbf{V}^{(d)} := \mathcal{R}\left\{\operatorname{diag}(\mathbf{\Sigma}^{(d)})\right\}_{M_d \times R}, \tag{29}
$$

such that rth column of $\mathbf{V}^{(d)}$ corresponds to $\operatorname{Var}(\mathbf{w}_r^{(d)})$ in (28). By combining equations (27), (28) with $\mathbf{V}^{(d)}$, we can further simplify the computation of $\mathbf{d}_N = [d_N^1, \ldots, d_N^R]^T$ as

$$
\mathbf{d}_N = \mathbf{d}_0 + \sum_{d=1}^{D} \operatorname{diag}\left(\tilde{\mathbf{W}}^{(d)T}\mathbf{\Lambda}_{M_d}\tilde{\mathbf{W}}^{(d)} + \mathbf{\Lambda}_{M_d}\mathbf{V}^{(d)}\right). \tag{30}
$$

The posterior expectation can be obtained by $\mathbb{E}_q[\boldsymbol{\lambda}_R] = [c_N^1/d_N^1, ..., c_N^R/d_N^R]^T$, and thus $\mathbb{E}_q[\mathbf{\Lambda}_R] = \operatorname{diag}(\mathbb{E}_q[\boldsymbol{\lambda}_R])$. An intuitive interpretation of equation (27) is that λ_r is updated based on the sum of squared L_2-norms of the rth column, scaled by $\mathbf{\Lambda}_{M_d}$ as in equation (28). Consequently, a smaller $||\mathbf{w}_r||$ or $\mathbf{\Lambda}_{M_d}$ leads to larger $\mathbb{E}_q[\lambda_r]$, and updated priors of factor matrices, which in turn more strongly enforces the rth column to be zero.

Just as $\boldsymbol{\lambda}_R$ operates on the columns of $\mathbf{W}^{(d)}$, $\boldsymbol{\lambda}_{M_d}$ acts on its rows . Therefore, learning the posterior of $\boldsymbol{\lambda}_{M_d}$ is essential for determining the feature dimension. The key difference is that $\boldsymbol{\lambda}_{M_d}$ is specific to each factor matrix, whereas $\boldsymbol{\lambda}_R$ is shared across all factor matrices. As shown in Figure 1, the inference of $\boldsymbol{\lambda}_{M_d}$ is performed by gathering information from the dth factor matrix and incorporating information from its hyperprior. By applying (18), we can identify the posteriors of λ_{m_d}, $\forall d \in [1, D]$ and $\forall m_d \in [1, M_d]$ as an independent Gamma distribution,

$$
q_{\boldsymbol{\lambda}_M}(\boldsymbol{\lambda}_{M_d}) = \prod_{m_d=1}^{M_d} \operatorname{Ga}(\lambda_{m_d} \mid g_N^{m_d}, h_N^{m_d}), \tag{31}
$$

where $g_N^{m_d}, h_N^{m_d}$ denote the posterior parameters learned from N observations and are updated by

$$\begin{aligned} g_N^{m_d} &= g_0^{m_d} + \frac{R}{2}, \\ h_N^{m_d} &= h_0^{m_d} + \mathbb{E}_q\left[\mathbf{w}_{m_d}^{(d)T} \mathbf{\Lambda}_R \mathbf{w}_{m_d}^{(d)}\right]. \end{aligned} \tag{32}$$

See Section 6 of the Appendix for a detailed derivation of (32). The expectation of the inner product of the m_dth row in dth factor matrix w.r.t. q distribution can be computed using the posterior parameters $\boldsymbol{W}^{(d)}$ in (21),

$$\mathbb{E}_q\left[\mathbf{w}_{m_d}^{(d)T} \mathbf{\Lambda}_R \mathbf{w}_{m_d}^{(d)}\right] = \tilde{\mathbf{w}}_{m_d}^{(d)T} \mathbf{\Lambda}_R \tilde{\mathbf{w}}_{m_d}^{(d)} + \mathrm{Var}\left(\mathbf{w}_{m_d}^{(d)}\right)^T \boldsymbol{\lambda}_R, \tag{33}$$

where $\tilde{\mathbf{w}}_{m_d}^{(d)}$ denotes the m_dth row of $\tilde{\mathbf{W}}^{(d)}$. The second term in the expression accounts for the uncertainty and is the sum of the variances of the each element in $\mathbf{w}_{m_d}^{(d)}$, each weighted by the corresponding λ_r in $\boldsymbol{\lambda}_R$. To efficiently evaluate this variance $\forall m_d \in [1, M_d]$ we can make use of $\mathbf{V}^{(d)}$ as m_dth row of it corresponds $\mathrm{Var}(\mathbf{w}_{m_d}^{(d)})$. By combining (32), (33) with $\mathbf{V}^{(d)}$, we can further simplify the computation of $\mathbf{h}_N^d = [h_N^{1_d}, ..., h_N^{M_d}]^T$ as

$$\mathbf{h}_N^d = \mathbf{h}_0^d + \mathrm{diag}\left(\tilde{\mathbf{W}}^{(d)} \mathbf{\Lambda}_R \tilde{\mathbf{W}}^{(d)T} + \mathbf{V}^{(d)} \mathbf{\Lambda}_R\right). \tag{34}$$

The posterior expectation can be obtained by $\mathbb{E}_q[\boldsymbol{\lambda}_{M_d}] = [g_N^{1_d}/h_N^{1_d}, ..., g_N^{M_d}/h_N^{M_d}]^T$, and thus $\mathbb{E}_q[\mathbf{\Lambda}_{M_d}] = \mathrm{diag}(\mathbb{E}_q[\boldsymbol{\lambda}_{M_d}])$. Similar to λ_r, λ_{m_d} is updated by the sum of squared L_2 norm of the m_dth row scaled by $\mathbf{\Lambda}_R$, expressed by (33) from the dth factor matrix. Therefore, intuitively, smaller $||\mathbf{w}_{m_d}^{(d)}||$ or $\mathbf{\Lambda}_R$ leads to larger $\mathbb{E}_q[\lambda_{m_d}]$ and updated priors of factor matrices, which in turn more strongly enforces the m_dth row to be zero.

The updates of $\boldsymbol{\lambda}_R$ and $\boldsymbol{\lambda}_{M_d}$ are interdependent. $\boldsymbol{\lambda}_R$ regulates the importance of rank components across all factor matrices, where a larger λ_r implies a less important column. Conversely, $\boldsymbol{\lambda}_{M_d}$ determines the relevance of feature dimensions in the dth factor matrix, with larger λ_{m_d} indicating less important rows. In updating the scale parameter d_N for $\boldsymbol{\lambda}_R$ in (30), the contributions of each row's mean and variance are weighted by λ_{m_d}. Similarly, when updating h_N^d for $\boldsymbol{\lambda}_{M_d}$ in (34), the contributions of each column are scaled by λ_r.

The equations (30) and (34) suggest a negative correlation between $\boldsymbol{\lambda}_R$ and $\boldsymbol{\lambda}_{M_d}$. To illustrate, consider the precision λ_{m_d} of the m_dth row in the dth factor matrix, which is updated based on the squared L_2 norm of that row, weighted by $\mathbf{\Lambda}_R$ as shown in (33). If the row contains large non-zero entries while the corresponding λ_r values are also large, these entries are heavily penalized by the large λ_r values. Consequently, scaling them by large λ_r values increases the corresponding scale parameter $h_N^{m_d}$, which in turn reduces the precision λ_{m_d}. This mechanism enables the model to compensate for the penalization imposed by λ_r, thus preserving the influence of important values in the row. As a result, the model gains more flexibility by balancing penalization between rank components and feature dimensions, allowing it to better capture significant patterns without overly suppressing relevant features.

3.2.3 Posterior distribution of noise precision τ

The noise precision τ can be inferred by receiving information from observed data and its co-parents, including all the factor matrices, and incorporating the information from its hyperprior. Applying (18), the variational posterior is a Gamma distribution , given by

$$q_\tau(\tau) = \mathrm{Ga}(\tau \mid a_N, b_N), \tag{35}$$

where the posterior parameters are updated by

$$\begin{aligned} a_N &= a_0 + \frac{N}{2}, \\ b_N &= b_0 + \frac{1}{2}\mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right]. \end{aligned} \tag{36}$$

See Section 7 of the Appendix for a detailed derivation of (36). The posterior expectation of model error in 36 cannot be computed straightforwardly and, therefore, we need to introduce the following results. First, following from Theorem 3, without isolating the dth factor matrix, the term $\mathbf{\Phi}^T\mathbf{w}$ in (19) can also be written as

$$\mathbf{\Phi}^T\mathbf{w} = \mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}} \mathbf{W}^{(d)T}\mathbf{\Phi}^{(d)}\right). \tag{37}$$

See Section 1 of the Appendix for a detailed derivation. Using this identity, the posterior expectation of model residual can be expressed as

$$\mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right] = \|\mathbf{y}\|_F^2 - 2\,\mathbf{y}^T\,(\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}} \mathbf{E}_q\left[\mathbf{W}^{(d)}\right]^T\mathbf{\Phi}^{(d)}\right)) + \mathbb{E}_q\left[\|\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}} \mathbf{W}^{(d)}\mathbf{\Phi}^{(d)}\right)\|_F^2\right], \tag{38}$$

where the last term can be reformulated isolating the random variables as in the following theorem.

Theorem 6 *Given a set of independent random matrices $\mathbf{W}^{(d)}$ for all $d \in [1, D]$, the following linear relation holds:*

$$\mathbb{E}_q\left[\|\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}} \mathbf{W}^{(d)}\mathbf{\Phi}^{(d)}\right)\|_F^2\right] = \mathbf{1}_N^T\left(\overset{D}{\underset{d=1}{\circledast}} \left(\mathbf{\Phi}^{(d)} \odot \mathbf{\Phi}^{(d)}\right)^T\mathbb{E}_q\left[\mathbf{W}^{(d)} \otimes \mathbf{W}^{(d)}\right]\right)\mathbf{1}_{R^2}, \tag{39}$$

where $\mathbb{E}_q[\mathbf{W}^{(d)} \otimes \mathbf{W}^{(d)}] = \mathbb{E}_q[\mathbf{W}^{(d)}] \otimes \mathbb{E}_q[\mathbf{W}^{(d)}] + \mathrm{Var}[\mathbf{W}^{(d)} \otimes \mathbf{W}^{(d)}]$, and $\mathbb{E}_q[\mathbf{W}^{(d)}] = \tilde{\mathbf{W}}^{(d)}$.

Proof See Section 8 of the Appendix. ■

Using Lemma 1, we can explicitly evaluate the variance term in terms of $\mathbf{\Sigma}^{(d)}$ by reshaping it as in (23). Therefore, the expectation of the residual sum of squares w.r.t. q distribution can be computed using the posterior parameters in (21) as in

$$\begin{aligned} \mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right] =& \|\mathbf{y}\|_F^2 - 2\,\mathbf{y}^T\,(\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)T}\mathbf{\Phi}^{(d)}\right)) \\ &+ \mathbf{1}_N^T\left(\overset{D}{\underset{d=1}{\circledast}} \left(\mathbf{\Phi}^{(d)} \odot \mathbf{\Phi}^{(d)}\right)^T\left(\tilde{\mathbf{W}}^{(d)} \otimes \tilde{\mathbf{W}}^{(d)} + \mathcal{R}\left\{\mathbf{\Sigma}^{(k)}\right\}_{M_d^2 \times R^2}\right)\right)\mathbf{1}_{R^2}. \end{aligned} \tag{40}$$

Finally, the posterior approximation of τ can be obtained $\mathbb{E}_q[\tau] = a_N/b_N$.

3.2.4 Lower Bound Model Evidence

The inference framework presented in the previous section can essentially maximize the lower bound of the model evidence that is defined in (16). Since the lower bound by the definition should not decrease at each iteration, it can be used as a convergence criteria. The lower bound of the log marginal likelihood is computed by

$$\mathcal{L}(q) = \mathbb{E}_{q(\Theta)}[\ln p(\mathbf{y}, \boldsymbol{\Theta})] + H(q(\Theta)), \tag{41}$$

where the first term denotes the posterior expectation of joint distribution, and the second term denotes the entropy of posterior distributions. Various terms in the lower bound are computed and derived by assuming parametric forms for the q distribution, leading to the following results

$$\begin{aligned}\mathcal{L}(q) = &-\frac{a_N}{2b_N}\mathbb{E}_q\left[\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right] - \frac{1}{2}\operatorname{Tr}\left\{\sum_d \left(\tilde{\boldsymbol{\Lambda}}_R \otimes \tilde{\boldsymbol{\Lambda}}_{M_d}\right)\left(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T + \mathbf{V}^{(d)}\right)\right\} \\ &+ \sum_r \left\{\ln\ \Gamma(c_N^r) + c_N^r\left(1 - \ln\ d_N^r - \frac{d_0^r}{d_N^r}\right)\right\} + \sum_d \sum_{m_d}\left\{\ln\Gamma(g_N^{m_d}) + g_N^{m_d}\left(1 - \ln h_N^{m_d} - \frac{h_0^{m_d}}{h_N^{m_d}}\right)\right\} \\ &+ \frac{1}{2}\sum_d \ln\ |\ \mathbf{V}^{(d)}\ | + \ln\ \Gamma(a_N) + a_N(1 - \ln\ b_N - \frac{b_0}{b_N}) + \text{const.}\end{aligned} \tag{42}$$

See Section 10 of the Appendix for a detailed derivation of (42). The posterior expectation of model residuals denoted by $\mathbb{E}_q\left[\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right]$ can be computed using (40). The lower bound can be interpreted as follows. The first term captures the model residual. The second term represents a weighted sum of the squared L_2 norms of the components in the factor matrices, incorporating uncertainty as well. The remaining terms correspond to the negative KL divergence between the posterior and prior distributions of the hyperparameters.

3.2.5 Implementation and Initialization of Hyperparameters

The variational Bayesian inference is guaranteed to converge to a local minimum. To avoid getting stuck in poor local solutions, it is important to choose an initialization point. A commonly used strategy is to adopt *uninformative priors* by setting the hyperparameters $\mathbf{c}_0, \mathbf{d}_0, \mathbf{g}_0^{M_d}$, and $\mathbf{h}_0^{M_d}$ to 10^{-6}. Based on this, the precision matrices for penalization are typically initialized as $\boldsymbol{\Lambda}_{M_d} = \boldsymbol{I}$ for all $d \in [1, D]$ and $\boldsymbol{\Lambda}_R = \boldsymbol{I}$.

For the noise precision, the posterior mean of τ is given by a_N/b_N, where a_N scales with the sample size and b_N decreases during training as it reflects the expected model error (see (36)). This can cause τ to grow large, leading to overfitting. To control this, we choose the hyperparameters for τ slightly higher as $a_0 = b_0 = 10^{-3}$, which still initializes τ to 1. These values can be adjusted based on the data set. For example, setting a_0 and b_0 to give a higher initial value of τ increases the influence of the likelihood in the posterior updates, while a smaller τ makes the prior more dominant as can be seen in (20). Similarly, setting higher initial values for the hyperparameters of $\boldsymbol{\lambda}_R$ or $\boldsymbol{\lambda}_{M_d}$ increases regularization

Algorithm 1 Learning BTN Kernel Machines

Require: Inputs $\mathbf{x} = \{x_n\}_{n=1}^N$ and outputs $\mathbf{y} = \{y_n\}_{n=1}^N$
Initialization: R, $\{M_d\}_{d=1}^D$, $\mathbf{W}^{(d)}, \mathbf{V}^{(d)}$, $\forall d \in [1, D]$, a_0, b_0, $\mathbf{c}_0$, $\mathbf{d}_0$, $\mathbf{g}_0^{M_d}$, $\mathbf{h}_0^{M_d}$ and set $\tau = a_0/b_0$, $\lambda_r = c_0^r/d_0^r$, $\forall r \in [1, R]$, $\lambda_{M_d} = g_0^{m_d}/h_0^{m_d}$, $\forall d \in [1, D]$, $\forall m_d \in [1, M_D]$
repeat
  for $d = 1$ to D **do**
    Update the posterior $q_d(\text{vec}(\mathbf{W}^{(d)}))$ using equation (21)
  end for
  for $d = 1$ to D **do**
    Update the posterior $q(\boldsymbol{\lambda}_{M_d})$ using equation (32)
  end for
  Update the posterior $q(\boldsymbol{\lambda}_R)$ using equation (26)
  Update the posterior $q(\tau)$ using equation (36)
  Evaluate the lower bound using equation (42)
  if truncation criterion met **then**
    Reduce rank R by eliminating zero-columns of $\mathbf{W}^{(d)}$ $\forall d \in [1, D]$
  end if
until convergence
Compute the predictive distribution using (43)

on the columns or rows of the factor matrices, promoting sparsity by pruning less relevant components.

The factor matrices $\mathbf{W}^{(d)}$ for all $d \in [1, D]$ are initialized from $\mathcal{N}(0, \boldsymbol{I})$, and the covariance matrix $\Sigma^{(d)}$ is set to $\sigma^2 \boldsymbol{I}$ with $\sigma^2 = 10^{-1}$. In the CPD case, due to the Hadamard product structure, initialization σ^2 is sensitive as it can cause numerical instability when D is large. As can be seen in (20), large σ^2 can lead to inflated posterior updates of the factor matrices, while small σ^2 can excessively shrink them. In other words, since these matrices are combined via elementwise multiplication, extreme values can result in outputs tending toward infinity or zero as D increases. Finally, the tensor rank R and the feature dimensions M_d, $\forall d \in [1, D]$ can be manually initialized based on the available computational resources, further discussion on their initialization can be found in Section 4.3.

The complete inference procedure is summarized in Algorithm 1. We begin by updating the posterior of the factor matrices, followed by the higher-order parameters in order from local to global. For instance, $q(\boldsymbol{\lambda}_{M_d})$ depends only on the corresponding factor matrix, while $q(\boldsymbol{\lambda}_R)$ gets information from all factor matrices. Since their updates are interdependent, $q(\boldsymbol{\lambda}_{M_d})$ is updated first to provide more accurate and stable input to the update of $q(\boldsymbol{\lambda}_R)$, which improves convergence and numerical stability.

To improve efficiency in the implementation, we avoid rebuilding $\mathbf{G}^{(d)}$ from scratch in every iteration. Instead, we update its components incrementally. For simplicity, we assume all feature dimensions are equal, i.e., $M_1 = \cdots = M_D = M$. Constructing $\mathbf{G}^{(d)} = \boldsymbol{\Phi}^{(d)} \odot \left(\circledast_{k \neq d} \mathbf{W}^{(k)^T} \boldsymbol{\Phi}^{(k)} \right)$ requires $\mathcal{O}(DNMR)$ operations per iteration, which can be costly. To reduce this cost, we reuse previously computed results. From (37), we know that $\mathbf{G}^{(d)}$ isolates the dth factor matrix from the full Hadamard product $\circledast_{d=1}^D \mathbf{W}^{(d)^T} \boldsymbol{\Phi}^{(d)}$. We compute this full product once, and for each mode d, we divide it elementwise by $\mathbf{W}^{(d)^T} \boldsymbol{\Phi}^{(d)}$, perform the update, and then multiply the updated version back in.

Likewise, to avoid recomputing the expected design matrix covariance $\mathbb{E}_q[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}]$ in (25), we calculate the full expression $\circledast_{d=1}^{D}(\mathbf{\Phi}^{(d)} \odot \mathbf{\Phi}^{(d)})^T(\tilde{\mathbf{W}}^{(d)} \otimes \tilde{\mathbf{W}}^{(d)} + \mathcal{R}\{\mathbf{\Sigma}^{(d)}\}_{M_d^2 \times R^2})$ only once. Then, for each factor matrix update, we remove the contribution from mode d by dividing out its term and reinsert it after the update. This reuse strategy reduces redundant computations and improves the overall efficiency.

We further speed up the implementation by eliminating the zero columns of $\{\mathbf{W}^{(d)}\}_{d=1}^{D}$ after each iteration. The reason for eliminating only the zero columns, instead of both columns and rows, is to avoid reconstructing $\mathbf{G}^{(d)}$ from scratch at every iteration. Since $\mathbf{G}^{(d)}$ is constructed by summing the rows of $\mathbf{W}^{(d)}$ weighted by $\mathbf{\Phi}^{(d)}$, removing rows from the factor matrices means reconstructing $\mathbf{G}^{(d)}$ from scratch at every iteration, which is computationally expensive. Truncation criteria can be set manually. In our implementation, we retain all R components for the first three iterations to allow the model enough flexibility before removing potentially useful components. After that, we remove components that contribute less than 10^{-5} to the total variance. To identify low-variance components, we stack the factor matrices column-wise to form $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)}, \tilde{\mathbf{W}}^{(2)}, \ldots, \tilde{\mathbf{W}}^{(D)}]$, compute $\tilde{\mathbf{W}}^{\top}\tilde{\mathbf{W}}$, and remove components whose diagonal values fall below the 10^{-5} variance threshold.

3.3 Predictive Distribution

The predictive distribution over unseen data points, given training data, can be approximated by using variational posterior distribution, that is,

$$\begin{aligned} p(\tilde{y}_i \mid \mathbf{y}) &= \int p\left(y_i \mid \Theta\right) p\left(\Theta \mid \mathbf{y}\right) d\Theta \\ &\simeq \int\int p\left(\tilde{y}_i \mid \left\{\mathbf{W}^{(d)}\right\}, \tau^{-1}\right) q\left(\left\{\mathbf{W}^{(d)}\right\}\right) q(\tau) d\left\{\mathbf{W}^{(d)}\right\} d\tau \end{aligned} \tag{43}$$

Approximation of these integrations yields a Student's t-distribution $\tilde{y}_i \mid \mathbf{y} \sim \mathcal{T}(\tilde{y}_i, \mathcal{S}_i, \nu_y)$ with its parameters given by

$$\begin{aligned} \tilde{y}_i &= \overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)} \varphi_i^{(d)}, \qquad \nu_y = 2a_N, \\ \mathcal{S}_i &= \left\{ \frac{b_N}{a_N} \sum_d \mathbf{g}^{(d)}(x_n)^T \mathbf{\Sigma}^{(d)} \mathbf{g}^{(d)}(x_n) \right\}^{-1}. \end{aligned}$$

See Section 11 of the Appendix for a detailed derivation. Thus, the predictive variance can be obtained by $\mathrm{Var}(y_i) = \frac{\nu_y}{\nu_y - 2}\mathcal{S}_i^{-1}$.

3.4 Computational Complexity

The total computational cost of computing the posterior parameters for all factor matrices $\mathbf{W}^{(d)}$ in (21) is $\mathcal{O}\left(\sum_d NM_d^2R^2 + M_d^3R^3\right)$, where N is the number of observations, M_d is the feature dimension of the dth feature, and R is the tensor rank. The overall cost grows linearly with the number of observations N and the input dimension D, and polynomially with the model complexity parameters M_d and R. The cost of computing the model complexity hyperparameters $\boldsymbol{\lambda}_R$ and $\boldsymbol{\lambda}_{M_d}$ is $\mathcal{O}(\sum_d M_d R^2)$ and $\mathcal{O}(\sum_d M_d^2 R)$ respectively. Finally the

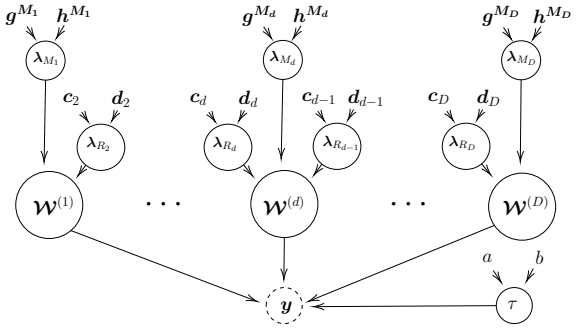

Figure 2: Representation of BTN-Kernel machines with the TT-decomposed weight vector $\mathbf{w}$ as a probabilistic graphical model showing the hierarchical sparsity inducing priors over the core tensors $\{\boldsymbol{\mathcal{W}}^{(d)}\}_{d=1}^{D}$ by the sparsity parameters $\{\boldsymbol{\lambda}_R\}_{d=2}^{D}$ and $\{\boldsymbol{\lambda}_{M_d}\}_{d=1}^{D}$. The dashed node denotes the observed data $\mathbf{y}$, while the solid nodes represent random variables. Shape and scale hyperparameters of the Gamma priors placed on $\{\boldsymbol{\lambda}_R\}_{d=2}^{D}$, $\{\boldsymbol{\lambda}_{M_d}\}_{d=1}^{D}$ and τ are shown as unbounded nodes.

computational cost of computing the noise precision τ is only $\mathcal{O}(NR)$. M_d and R are typically much smaller than N and the rank R is automatically inferred during training and zero components are pruned early, its value tends to decrease rapidly in the first few iterations. As a result when $N >> M_d R$ the total computational complexity of Algorithm 1 is dominated by the factor matrix updates which is $\mathcal{O}\left(\sum_d NM_d^2R^2 + M_d^3R^3\right)$, making it suitable for learning problems which are large in both N and D.

3.5 Tensor Train Kernel Machines

Tensor Trains are another common choice to parameterize $\mathbf{w}$. These models replace the factor matrices $\boldsymbol{W}^{(d)}$ of the CPD with third-order core tensors $\boldsymbol{\mathcal{W}}^{(d)} \in \mathbb{R}^{R_d \times M_d \times R_{d+1}}$. Extending the probabilistic model to Tensor Train Kernel Machines is straightforward. The prior of the Tensor Train core tensors is specified as

$$p(\mathrm{vec}(\boldsymbol{\mathcal{W}}^{(d)}) \mid \boldsymbol{\lambda}_{R_d}, \boldsymbol{\lambda}_{M_d}, \boldsymbol{\lambda}_{R_{d+1}}) = \mathcal{N}\left(\mathrm{vec}(\boldsymbol{\mathcal{W}}^{(d)}) \mid \mathbf{0}, \boldsymbol{\Lambda}_{R_{d+1}}^{-1} \otimes \boldsymbol{\Lambda}_{M_d}^{-1} \otimes \boldsymbol{\Lambda}_{R_d}^{-1}\right), \quad \forall d \in [1, D].$$

Since a Tensor Train has $D-1$ ranks $R_2, \ldots, R_D$ there will be $D-1$ corresponding prior precision matrices $\boldsymbol{\Lambda}_{R_2}, \ldots, \boldsymbol{\Lambda}_{R_D}$. The increase in number of ranks leads to models that are more flexible but hence come at the cost of more parameters to optimize. Just as with the CPD-based model, the computational complexity of updating the posterior is dominated by the update of $q_d(\mathrm{vec}(\boldsymbol{\mathcal{W}}^{(d)}))$, which is $\mathcal{O}(N(R_d M_d R_{d+1})^2 + (R_d M_d R_{d+1})^3)$ flops.

Figure 2 shows the probabilistic graphical model representing the joint distribution for the case where $\mathbf{w}$ is decomposed using the Tensor Train (TT) format. For simplicity, the deterministic design matrices $\{\boldsymbol{\Phi}^{(d)}\}_{d=1}^{D}$ are not shown, similar to the CPD case in Figure 1. As illustrated in Figure 2, the main difference in the model's parameterization, compared to the CPD decomposed $\mathbf{w}$ model is that the TT model includes $D-1$ independent sparsity parameters $\boldsymbol{\lambda}_{R_d}$. These parameters are used to infer the tensor rank of each core in the TT decomposition.

4 Experiments

We first conducted an experiment using synthetic data to illustrate the effect of penalization through the sparsity parameters $\boldsymbol{\lambda}_{M_d}$ and $\boldsymbol{\lambda}_R$ on the rows and columns of the factor matrices. Next, using real data, we analyzed the model's convergence as well as the impact of the initial rank and feature dimension on its performance. Finally, we compared BTN-kernel machines with three state-of-the-art methods: T-KRR (Wesel and Batselier, 2021), SP-BTN (Konstantinidis et al., 2022), and GP (Rasmussen and Williams, 2006), evaluating their predictive performance. In all experiments, we applied a polynomial feature mapping followed by normalization to unit norm. To improve numerical stability for large D, we added a constant offset of 0.2 to each feature matrix $\boldsymbol{\Phi}^{(d)}$ after construction. This shift mitigates the risk of vanishing values in the final output in (37), which involves the Hadamard product applied D times. Our Python code is available at `https://github.com/afrakilic/BTN-Kernel-Machines` and allows the reproduction of all experiments in this section.

4.1 Ground Truth Recovery with Synthetic Data

To illustrate the effect of penalization via $\boldsymbol{\lambda}_{M_d}$ and $\boldsymbol{\lambda}_R$ on the rows and columns of the factor matrices, we first present a synthetic experiment. The ground truth is constructed using the following procedure. Each column of the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ is independently sampled from a standard normal distribution, $\mathcal{N}(0, \boldsymbol{I})$, with $N = 500$ and $D = 3$. Then, D factor matrices $\mathbf{W}^{(d)} \in \mathbb{R}^{M_d \times R}$ are drawn from $\mathcal{N}(0, \boldsymbol{I})$ and scaled by 10 to ensure non-zero values, with the true rank set to $R = 3$. For each feature dimension M_d, a random subset of feature indices (up to 5) is selected, resulting in $M_1 = 1$, $M_2 = 4$, and $M_3 = 3$. The true observed data is then constructed as $\mathbf{y} = \boldsymbol{\Phi}^T \mathbf{w} + \mathbf{e}$, where $\epsilon \sim \mathcal{N}(0, \sigma_e^2)$ with $\sigma_e = 0.001$ denotes i.i.d. additive noise.

To test whether $\boldsymbol{\lambda}_{M_d}$ and $\boldsymbol{\lambda}_R$ penalize noisy rows and columns respectively, the model is initialized as described in Section 3.2.5, with an initial rank of 5 and feature dimension of 5 for all modes. The posterior means of the factor matrices, $\tilde{\mathbf{W}}^{(d)}$ are illustrated in Figure 3 along with $\tilde{\boldsymbol{\lambda}}_R$ and $\tilde{\boldsymbol{\lambda}}_{M_d}$. Three factor matrices are inferred in which two columns become zero, resulting in the correct estimation of the tensor rank as 3. Furthermore, in order to asses the matrix ranks of the estimated factor matrices, singular value analysis was performed. For $\tilde{\mathbf{W}}^{(1)}$, whose true rank is 1, the second singular value drops by over five orders of magnitude from 447.34 to 1.60×10^{-3}, indicating rank 1. For $\tilde{\mathbf{W}}^{(2)}$ and $\tilde{\mathbf{W}}^{(3)}$, both with true rank 3, the fourth singular values are near zero (1.40×10^{-16} and 1.64×10^{-17}), showing drops of more than 15 orders of magnitude from the third singular values, indicating rank 3.

Figure 3 shows that the posterior mean of $\tilde{\lambda}_r$ is smaller for columns that remain nonzero after estimation and larger for those that shrink to zero. In particular, it is at least 5.5 times larger for zero columns compared to nonzero ones. Likewise, for the correct feature dimensions of each factor matrix, rows with large corresponding $\tilde{\lambda}_{m_d}$ values are strongly penalized, causing their entries to become close to zero. To illustrate this, consider the first factor matrix $\mathbf{W}^{(1)}$, where the true feature dimension is $M_1 = 1$. In the posterior mean $\tilde{\mathbf{W}}^{(1)}$, the smallest nonzero entry in the first row is approximately five orders of magnitude

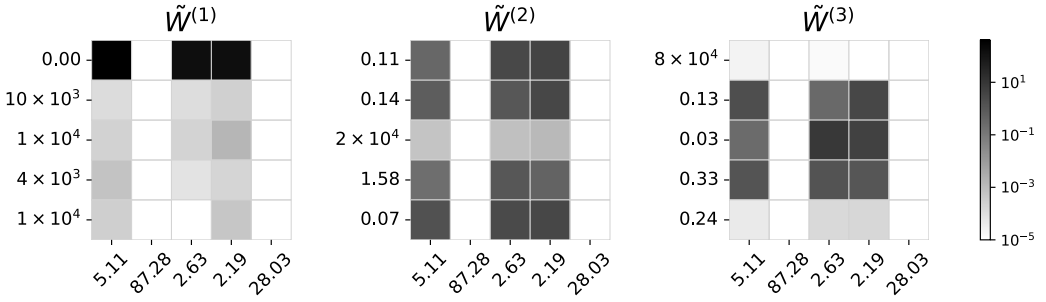

Figure 3: Posterior means of the factor matrices $\tilde{\mathbf{W}}^{(d)}$ for all $d \in [1, 3]$, along with $\tilde{\boldsymbol{\lambda}}_R$ and $\tilde{\boldsymbol{\lambda}}_{M_d}$. The row labels to the left of each matrix show the posterior mean of $\tilde{\boldsymbol{\lambda}}_{M_d}$, specific to each factor matrix. The column labels below the matrices represent the posterior mean of $\tilde{\boldsymbol{\lambda}}_R$, shared across all factor matrices. The model correctly identifies the tensor rank and feature dimensions by assigning larger precision values to unnecessary rows and columns, which makes their values shrink zero or close to zero.

larger than the largest entry in the other rows, consistent with the corresponding precision value $\lambda_{m_1=1} \approx 0$.

An interesting observation is that the range of λ_M values is much larger than that of λ_R, spanning from approximately 5.5×10^{-6} to 7.9×10^4, which covers over ten orders of magnitude. In contrast, λ_R ranges only from 2.2 to 87.3, just under two orders of magnitude. However, even when λ_{m_d} takes very large values, in most cases the corresponding rows in the factor matrices do not become exactly zero, unlike the columns affected by large λ_r. This difference is due to the structure of CPD, which involves an elementwise (Hadamard) product across the columns of the factor matrices. This structure appears in the design matrix $\mathbf{G}^{(d)} = \boldsymbol{\Phi}^{(d)} \odot \left(\circledast_{k \neq d} \mathbf{W}^{(k)^T} \boldsymbol{\Phi}^{(k)} \right)$ which is used in the posterior mean update of the factor matrices (20). When any column is strongly penalized by a large λ_r, its values become close to zero. Since the CPD model multiplies these columns elementwise, the entire product for that component also approaches zero, causing those columns themselves to become zero. In contrast, rows contribute through summation as in $\mathbf{W}^{(k)T} \boldsymbol{\Phi}^{(k)}$, rather than through multiplication.. As a result, even when strongly penalized, their values usually become very small rather than exactly zero. Moreover, as shown in Figure 3, $\boldsymbol{\lambda}_R$ is shared across all factor matrices and penalizes the same columns in every mode. In contrast, $\boldsymbol{\lambda}_{M_d}$ is independent across modes and penalizes the rows of its corresponding factor matrix, resulting in different rows being penalized in each mode. Finally, the posterior mean of the noise precision was estimated as $\tau \approx 204000$, indicates the method's effectiveness in denoising, with the estimated noise variance $\tilde{\sigma}_e \approx 0.002$.

4.2 Model Convergence

In this experiment, we examine the convergence behavior of learning the model by tracking the variational lower bound and effective rank during training. The variational lower bound,

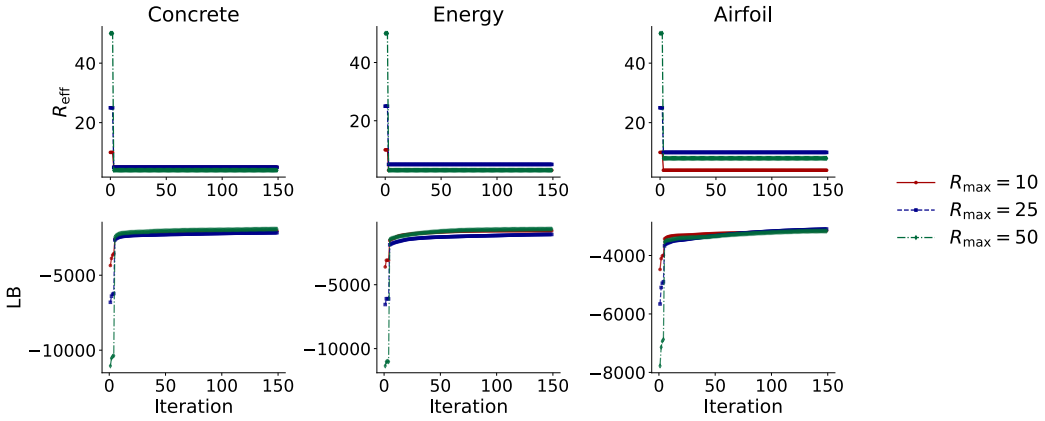

Figure 4: R_{eff} and the variational lower bound (LB) during the training for varying initial rank values $R_{\max}$. The variational LB and R_{eff} quickly stabilize early in training, with all models converging to low-rank solutions regardless of the initial $R_{\max}$.

which by definition should not decrease at each iteration, serves as a convergence criterion. To evaluate model convergence, we analyzed the behavior of the variational lower bound during training on three UCI data sets: Airfoil, Concrete, and Energy. Furthermore, since the rank is reduced during training, we also analyzed the behavior of the effective rank R_{eff}, specifically to assess whether it converges to a stable value or continues to decrease throughout training. We define R_{eff} as the number of components that explain more than 10^{-5} of the total variance. To compute it, we stack the factor matrices column-wise to form the matrix $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)}, \tilde{\mathbf{W}}^{(2)}, \ldots, \tilde{\mathbf{W}}^{(D)}]$, compute the variance contributions from the diagonal of $\tilde{\mathbf{W}}^T\tilde{\mathbf{W}}$, and count the number of components exceeding the threshold.

The data sets are split into random training and test sets with 90% and 10% of the data, respectively. Each data set is normalized within the training set so that the features and target values had zero mean and unit variance. All models are initialized as described in Section 3.2.5, with the feature dimension fixed at 20 across all modes. To examine whether the effective rank and lower bound convergence change with different initial rank values, we consider three initial rank values, $R_{\max}$, 10, 25, and 50. To promote low-rank solutions, we set the hyperparameters to $c_0 = 10^{-5}$ and $d_0 = 10^{-6}$.

In Figure 4, observe that the variational lower bound converges after only a few iterations. Similarly, the effective rank R_{eff} stabilizes early in training and remains consistent, regardless of the initial rank. In all cases, the models converged to low-rank solutions. Therefore, we assume convergence when the relative change in the variational lower bound is smaller than a user-defined threshold of 10^{-4} between iterations; otherwise, training is stopped when the maximum number of iterations, set to 50, is reached.

			RMSE			NLL		
Data set	N	D	$R_{\max} = 10$	$R_{\max} = 25$	$R_{\max} = 50$	$R_{\max} = 10$	$R_{\max} = 25$	$R_{\max} = 50$
Airfoil	1503	5	2.047 ± 0.204	**1.723 ± 0.151**	1.786 ± 0.209	3.491 ± 0.110	**2.865 ± 0.152**	2.884 ± 0.160
Concrete	1030	8	6.165 ± 1.207	5.452 ± 1.242	**5.343 ± 0.860**	3.403 ± 0.091	3.387 ± 0.171	**3.317 ± 0.161**
Energy	768	9	1.321 ± 0.227	**1.114 ± 0.312**	1.419 ± 0.288	3.268 ± 0.450	**2.174 ± 0.542**	2.466 ± 0.982

Table 1: Average test RMSE and NLL with standard errors for varying initial rank values $R_{\max}$. Increasing $R_{\max}$ from 10 to 25 generally improves performance, while further increasing it to 50 yields no consistent improvement in RMSE or NLL.

			$R_{\text{eff}}\pm$ **Std.**			**% of** $R_{\max}$		
Dataset	N	D	$R_{\max} = 10$	$R_{\max} = 25$	$R_{\max} = 50$	$R_{\max} = 10$	$R_{\max} = 25$	$R_{\max} = 50$
Airfoil	1503	5	4.0 ± 0.0	9.5 ± 0.5	7.7 ± 1.4	40.0%	38.0%	15.4%
Concrete	1030	8	4.4 ± 0.7	5.3 ± 0.5	5.2 ± 0.7	44.0%	21.2%	10.4%
Energy	768	9	3.0 ± 0.0	4.9 ± 0.9	3.0 ± 0.0	30.0%	19.6%	6.0%

Table 2: Average effective rank with standard errors and percentage of R_{eff} relative to $R_{\max}$. The model automatically lowers its R_{eff} despite a high initial $R_{\max}$ by applying stronger penalties that remove unnecessary components.

4.3 Initial rank and feature dimension

We performed another series of experiments to evaluate the impact of varying initial rank values $R_{\max}$ and initial feature dimension values $M_{\max}$ on model performance. Additionally, in this section, we present results illustrating the effect of penalization by $\mathbf{\Lambda}_{M_d}$ on the rows of the factor matrices, independently of Λ_R. In the experiments, the same UCI data sets and training procedures from the previous section are used. For each data set, the data splitting is repeated 10 times. The normalization on the targets is removed for prediction and the average test performance of each method is reported.

To assess how the initial rank affects the test performance, we compared models with different $R_{\max}$ values. Table 1, shows the average test root mean squared error (RMSE) and negative log-likelihood (NLL) for each initial rank. The best result for each data set is highlighted in bold. Increasing $R_{\max}$ from 10 to 25 generally improved results, but raising it to 50 did not consistently improve RMSE or NLL.

Table 2 presents the average effective rank with standard errors, along with its percentage relative to the initial rank ($R_{\max}$). As $R_{\max}$ increases, the percentage of effective rank consistently decreases across all data sets. This shows that the penalization term Λ_R limits model complexity by reducing the use of unnecessary rank components. Although the percentage of retained rank is highest at $R_{\max} = 10$, more than half of the columns in the factor matrices are still eliminated. Consequently, the average effective rank is relatively small. This helps explain the improvement in RMSE observed in Table 1 when $R_{\max}$ is increased from 10 to 25. Given that the model promotes low rank solutions with the hyperparameters $c_0 = 10^{-5}$ and $d_0 = 10^{-6}$, initializing with a low $R_{\max}$ limits the model's capacity, often leading to underfitting.

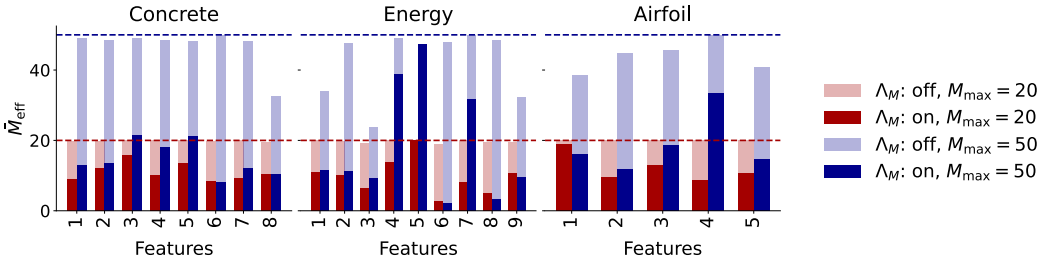

Figure 5: The average feature dimension, $\bar{M}_{\text{eff}}$, for each feature across the data sets when $\mathbf{\Lambda}_{M_d}$ is on and off. The average feature dimension $\bar{M}_{\text{eff}}$ is consistently lower when the row-wise penalization term $\mathbf{\Lambda}_{M_d}$ is applied.

		RMSE		**NLL**		$\mathbf{R}_{\text{eff}}$	
Dataset	$M_{\max}$	λ_M-on	λ_M-off	λ_M-on	λ_M-off	λ_M-on	λ_M-off
Concrete	20	5.452 ± 1.242	**5.328 ± 1.192**	**3.387 ± 0.171**	4.095 ± 0.168	5.3 ± 0.5	5.3 ± 0.5
	50	**5.241 ± 1.075**	5.490 ± 0.679	**3.811 ± 0.203**	5.381 ± 0.314	6.3 ± 0.8	6.3 ± 0.8
Energy	20	**1.114 ± 0.312**	1.150 ± 0.155	**2.174 ± 0.543**	2.810 ± 0.621	4.9 ± 0.9	4.5 ± 0.7
	50	**1.353 ± 0.181**	1.441 ± 0.202	**3.738 ± 0.379**	4.697 ± 0.374	2.7 ± 0.6	2.7 ± 0.6
Airfoil	20	**1.723 ± 0.151**	1.746 ± 0.182	**2.865 ± 0.152**	3.046 ± 0.113	9.5 ± 0.5	9.3 ± 0.5
	50	1.727 ± 0.181	**1.700 ± 0.155**	**3.557 ± 0.288**	3.971 ± 0.216	7.9 ± 0.7	7.8 ± 0.7

Table 3: Average predictive RMSE and NLL with standard errors, along with R_{eff}, for varying initial input dimension values $M_{\max}$ when $\mathbf{\Lambda}_{M_d}$ is on and off. The row-wise penalization $\mathbf{\Lambda}_{M_d}$ improves uncertainty estimation by reducing NLL and promotes row sparsity without affecting the effective rank, helping prevent overfitting especially as $M_{\max}$ increases.

The results also suggest that the degree of penalization increases with higher $R_{\max}$, as reflected by the slightly lower average effective rank when $R_{\max} = 50$ compared to $R_{\max} = 25$. However, as shown in Table 1, the performance difference between $R_{\max} = 25$ and $R_{\max} = 50$ is marginal. This can be attributed to the fact that once $R_{\max}$ is set sufficiently high, the model is given enough flexibility at initialization and tends to converge to a similar level of complexity. Therefore, we recommend initializing $R_{\max}$ at a sufficiently high value, given the available computational resources. For instance, in data sets with higher dimensionality (D), initializing $R_{\max}$ to lower values with lower Λ_R can be more computationally efficient, especially since the rank remains fixed during the initial iterations.

As shown with the synthetic data experiment in Figure 3, the Hadamard structure of CPD leads strongly penalized columns to become exactly zero, while the corresponding rows tend to remain small but nonzero. Therefore, for both structural and computational reasons, we eliminate zero columns during training but retain the rows. Since this behavior has already been demonstrated on synthetic data, the primary goal of the following experiment is twofold: first, to confirm that penalization on the rows is also effective for real data sets, and second, to assess the impact of the initial feature dimension on test performance. To this end, we fix $R_{\max} = 25$ and consider two values for $M_{\max}$: 25 and 50. We train the

models both when the row-wise penalization term $\mathbf{\Lambda}_{M_d}$ is enabled and disabled. Specifically, setting $\mathbf{\Lambda}_{M_d} = \boldsymbol{I}$ and keeping it fixed throughout training results in no penalization on the rows; we refer to this setting as $\mathbf{\Lambda}_{M_d}$: off. Conversely, when $\mathbf{\Lambda}_{M_d}$ is updated during training, we denote it as $\mathbf{\Lambda}_{M_d}$: on.

We define the effective feature dimension, M_{eff}, for each factor matrix $\tilde{\mathbf{W}}^{(d)}$ as the count of rows whose variance contribution, given by the corresponding diagonal entry of $\tilde{\mathbf{W}}^{(d)}\tilde{\mathbf{W}}^{(d)T}$, exceeds 0.25% of the total variance, defined as the sum of all diagonal entries. Figure 5 shows the average feature dimension, $\bar{M}_{\text{eff}}$, per feature, computed over 10 train-test splits for each data set with and without the row-wise penalization term $\mathbf{\Lambda}_{M_d}$. The results show that for both values of $M_{\max}$, the average feature dimension, $\bar{M}_{\text{eff}}$, is consistently lower when $\mathbf{\Lambda}_{M_d}$ is on compared to when it is off. Furthermore, $\mathbf{\Lambda}_{M_d}$ provides insights into feature relevance. For example, in the Energy data set, feature 6 contributes less to the total variance in the outcome variable compared to feature 5. This concludes that $\mathbf{\Lambda}_{M_d}$ penalizes less relevant feature dimensions while reinforcing those with higher contributions.

Table 3 shows that the row-wise penalization $\mathbf{\Lambda}_{M_d}$ generally reduces NLL across all data sets and initial feature dimension choices, indicating improved uncertainty estimation. RMSE is also slightly better or comparable in all cases. The effective rank R_{eff} remains stable across penalized and unpenalized settings, indicating that $\mathbf{\Lambda}_{M_d}$ mainly promotes sparsity in the row space without affecting the columns space *or* the rank structure.

Increasing $M_{\max}$ from 20 to 50 does not consistently improve RMSE but leads to higher NLL across all data sets, regardless of whether $\mathbf{\Lambda}_{M_d}$ is on or off. However, the increase in NLL is consistently smaller when $\mathbf{\Lambda}_{M_d}$ is on. For example, in the Energy data set, RMSE slightly improves with larger $M_{\max}$ and $\mathbf{\Lambda}_{M_d}$ on, but NLL worsens without regularization. This suggests that $\mathbf{\Lambda}_{M_d}$ helps prevent overfitting by controlling complexity through row sparsity.

4.4 Predictive Performance with baseline

We considered six UCI data sets in order to compare the predictive performance of BTN-Kernel machines with GP, T-KRR and SP-BTN. The data sets are split into random training and test sets with 90% and 10% of the data, respectively. This splitting process is repeated 10 times and the average test performance is reported. For regression tasks, each data set is normalized within the training set so that the features and target values have zero mean and unit variance. The normalization on the targets is removed for prediction. For binary classification tasks, only the features are normalized based on the training data, and inference is performed by considering the sign of the model response (Suykens and Vandewalle, 1999).

For the GP models, we followed the procedure described in Wesel and Batselier (2021), while for the SP-BTN models, we reported the average test RMSE values from the original paper, as the implementation was not available. NLL values or other uncertainty quantification metrics for SP-BTN were not reported in the original work, so we have excluded them from our comparison. For a fair comparison between T-KRR and BTN-Kernel machines, we used polynomial feature mapping with unit norm also for T-KRR. Since BTN-Kernel machines retain the number of rows, we used the same feature dimensionality, denoted as

					RMSE				NLL	
Dataset	N	D	M	R_{eff}	**GP**	**BTN-Kernels**	**T-KRR**	**SP-BTN**	**GP**	**BTN-Kernels**
Yacht	308	6	20	5.2 ± 0.6	0.402 ± 0.131	**0.379 ± 0.132**	0.894 ± 0.478	0.506 ± 0.091	**0.105 ± 0.336**	0.598 ± 0.782
Energy	768	9	20	10.1 ± 1.3	1.296 ± 0.290	**0.456 ± 0.069**	0.641 ± 0.135	0.549 ± 0.200	1.680 ± 0.223	**1.530 ± 0.271**
Concrete	1030	8	20	5.3 ± 0.5	5.565 ± 0.520	5.452 ± 1.242	**4.959 ± 0.620**	5.500 ± 0.230	**3.078 ± 0.080**	3.387 ± 0.171
Airfoil	1503	5	20	6.2 ± 0.4	2.293 ± 0.199	**1.723 ± 0.151**	1.806 ± 0.144	-	**2.281 ± 0.084**	2.865 ± 0.152
Spambase	4601	57	30	8.6 ± 1.2	0.095 ± 0.016	0.075 ± 0.015	**0.066 ± 0.012**	-	0.720 ± 0.146	**0.499 ± 0.047**
Adult	45222	96	40	6.3 ± 0.5	N/A	**0.144 ± 0.005**	0.1658 ± 0.0069	-	N/A	**0.674 ± 0.003**

Table 4: Average predictive RMSE (for regression) or misclassification error (for classification) and NLL with standard errors. BTN-Kernel machines and T-KRR achieve the lowest overall errors, with BTN-Kernel machines matching or outperforming GP in both prediction accuracy and uncertainty estimation, especially on higher-dimensional datasets, while also remaining scalable to large datasets like Adult.

M in Table 4, for both models. For T-KRR, the rank R was set to 10 for all data sets, and for BTN-Kernel machines, we initialized the model with $R_{\max} = 25$ for all data sets except the Adult data set. Due to the high number of features (D) in the Adult data set, we set $R_{\max} = 10$ to maintain computational efficiency.

All BTN-Kernel machines models are initialized as described in Section 3.2.5. To encourage low-rank solutions, we set the hyperparameters to $c_0 = 10^{-5}$ and $d_0 = 10^{-6}$, with the exception of the Adult data set, for which $R_{\max}$ is initialized to a lower value. The precision hyperparameters are initialized with a slightly more informative prior, using $a_0 = 10^{-2}$ and $b_0 = 10^{-3}$ for all data sets, except for the Airfoil and Concrete data sets.

Table 4 shows the average RMSE or misclassification rate for each method, along with the average NLL for GP and BTN-Kernel machines. The best results for each data set are highlighted in bold. Overall, BTN-Kernel machines and T-KRR have the lowest errors, with BTN-Kernel machines achieving the lowest RMSE on 4 out of 6 data sets. SP-BTN was tested only on three low-dimensional data sets, where its performance was similar to other methods, but its effectiveness on higher-dimensional data sets is not known. Although GP performs slightly worse for some data sets like Energy, its RMSE and misclassification rate remain close to the other methods. However, GP is not suitable for very large data sets, such as the Adult data set. All methods perform well on the test data sets, with BTN-Kernel machines and T-KRR generally doing better. Both methods are scalable and suitable for high-dimensional data, but unlike BTN-Kernel machines, T-KRR does not provide uncertainty estimates.

GP and BTN-Kernel machines both offer uncertainty estimates, which is an advantage over the other methods. GP shows slightly better average test NLL on the Yacht, Concrete, and Airfoil data sets, while BTN-Kernel machines performs better on Energy and Spambase. However, the differences between them are small overall. This indicates that BTN-Kernel machines performs similarly to GP, with some advantage on higher-dimensional data sets for both prediction accuracy and uncertainty estimation. For the Adult data set, which is large and high-dimensional, BTN-Kernel machines achieves a considerable level of performance performance with a misclassification rate of 14.4% and an NLL score of 0.674 ± 0.003.

5 Conclusion

We have presented BTN Kernel Machines, a probabilistic extension of tensor network-based kernel methods. By placing sparsity-inducing hierarchical priors on the tensor network factors, the model can automatically infer the appropriate tensor rank and feature dimensions. This enables the selection of relevant features, enhances interpretability, and helps prevent overfitting by controlling model complexity. To make Bayesian learning tractable, we employed mean-field variational inference, resulting in a Bayesian ALS algorithm with the same computational complexity as its conventional deterministic counterpart. Hence, BTN Kernel Machines provide uncertainty quantification without incurring additional computational cost. Numerical experiments demonstrate the model's effectiveness in automatic complexity inference, overfitting prevention, and interpretability. Empirical evaluations on six data sets demonstrate that BTN Kernel Machines achieve competitive performance in both predictive accuracy and uncertainty quantification. The results further show that the model effectively scales to large data sets with high-dimensional inputs, delivering state-of-the-art performance while capturing predictive uncertainty. Potential future work can be the implementation of Tensor Train Kernel machines described in Section 3.5 and the comparison with the CPD case. Furthermore, instead of sparsity-inducing hierarchical priors, alternative prior structures on tensor network factors can be explored, considering both CPD and TT decomposed model weights.

Acknowledgments and Disclosure of Funding

This publication is part of the project Sustainable learning for Artificial Intelligence from noisy large-scale data (with project number VI.Vidi.213.017) which is financed by the Dutch Research Council (NWO).

Appendix A.

1. Proof of Theorem 3

Derivation of the term in (20) only for one sample. We make use of the multi-linearity property of the CPD and rely on re-ordering the summations:

$$\begin{aligned}
\boldsymbol{\varphi}(x_n)^T\mathbf{w} &= \left\langle \sum_{r=1}^{R} \mathbf{w}_r^{(1)} \otimes \mathbf{w}_r^{(2)} \otimes \cdots \otimes \mathbf{w}_r^{(D)}, \boldsymbol{\varphi}^{(1)} \otimes \boldsymbol{\varphi}^{(2)} \otimes \cdots \otimes \boldsymbol{\varphi}^{(D)} \right\rangle \\
&= \sum_{r=1}^{R} \left\langle \mathbf{w}_r^{(1)} \otimes \mathbf{w}_r^{(2)} \otimes \cdots \otimes \mathbf{w}_r^{(D)}, \boldsymbol{\varphi}^{(1)} \otimes \boldsymbol{\varphi}^{(2)} \otimes \cdots \otimes \boldsymbol{\varphi}^{(D)} \right\rangle \\
&= \sum_{r=1}^{R} \sum_{m_1=1}^{M_1} \cdots \sum_{m_d=1}^{M_D} w_{m_1 r}^{(1)} \varphi_{m_1}^{(1)} \ldots w_{m_d r}^{(d)} \varphi_{m_d}^{(d)} \ldots w_{m_D r}^{(D)} \varphi_{m_D}^{(D)} \\
&= \sum_{r=1}^{R} \sum_{m_1=1}^{M_1} w_{m_1 r}^{(1)} \varphi_{m_1}^{(1)} \cdots \sum_{m_d=1}^{M_d} w_{m_d r}^{(d)} \varphi_{m_d}^{(d)} \cdots \sum_{m_D=1}^{M_D} w_{m_D r}^{(D)} \varphi_{m_D}^{(D)} \\
&= \sum_{r=1}^{R} \prod_{d=1}^{D} \sum_{m_d=1}^{M_d} w_{m_d r}^{(d)} \varphi_{m_d}^{(d)} \\
&= \sum_{r=1}^{R} \sum_{m_d=1}^{M_d} w_{m_d r}^{(d)} \left(\varphi_{m_d}^{(d)} \prod_{k \neq d}^{D} \sum_{m_k=1}^{M_k} w_{m_k r}^{(k)} \varphi_{m_k}^{(k)} \right) \\
&= \sum_{r=1}^{R} \mathbf{w}_r^{(d)^T} \left(\boldsymbol{\varphi}^{(d)} \prod_{k \neq d}^{D} \mathbf{w}_{.r}^{(k)^T} \boldsymbol{\varphi}^{(k)} \right) \\
&= \mathrm{vec}(\mathbf{W}^{(d)})^T \left(\left(\circledast_{k \neq d} \mathbf{W}^{(k)^T} \boldsymbol{\varphi}^{(k)} \right) \otimes \boldsymbol{\varphi}^{(d)} \right)
\end{aligned}$$

and for $n = 1...N$:

$$\begin{aligned}
\boldsymbol{\Phi}^T\mathbf{w} &= \mathrm{vec}(\mathbf{W}^{(d)})^T \left(\left(\circledast_{k \neq d} \mathbf{W}^{(k)^T} \boldsymbol{\Phi}^{(k)} \right) \odot \boldsymbol{\Phi}^{(d)} \right) \\
&= \mathrm{vec}(\mathbf{W}^{(d)})^T \mathbf{G}^{(d)}.
\end{aligned} \tag{44}$$

Notice that if we do not isolate the d-th factor matrix, the data fitting term can also be written as

$$\boldsymbol{\Phi}^T\mathbf{w} = \mathbf{1}_R^T \left(\circledast_{d=1}^{D} \mathbf{W}^{(d)^T} \boldsymbol{\Phi}^{(d)} \right). \qquad \blacksquare$$

2. The log of the joint distribution

$$
\begin{aligned}
l(\Theta) &= \sum_{n=1}^{N}\left(\frac{1}{2}\ln\tau - \frac{\tau}{2}\left(y_n - \boldsymbol{\varphi}(x_n)^T\mathbf{w}\right)^2\right) + \sum_{d=1}^{D}\sum_{r}\sum_{m_d}\left(\frac{1}{2}\ln|\lambda_r\lambda_{m_d}| - \frac{1}{2}(w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)})\right)\\
&+ \sum_{r}\left((c_0^r-1)\ln\lambda_r - d_0^r\lambda_r\right) + \sum_{d}^{D}\sum_{m_d}\left((g_0^{dm_d}-1)\ln\lambda_{m_d}^d - g_0^{dm_d}\lambda_{m_d}^d\right) + (a_0-1)\ln\tau - b_0\tau + \text{const}\\
&= \frac{N}{2}\ln\tau - \frac{\tau}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{\varphi}(x_n)^T\mathbf{w}\right)^2 + \frac{\sum_d M_d}{2}\ln|\boldsymbol{\Lambda}_R| + \frac{R}{2}\sum_{d=1}^{D}\ln|\boldsymbol{\Lambda}_{M_d}| - \frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d}w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)}\\
&+ \sum_{r}\left((c_0^r-1)\ln\lambda_r - d_0^r\lambda_r\right) + \sum_{d}^{D}\sum_{m_d}\left((g_0^{dm_d}-1)\ln\lambda_{m_d}^d - g_0^{dm_d}\lambda_{m_d}^d\right) + (a_0-1)\ln\tau - b_0\tau + \text{const}\\
&= -\frac{\tau}{2}\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2 - \frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d}w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)} + \left(\frac{N}{2} + a_0 - 1\right)\ln\tau + \sum_{r}\left(\frac{\sum_d M_d}{2} + (c_0^r - 1)\right)\ln\lambda_r\\
&+ \sum_{d}^{D}\sum_{m_d}\left(\frac{R}{2} + (g_0^{dm_d}-1)\right)\ln\lambda_{m_d}^d - \sum_{d}^{D}\sum_{m_d}h_0^{dm_d}\lambda_{m_d}^d - \sum_{r}d_0^r\lambda_r - b_0\tau + \text{const}
\end{aligned}
\tag{45}
$$

This form will be used frequently in variational Bayesian inference, because the inference of each Θ_j can be done by

$$
\ln q_j(\Theta_j) = \mathbb{E}_{q(\Theta\setminus\Theta_j)}[\ln p(\mathbf{y},\boldsymbol{\Theta})] + \text{const} \tag{46}
$$

3. Factor matrix update

$$
\begin{aligned}
\ln q\left(\text{vec}(\mathbf{W}^{(d)})\right) &= \mathbb{E}_{q(\Theta\setminus\mathbf{W}^{(d)})}[\ln p(\mathbf{y},\{\mathbf{W}^{(d)}\},\boldsymbol{\lambda}_{M_d},\boldsymbol{\lambda}_R,\tau)] + \text{const}\\
&= \mathbb{E}\left[-\frac{\tau}{2}\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2 - \frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d}w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)}\right] + \text{const}\\
&= \mathbb{E}\left[-\frac{\tau}{2}\left(\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\right)^2 + -\frac{1}{2}\,\text{vec}(\mathbf{W}^{(d)})^T\left(\boldsymbol{\Lambda}_R\otimes\boldsymbol{\Lambda}_{M_d}\right)\text{vec}(\mathbf{W}^{(d)})\right] + \text{const}\\
&= \mathbb{E}\left[-\frac{\tau}{2}\left(\mathbf{y} - \text{vec}(\mathbf{W}^{(d)})^T\mathbf{G}^{(d)}\right)^2 + -\frac{1}{2}\,\text{vec}(\mathbf{W}^{(d)})^T\left(\boldsymbol{\Lambda}_R\otimes\boldsymbol{\Lambda}_{M_d}\right)\text{vec}(\mathbf{W}^{(d)})\right] + \text{const}\\
&= \mathbb{E}\left[-\frac{\tau}{2}\,\text{vec}(\mathbf{W}^{(d)})^T\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\,\text{vec}(\mathbf{W}^{(d)}) - \frac{1}{2}\,\text{vec}(\mathbf{W}^{(d)})^T\left(\boldsymbol{\Lambda}_R\otimes\boldsymbol{\Lambda}_{M_d}\right)\text{vec}(\mathbf{W}^{(d)}) + \tau\,\text{vec}(\mathbf{W}^{(d)})^T\mathbf{G}^{(d)}\mathbf{y}\right]\\
&+ \text{const}\\
&= -\frac{1}{2}\,\text{vec}(\mathbf{W}^{(d)})^T\left(\mathbb{E}\left[\tau\right]\mathbb{E}\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right] + \mathbb{E}\left[\boldsymbol{\Lambda}_R\otimes\boldsymbol{\Lambda}_{M_d}\right]\right)\text{vec}(\mathbf{W}^{(d)}) + \text{vec}(\mathbf{W}^{(d)})^T\mathbb{E}\left[\tau\right]\mathbb{E}\left[\mathbf{G}^{(d)}\right]\mathbf{y}\\
&+ \text{const}
\end{aligned}
\tag{47}
$$

4. Proof of Theorem 4

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{G}^{(d)}\mathbf{G}^{(d)T}\right] &= \mathbb{E}\left[\left(\left(\underset{k\neq d}{\circledast}\mathbf{W}^{(k)^T}\boldsymbol{\Phi}^{(k)}\right)\odot\boldsymbol{\Phi}^{(d)}\right)\left(\left(\underset{k\neq d}{\circledast}\mathbf{W}^{(k)^T}\boldsymbol{\Phi}^{(k)}\right)\odot\boldsymbol{\Phi}^{(d)}\right)^T\right] \\
&= \sum_{n=1}^{N}\mathbb{E}\left[\left(\left(\prod_{k\neq d}^{D}\mathbf{W}^{(k)^T}\boldsymbol{\varphi}^{(k)}\right)\otimes\boldsymbol{\varphi}^{(d)}\right)\left(\left(\prod_{k\neq d}^{D}\mathbf{W}^{(k)^T}\boldsymbol{\varphi}^{(k)}\right)\otimes\boldsymbol{\varphi}^{(d)}\right)^T\right] \\
&= \sum_{n=1}^{N}\mathbb{E}\left[\left(\left(\prod_{k\neq d}^{D}\sum_{m_k=1}^{M_k}\mathbf{w}^{(k)}_{m_k\cdot}\varphi^{(k)}_{m_k}\right)\otimes\boldsymbol{\varphi}^{(d)}\right)\left(\left(\prod_{k\neq d}^{D}\sum_{j_k=1}^{M_k}\mathbf{w}^{(k)}_{j_k\cdot}\varphi^{(k)}_{j_k}\right)\otimes\boldsymbol{\varphi}^{(d)}\right)^T\right] \\
&= \sum_{n=1}^{N}\mathbb{E}\left[\mathcal{R}\left\{\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)\prod_{k\neq d}^{D}\sum_{m_k=1}^{M_k}\sum_{j_k=1}^{M_k}\varphi^{(k)}_{m_k}\varphi^{(k)}_{j_k}\mathbf{w}^{(k)}_{m_k\cdot}\otimes\mathbf{w}^{(k)}_{j_k\cdot}\right\}_{M_dR\times M_dR}\right] \\
&= \sum_{n=1}^{N}\mathcal{R}\left\{\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)\prod_{k\neq d}^{D}\sum_{m_k=1}^{M_k}\sum_{j_k=1}^{M_k}\varphi^{(k)}_{m_k}\varphi^{(k)}_{j_k}\mathbb{E}\left[\mathbf{w}^{(k)}_{m_k\cdot}\otimes\mathbf{w}^{(k)}_{j_k\cdot}\right]\right\}_{M_dR\times M_dR} \\
&= \sum_{n=1}^{N}\mathcal{R}\left\{\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)\prod_{k\neq d}^{D}\left(\boldsymbol{\varphi}^{(k)}\otimes\boldsymbol{\varphi}^{(k)}\right)^T\mathbb{E}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]\right\}_{M_dR\times M_dR} \\
&= \mathcal{R}\left\{\left(\boldsymbol{\Phi}^{(d)}\odot\boldsymbol{\Phi}^{(d)}\right)\prod_{k\neq d}^{D}\left(\boldsymbol{\Phi}^{(k)}\odot\boldsymbol{\Phi}^{(k)}\right)^T\mathbb{E}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]\right\}_{M_dR\times M_dR} \\
&= \mathcal{R}\left\{\left(\boldsymbol{\Phi}^{(d)}\odot\boldsymbol{\Phi}^{(d)}\right)\prod_{k\neq d}^{D}\left(\boldsymbol{\Phi}^{(k)}\odot\boldsymbol{\Phi}^{(k)}\right)^T\left(\mathbb{E}\left[\mathbf{W}^{(k)}\right]\otimes\mathbb{E}\left[\mathbf{W}^{(k)}\right]+\mathrm{Var}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]\right)\right\}_{M_dR\times M_dR} \\
&= \mathcal{R}\left\{\left(\boldsymbol{\Phi}^{(d)}\odot\boldsymbol{\Phi}^{(d)}\right)\overset{D}{\underset{k\neq d}{\circledast}}\left(\boldsymbol{\Phi}^{(k)}\odot\boldsymbol{\Phi}^{(k)}\right)^T\left(\tilde{\mathbf{W}}^{(k)}\otimes\tilde{\mathbf{W}}^{(k)}+\mathcal{R}\left\{\boldsymbol{\Sigma}^{(k)}\right\}_{M_k^2\times R^2}\right)\right\}_{M_dR\times M_dR},
\end{aligned}
\tag{48}
$$

In above expression the term $\mathrm{Var}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]$ can be evaluated by reshaping the $\boldsymbol{\Sigma}^{(k)}\in\mathbb{R}^{MR\times MR}$ into a size $M^2\times R^2$

$$
\mathrm{Var}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]=\mathcal{R}\left\{\boldsymbol{\Sigma}^{(k)}\right\}_{M_k^2\times R^2} \tag{49}
$$

5. The variational posterior distribution of hyperparameter $\boldsymbol{\Lambda}_R$

$$\begin{aligned}
\ln\ q(\boldsymbol{\lambda}_R) &= \mathbb{E}_{q(\Theta\setminus\boldsymbol{\lambda})}[\ln p(\mathbf{y},\{\mathbf{W}^{(d)}\},\boldsymbol{\lambda}_{M_d},\boldsymbol{\lambda}_R,\tau)] + \text{const} \\
&= \mathbb{E}\left[-\frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d} w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)} + \sum_r\left(\frac{\sum_d M_d}{2}+(c_0^r-1)\right)\ln\lambda_r - \sum_r d_0^r\lambda_r\right] + \text{const} \\
&= \mathbb{E}\left[\sum_r\left\{-\frac{1}{2}\sum_d\sum_{m_d}(w_{m_dr}^{(d)}\lambda_{m_d}w_{m_dr}^{(d)})\lambda_r + \left(c_0^r+\frac{\sum_d M_d}{2}-1\right)\ln\lambda_r - d_0^r\lambda_r\right\}\right] + \text{const} \\
&= \mathbb{E}\left[\sum_r\left\{\left(c_0^r+\frac{\sum_d M_d}{2}-1\right)\ln\lambda_r - \left(d_0^r+\frac{1}{2}\sum_d\sum_{m_d} w_{m_dr}^{(d)}\lambda_{m_d}w_{m_dr}^{(d)}\right)\lambda_r\right\}\right] + \text{const} \\
&= \mathbb{E}\left[\sum_r\left\{\left(c_0^r+\frac{\sum_d M_d}{2}-1\right)\ln\lambda_r - \left(d_0^r+\frac{1}{2}\sum_d \mathbf{w}_r^{(d)T}\boldsymbol{\Lambda}_{M_d}\mathbf{w}_r^{(d)}\right)\lambda_r\right\}\right] + \text{const} \\
&= \sum_r\left\{\left(c_0^r+\frac{\sum_d M_d}{2}-1\right)\ln\lambda_r - \left(d_0^r+\frac{1}{2}\sum_d \mathbb{E}\left[\mathbf{w}_r^{(d)T}\boldsymbol{\Lambda}_{M_d}\mathbf{w}_r^{(d)}\right]\right)\lambda_r\right\} + \text{const}
\end{aligned} \tag{50}$$

Thus we observe that the posterior is a Gamma distribution with updated parameters . In above expression, the posterior expectation term can be computed by

$$\begin{aligned}
\mathbb{E}_q\left[\mathbf{w}_{r.}^{(d)T}\boldsymbol{\Lambda}_{M_d}\mathbf{w}_{r.}^{(d)}\right] &= \mathbb{E}_q\left[\sum_{m_d}\lambda_{m_d}\left(w_{m_dr}^{(d)}\right)^2\right] = \sum_{m_d}\lambda_{m_d}\mathbb{E}_q\left[\left(w_{m_dr}^{(d)}\right)^2\right] \\
&= \sum_{m_d}\lambda_{m_d}\left\{\left(\mathbb{E}_q\left[w_{m_dr}^{(d)}\right]\right)^2 + \text{Var}\left(w_{m_dr}^{(d)}\right)\right\} \\
&= \tilde{\mathbf{w}}_r^{(d)T}\boldsymbol{\Lambda}_{M_d}\tilde{\mathbf{w}}_r^{(d)} + \boldsymbol{\lambda}_{M_d}^T\ \text{Var}\left(\mathbf{w}_r^{(d)}\right).
\end{aligned} \tag{51}$$

6. The variational posterior distribution of hyperparameter $\boldsymbol{\lambda}_{M_d}$

$$\begin{aligned}
\ln\ q(\boldsymbol{\lambda}_{M_d}) &= \mathbb{E}_{q(\Theta\setminus\boldsymbol{\lambda})}[\ln p(\mathbf{y},\{\mathbf{W}^{(d)}\},\boldsymbol{\lambda}_{M_d},\boldsymbol{\lambda}_R,\tau)] + \text{const} \\
&= \mathbb{E}\left[-\frac{1}{2}\sum_{d=1}^{D}\sum_{r}\sum_{m_d} w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)} + \sum_d^D\sum_{m_d}\left(\frac{R}{2}+(g_0^{dm_d}-1)\right)\ln\lambda_{m_d}^d - \sum_d^D\sum_{m_d} h_0^{dm_d}\lambda_{m_d}^d\right] + \text{const} \\
&= \mathbb{E}\left[-\frac{1}{2}\sum_{m_d}\sum_r(w_r^{(d)}\lambda_{m_d}w_{m_dr}^{(d)})\lambda_{m_d} + \sum_{m_d}\left(\frac{R}{2}+(g_0^{dm_d}-1)\right)\ln\lambda_{m_d} - \sum_{m_d} h_0^{dm_d}\lambda_{m_d}\right] + \text{const} \\
&= \mathbb{E}\left[\sum_{m_d}\left\{\left(g_0^{dm_d}+\frac{R}{2}-1\right)\ln\lambda_{m_d} - \left(h_0^{dm_d}+\frac{1}{2}\mathbf{w}_{m_d}^{(d)T}\boldsymbol{\Lambda}_R\mathbf{w}_{m_d}^{(d)}\right)\lambda_{m_d}\right\}\right] + \text{const} \\
&= \sum_{m_d}\left\{\left(g_0^{dm_d}+\frac{R}{2}-1\right)\ln\lambda_{m_d} - \left(h_0^{dm_d}+\frac{1}{2}\mathbb{E}\left[\mathbf{w}_{m_d}^{(d)T}\boldsymbol{\Lambda}_R\mathbf{w}_{m_d}^{(d)}\right]\right)\lambda_{m_d}\right\} + \text{const}
\end{aligned} \tag{52}$$

Thus we observe that the posterior is a Gamma distribution with updated parameters . In above expression, the posterior expectation term can be computed by

$$\begin{aligned}
\mathbb{E}_q\left[\mathbf{w}_{m_d}^{(d)T}\boldsymbol{\Lambda}_R\mathbf{w}_{m_d}^{(d)}\right] &= \mathbb{E}_q\left[\sum_r\lambda_r\left(w_{m_dr}^{(d)}\right)^2\right] = \sum_r\lambda_r\mathbb{E}_q\left[\left(w_{m_dr}^{(d)}\right)^2\right] \\
&= \sum_r\lambda_r\left\{\left(\mathbb{E}_q\left[w_{m_dr}^{(d)}\right]\right)^2 + \text{Var}\left(w_{m_dr}^{(d)}\right)\right\} \\
&= \tilde{\mathbf{w}}_{m_d}^{(d)T}\boldsymbol{\Lambda}_R\tilde{\mathbf{w}}_{m_d}^{(d)} + \text{Var}\left(\mathbf{w}_{m_d}^{(d)}\right)^T\boldsymbol{\lambda}_R.
\end{aligned} \tag{53}$$

7. The variational posterior distribution of hyperparameter τ

$$\begin{aligned}\ln\ q(\tau) &= \mathbb{E}_{q(\Theta\setminus\tau)}[\ln\ p(\mathbf{y},\{\mathbf{W}^{(d)}\},\boldsymbol{\lambda}_{M_d},\boldsymbol{\lambda}_R,\tau)]+\text{const}\\ &= \mathbb{E}\left[-\frac{\tau}{2}\|\mathbf{y}-\boldsymbol{\Phi}^T\mathbf{w}\|_F^2+\left(\frac{N}{2}+a_0-1\right)\ln\tau-b_0\tau\right]+\text{const}\\ &= \left(\frac{N}{2}+a_0-1\right)\ln\tau-\left(b_0+\frac{1}{2}\mathbb{E}_q\left[\|\mathbf{y}-\boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right]\right)\tau+\text{const}\end{aligned} \tag{54}$$

8. Proof of Theorem 6

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right] &= \mathbb{E}\left[\|\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}}\mathbf{W}^{(d)}\boldsymbol{\Phi}^{(d)}\right)\|_F^2\right]\\ &= \mathbb{E}\left[\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}}\mathbf{W}^{(d)}\boldsymbol{\Phi}^{(d)}\right)\left(\overset{D}{\underset{d=1}{\circledast}}\mathbf{W}^{(d)}\boldsymbol{\Phi}^{(d)}\right)^T\mathbf{1}_R\right]\\ &= \mathbb{E}\left[\sum_n\sum_{r_1}\sum_{r_2}\prod_{d=1}^D\sum_{m_d}\sum_{j_d}\varphi_{m_d}^{(d)}\varphi_{j_d}^{(d)}w_{m_dr_1}^{(d)}w_{j_dr_2}^{(d)}\right]\\ &= \mathbb{E}\left[\sum_n\sum_{r_1}\sum_{r_2}\prod_{d=1}^D\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)^T\left(\mathbf{w}_{\cdot r_1}^{(d)}\otimes\mathbf{w}_{\cdot r_2}^{(d)}\right)\right]\\ &= \sum_n\sum_{r_1}\sum_{r_2}\prod_{d=1}^D\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)^T\mathbb{E}\left[\mathbf{w}_{\cdot r_1}^{(d)}\otimes\mathbf{w}_{\cdot r_2}^{(d)}\right]\\ &= \sum_n\prod_{d=1}^D\left\{\left(\boldsymbol{\varphi}^{(d)}\otimes\boldsymbol{\varphi}^{(d)}\right)^T\mathbb{E}\left[\mathbf{W}^{(d)}\otimes\mathbf{W}^{(d)}\right]\right\}\mathbf{1}_{R^2}\\ &= \mathbf{1}_N^T\prod_{d=1}^D\left\{\left(\boldsymbol{\Phi}^{(d)}\odot\boldsymbol{\Phi}^{(d)}\right)^T\mathbb{E}\left[\mathbf{W}^{(d)}\otimes\mathbf{W}^{(d)}\right]\right\}\mathbf{1}_{R^2}\\ &= \mathbf{1}_N^T\prod_{d=1}^D\left\{\left(\boldsymbol{\Phi}^{(d)}\odot\boldsymbol{\Phi}^{(d)}\right)^T\left(\mathbb{E}\left[\mathbf{W}^{(k)}\right]\otimes\mathbb{E}\left[\mathbf{W}^{(k)}\right]+\text{Var}\left[\mathbf{W}^{(k)}\otimes\mathbf{W}^{(k)}\right]\right)\right\}\mathbf{1}_{R^2}\\ &= \mathbf{1}_N^T\left(\overset{D}{\underset{k\neq d}{\circledast}}\left(\boldsymbol{\Phi}^{(k)}\odot\boldsymbol{\Phi}^{(k)}\right)^T\left(\tilde{\mathbf{W}}^{(k)}\otimes\tilde{\mathbf{W}}^{(k)}+\mathcal{R}\left\{\boldsymbol{\Sigma}^{(k)}\right\}_{M_k^2\times R^2}\right)\right)\mathbf{1}_{R^2}\end{aligned} \tag{55}$$

■

9. Posterior expectation of model error or residual

$$\begin{aligned}&\mathbb{E}\left[\|\mathbf{y}-\boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right]\\ =&\mathbb{E}\left[\|\mathbf{y}\|_F^2-2\mathbf{y}^T\boldsymbol{\Phi}^T\mathbf{w}+\|\boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right]\\ =&\|\mathbf{y}\|_F^2-2\,\mathbf{y}^T\left(\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}}\mathbf{E}\left[\mathbf{W}^{(d)}\right]^T\boldsymbol{\Phi}^{(d)}\right)\right)+\mathbb{E}\left[\|\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}}\mathbf{W}^{(d)}\boldsymbol{\Phi}^{(d)}\right)\|_F^2\right]\\ =&\|\mathbf{y}\|_F^2-2\,\mathbf{y}^T\left(\mathbf{1}_R^T\left(\overset{D}{\underset{d=1}{\circledast}}\tilde{\mathbf{W}}^{(d)T}\boldsymbol{\Phi}^{(d)}\right)\right)+\mathbf{1}_N^T\left(\overset{D}{\underset{k\neq d}{\circledast}}\left(\boldsymbol{\Phi}^{(k)}\odot\boldsymbol{\Phi}^{(k)}\right)^T\left(\tilde{\mathbf{W}}^{(k)}\otimes\tilde{\mathbf{W}}^{(k)}+\mathcal{R}\left\{\boldsymbol{\Sigma}^{(k)}\right\}_{M_k^2\times R^2}\right)\right)\mathbf{1}_{R^2}\end{aligned} \tag{56}$$

10. Lower bound model evidence

$$
\begin{aligned}
L(q) &= \mathbb{E}_{q(\Theta)}[\ln p(\mathbf{y}, \boldsymbol{\Theta})] + H(q(\Theta)) \\
&= \mathbb{E}_{q(\{\mathbf{W}^{(d)}\},\tau)}\left[\ln\ p(\mathbf{y} \mid \{\mathbf{W}^{(d)}\}_{d=1}^{D}, \tau^{-1})\right] + \mathbb{E}_{q(\{\mathbf{W}^{(d)}\},\{\boldsymbol{\lambda}_{M_d}\},\boldsymbol{\lambda}_R,)}\left[\sum_{d=1}^{D}\ln\ p(\mathbf{W}^{(d)} \mid \boldsymbol{\lambda}_R, \boldsymbol{\lambda}_{M_d}))\right] \\
&+ \mathbb{E}_{q(\{\boldsymbol{\lambda}_{M_d}\})}\left[\sum_{d=1}^{D}\ln\ p(\boldsymbol{\lambda}_{M_d})\right] + \mathbb{E}_{q(\boldsymbol{\lambda}_R)}\left[\ln\ p(\boldsymbol{\lambda}_R)\right] + \mathbb{E}_{q(\tau)}\left[\ln\ p(\tau)\right] - \mathbb{E}_{q(\{\mathbf{W}^{(d)}\}}\left[\sum_{d=1}^{D}\ln\ q(\mathbf{W}^{(d)})\right] \\
&- \mathbb{E}_{q(\{\boldsymbol{\lambda}_{M_d}\})}\left[\sum_{d=1}^{D}\ln\ q(\boldsymbol{\lambda}_{M_d})\right] - \mathbb{E}_{q(\boldsymbol{\lambda}_R)}\left[\ln\ q(\boldsymbol{\lambda}_R)\right] - \mathbb{E}_{q(\tau)}\left[\ln\ q(\tau)\right]
\end{aligned}
\tag{57}
$$

All expectations above are with respect to the posterior distribution q. The first term is the expected log-likelihood; the next four terms are the expected log-priors over $\{\mathbf{W}^{(d)}\}_{d=1}^{D}$, $\{\boldsymbol{\lambda}_{M_d}\}_{d=1}^{D}$, $\boldsymbol{\lambda}_R$, and τ, respectively. The final four terms represent the entropies of the posterior distributions over the factor matrices and the hyperparameters. Each term can be computed by

$$
\begin{aligned}
\mathbb{E}_q\left[\ln\ p(\mathbf{y} \mid \{\mathbf{W}^{(d)}\}_{d=1}^{D}, \tau^{-1})\right] &= -\frac{N}{2}\ln\,(2\pi) + \frac{N}{2}\mathbb{E}_q\left[\ln\ \tau\right] - \frac{1}{2}\mathbb{E}_q\left[\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right] \\
&= -\frac{N}{2}\ln\,(2\pi) + \frac{N}{2}(\psi(a_N) - \ln\ b_N) - \frac{a_N}{2b_N}\mathbb{E}_q\left[\|\mathbf{y} - \boldsymbol{\Phi}^T\mathbf{w}\|_F^2\right]
\end{aligned}
\tag{58}
$$

where N denotes the number of observations. $\mathbb{E}_q\left[\|\mathbf{y} - \boldsymbol{\Phi}^{\mathbf{w}}\|_F^2\right]$ can be computed as show in Theorem 3.

$$
\begin{aligned}
&\mathbb{E}_q\Big[\sum_{d=1}^{D}\ln p(\mathbf{W}^{(d)} \mid \boldsymbol{\lambda}_R, \boldsymbol{\lambda}_{M_d})\Big] \\
&= \mathbb{E}_q\Big[\sum_{d=1}^{D}\sum_{r}\sum_{m_d}\Big(-\frac{\ln(2\pi)}{2} + \frac{1}{2}\ln|\lambda_r\lambda_{m_d}| - \frac{1}{2}w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)}\Big)\Big] \\
&= \mathbb{E}_q\Big[\sum_{d=1}^{D}\sum_{r}\sum_{m_d}\Big(-\frac{\ln(2\pi)}{2} + \frac{1}{2}\big(\ln|\lambda_r| + \ln|\lambda_{m_d}|\big) - \frac{1}{2}w_{m_dr}^{(d)}\lambda_r\lambda_{m_d}w_{m_dr}^{(d)}\Big)\Big] \\
&= \sum_d\Big\{-\frac{RM_d}{2}\ln(2\pi) + \frac{M_d}{2}\sum_r\mathbb{E}_q[\ln\lambda_r] + \frac{R}{2}\sum_{m_d}\mathbb{E}_q[\ln\lambda_{m_d}] - \frac{1}{2}\mathbb{E}_q\Big[\operatorname{vec}(\mathbf{W}^{(d)})^T\big(\boldsymbol{\Lambda}_R\otimes\boldsymbol{\Lambda}_{M_d}\big)\operatorname{vec}(\mathbf{W}^{(d)})\Big]\Big\} \\
&= -\frac{R\sum_d M_d}{2}\ln(2\pi) + \frac{\sum_d M_d}{2}\sum_r\mathbb{E}_q[\ln\lambda_r] + \frac{R}{2}\sum_d\sum_{m_d}\mathbb{E}_q[\ln\lambda_{m_d}] \\
&- \frac{1}{2}\sum_d\Big\{\operatorname{Tr}\Big(\mathbb{E}_q[\boldsymbol{\Lambda}_R]\otimes\mathbb{E}_q[\boldsymbol{\Lambda}_{M_d}]\operatorname{Var}\big(\operatorname{vec}(\mathbf{W}^{(d)})\big)\Big) \quad + \mathbb{E}_q\big[\operatorname{vec}(\mathbf{W}^{(d)})^T\big]\big(\mathbb{E}_q[\boldsymbol{\Lambda}_R]\otimes\mathbb{E}_q[\boldsymbol{\Lambda}_{M_d}]\big)\mathbb{E}_q\big[\operatorname{vec}(\mathbf{W}^{(d)})\big]\Big\} \\
&= -\frac{R\sum_d M_d}{2}\ln(2\pi) + \frac{\sum_d M_d}{2}\sum_r(\psi(c_N^r) - \ln d_N^r) + \frac{R}{2}\sum_d\sum_{m_d}(\psi(g_N^{m_d}) - \ln h_N^{m_d}) \\
&\quad - \frac{1}{2}\sum_d\Big\{\operatorname{Tr}\Big(\big(\tilde{\boldsymbol{\Lambda}}_R\otimes\tilde{\boldsymbol{\Lambda}}_{M_d}\big)\mathbf{V}^{(d)}\Big)\Big\} - \frac{1}{2}\sum_d\Big\{\operatorname{Tr}\Big(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T\big(\tilde{\boldsymbol{\Lambda}}_R\otimes\tilde{\boldsymbol{\Lambda}}_{M_d}\big)\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\Big)\Big\} \\
&= -\frac{R\sum_d M_d}{2}\ln(2\pi) + \frac{\sum_d M_d}{2}\sum_r(\psi(c_N^r) - \ln d_N^r) + \frac{R}{2}\sum_d\sum_{m_d}(\psi(g_N^{m_d}) - \ln h_N^{m_d}) \\
&- \frac{1}{2}\operatorname{Tr}\Big\{\sum_d\big(\tilde{\boldsymbol{\Lambda}}_R\otimes\tilde{\boldsymbol{\Lambda}}_{M_d}\big)\Big(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T + \mathbf{V}^{(d)}\Big)\Big\}.
\end{aligned}
\tag{59}
$$

$$\begin{aligned}\mathbb{E}_q\left[\ln\ p(\boldsymbol{\lambda}_R)\right] &= \sum_r \left\{\ln\ \Gamma(c_0^r) + c_0^r \ln\ d_o^r + (c_0^r - 1)\mathbb{E}_q\left[\ln\ \lambda_r\right] - d_0^r \mathbb{E}_q\left[\lambda_r\right]\right\} \\ &= \sum_r \left\{\ln\ \Gamma(c_0^r) + c_0^r \ln\ d_o^r + (c_0^r - 1)(\psi(c_N^r) - \ln\ d_N^r) - d_0^r \frac{c_N^r}{d_N^r}\right\}\end{aligned} \tag{60}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln\ p(\boldsymbol{\lambda}_{M_d})\right] &= \sum_d \sum_{m_d} \left\{\ln\ \Gamma(g_0^{m_d}) + g_0^{m_d} \ln\ h_o^{m_d} + (g_0^{m_d} - 1)\mathbb{E}_q\left[\ln\ \lambda_{m_d}\right] - h_0^{m_d} \mathbb{E}_q\left[\lambda_{m_d}\right]\right\} \\ &= \sum_d \sum_{m_d} \left\{\ln\ \Gamma(g_0^{m_d}) + g_0^{m_d} \ln\ h_o^{m_d} + (g_0^{m_d} - 1)(\psi(g_N^{m_d}) - \ln h_N^{m_d}) - h_0^{m_d} \frac{g_N^{m_d}}{h_N^{m_d}}\right\}\end{aligned} \tag{61}$$

$$\begin{aligned}\mathbb{E}_q\left[\ln\ p(\tau)\right] &= -\ln\ \Gamma(a_0) + a_0 \ln\ b_0 + (a_0 - 1)\mathbb{E}_q\left[\ln\ \tau\right] - b_0 \mathbb{E}_q\left[\tau\right] \\ &= -\ln\ \Gamma(a_0) + a_0 \ln\ b_0 + (a_0 - 1)(\psi(a_N) - \ln\ b_N) - b_0 \frac{a_N}{b_N}\end{aligned} \tag{62}$$

$$\begin{aligned}-\mathbb{E}_q\left[\sum_{d=1}^{D} \ln\ q(\mathbf{W}^{(d)})\right] &= \sum_d \sum_r \sum_{m_d} \mathbb{E}_q\left[\ln\ q(w_{m_d}^{(d)})\right] \\ &= \sum_d \left\{\frac{1}{2} \ln\ |\ \mathbf{V}_{m_d}^{(d)}\ |\right\} + \frac{R \sum_d M_d}{2}[1 + \ln(2\pi)]\end{aligned} \tag{63}$$

$$-\mathbb{E}_q[\ln\ q(\boldsymbol{\lambda}_R)] = \sum_r \left\{\ln\ \Gamma(c_N^r) - (c_N^r - 1)\psi(c_N^r) - \ln d_N^r + c_N^r\right\} \tag{64}$$

$$-\mathbb{E}_q[\ln\ q(\boldsymbol{\lambda}_{M_d})] = \sum_d \sum_{m_d} \left\{\ln\ \Gamma(g_N^{m_d}) - (g_N^{m_d} - 1)\psi(c_N^{m_d}) - \ln h_N^{m_d} + g_N^{m_d}\right\} \tag{65}$$

$$-\mathbb{E}_q[\ln\ q(\tau)] = \ln\ \Gamma(a_N) - (a_N - 1)\psi(a_N) - \ln\ b_N + a_N \tag{66}$$

In these expressions, $\psi(.)$ denotes the digamma function and $\Gamma(.)$ denotes Gamma function.

Combining all these terms together and omitting the const term, we obtain the following lower bound in a compact form

$$
\begin{aligned}
L(q) &= \frac{N}{2}(\psi(a_N) - \ln\ b_N) - \frac{a_N}{2b_N}\mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right] + \frac{\sum_d M_d}{2}\sum_r(\psi(c_N^r) - \ln d_N^r) \\
&+ \frac{R}{2}\sum_d\sum_{m_d}(\psi(g_N^{m_d}) - \ln h_N^{m_d}) - \frac{1}{2}\operatorname{Tr}\left\{\sum_d\left(\tilde{\mathbf{\Lambda}}_R \otimes \tilde{\mathbf{\Lambda}}_{M_d}\right)\left(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T + \mathbf{V}^{(d)}\right)\right\} \\
&+ \sum_r\left\{(c_0^r - 1)(\psi(c_N^r) - \ln\ d_N^r) - d_0^r\frac{c_N^r}{d_N^r}\right\} + \sum_d\sum_{m_d}\left\{(g_0^{m_d} - 1)(\psi(g_N^{m_d}) - \ln h_N^{m_d}) - h_0^{m_d}\frac{g_N^{m_d}}{h_N^{m_d}}\right\} \\
&+ (a_0 - 1)(\psi(a_N) - \ln\ b_N) - b_0\frac{a_N}{b_N} + \sum_d\left\{\frac{1}{2}\ln\mid\mathbf{V}^{(d)}\mid\right\} + \sum_r\{\ln\ \Gamma(c_N^r) - (c_N^r - 1)\psi(c_N^r) - \ln d_N^r + c_N^r\} \\
&+ \sum_d\sum_{m_d}\{\ln\ \Gamma(g_N^{m_d}) - (g_N^{m_d} - 1)\psi(c_N^{m_d}) - \ln h_N^{m_d} + g_N^{m_d}\} + (a_0 - 1)(\psi(a_N) - \ln\ b_N) - b_0\frac{a_N}{b_N} + \text{const}
\end{aligned}
$$

$$
\begin{aligned}
&= \left(\frac{N}{2} + a_0 - a_N\right)\psi(a_N) - \left(\frac{N}{2} + a_0\right)\ln\ b_N - \frac{a_N}{2b_N}\mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right] \\
&- \frac{1}{2}\operatorname{Tr}\left\{\sum_d\left(\tilde{\mathbf{\Lambda}}_R \otimes \tilde{\mathbf{\Lambda}}_{M_d}\right)\left(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T + \mathbf{V}^{(d)}\right)\right\} + \sum_d\left\{\frac{1}{2}\ln\mid\mathbf{V}^{(d)}\mid\right\} \\
&+ \sum_r\left\{\left(\frac{\sum_d M_d}{2} + c_0^r - c_N^r\right)\psi(c_N^r) - \left(\frac{\sum_d M_d}{2} + c_0^r\right)\ln\ d_N^r\right\} \\
&+ \sum_d\sum_{m_d}\left\{\left(\frac{R}{2} + g_0^{m_d} - g_N^{m_d}\right)\psi(g_N^{m_d}) - \left(\frac{R}{2} + g_0^{m_d}\right)\ln h_N^{m_d}\right\} \\
&+ \sum_r\left\{\ln\ \Gamma(c_N^r) + c_N^r - d_0^r\frac{c_N^r}{d_N^r}\right\} + \sum_d\sum_{m_d}\left\{\ln\Gamma(g_N^{m_d}) + g_N^{m_d} - h_0^{m_d}\frac{g_N^{m_d}}{h_N^{m_d}}\right\} + \ln\ \Gamma(a_N) + a_N - b_0\frac{a_N}{b_N} + \text{const}
\end{aligned}
$$

$$
\begin{aligned}
&= -\frac{a_N}{2b_N}\mathbb{E}_q\left[\|\mathbf{y} - \mathbf{\Phi}^T\mathbf{w}\|_F^2\right] - \frac{1}{2}\operatorname{Tr}\left\{\sum_d\left(\tilde{\mathbf{\Lambda}}_R \otimes \tilde{\mathbf{\Lambda}}_{M_d}\right)\left(\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})\operatorname{vec}(\tilde{\mathbf{W}}^{(d)})^T + \mathbf{V}^{(d)}\right)\right\} \\
&+ \sum_r\left\{\ln\ \Gamma(c_N^r) + c_N^r\left(1 - \ln\ d_N^r - \frac{d_0^r}{d_N^r}\right)\right\} + \sum_d\sum_{m_d}\left\{\ln\Gamma(g_N^{m_d}) + g_N^{m_d}\left(1 - \ln h_N^{m_d} - \frac{h_0^{m_d}}{h_N^{m_d}}\right)\right\} \\
&+ \frac{1}{2}\sum_d\ln\ \mid\mathbf{V}^{(d)}\mid + \ln\ \Gamma(a_N) + a_N(1 - \ln\ b_N - \frac{b_0}{b_N}) + \text{const}
\end{aligned}
$$

Finally the lower bound can be computed by using the posterior parameters of all unknowns in Θ, where the posterior parameters are updated at each iteration.

11. Predictive Distribution

$$
\begin{aligned}
p(\tilde{y_i} \mid \mathbf{y}) &= \int p\left(y_i \mid \Theta\right) p\left(\Theta \mid \mathbf{y}\right) d\Theta \\
&\simeq \int\int p\left(\tilde{y_i} \mid \left\{\mathbf{W}^{(d)}\right\}, \tau^{-1}\right) q\left(\left\{\mathbf{W}^{(d)}\right\}\right) q(\tau) d\left\{\mathbf{W}^{(d)}\right\} d\tau \\
&= \int\int \mathcal{N}\left(\tilde{y_i} \mid \overset{D}{\underset{d=1}{\circledast}} \mathbf{W}^{(d)} \varphi_i^{(d)}, \tau^{-1}\right) \prod_d \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(d)}), \mathbf{\Sigma}^{(d)}\right) q(\tau) d\left\{\mathrm{vec}(\mathbf{W}^{(d)})\right\} d\tau \\
&= \int\int \mathcal{N}\left(\tilde{y_i} \mid \mathrm{vec}(\mathbf{W}^{(1)})^T \mathbf{g}^{(d)}(x_n), \tau^{-1}\right) \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(1)}) \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(1)}), \mathbf{V}^{(1)}\right) d\left\{\mathrm{vec}(\mathbf{W}^{(1)})\right\} ... \\
&\quad ... \prod_{d\neq 1} \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(d)}), \mathbf{\Sigma}^{(d)}\right) q(\tau) d\left\{\mathrm{vec}(\mathbf{W}^{(d)})\right\}_{d\neq 1} d\tau \\
&= \int\int \mathcal{N}\left(\tilde{y_i} \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(1)})^T \mathbf{g}^{(d)}(x_n), \tau^{-1} + \mathbf{g}^{(d)}(x_n)^T \mathbf{V}^{(1)} \mathbf{g}^{(d)}(x_n)\right) ... \\
&\quad ... \prod_{d\neq 1} \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(d)}), \mathbf{\Sigma}^{(d)}\right) q(\tau) d\left\{\mathrm{vec}(\mathbf{W}^{(d)})\right\}_{d\neq 1} d\tau \\
&= \int\int \mathcal{N}\left(\tilde{y_i} \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(1)})^T \mathbf{g}^{(d)}(x_n), \tau^{-1} + \mathbf{g}^{(d)}(x_n)^T \mathbf{V}^{(1)} \mathbf{g}^{(d)}(x_n)\right) ... \\
&\quad ... \prod_{d\neq 1} \mathcal{N}\left(\mathrm{vec}(\mathbf{W}^{(d)}) \mid \mathrm{vec}(\tilde{\mathbf{W}}^{(d)}), \mathbf{\Sigma}^{(d)}\right) q(\tau) d\left\{\mathrm{vec}(\mathbf{W}^{(d)})\right\}_{d\neq 1} d\tau \\
&\quad \vdots \\
&= \int \mathcal{N}\left(\tilde{y_i} \mid \overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)} \varphi_i^{(d)}, \tau^{-1} + \sum_d \mathbf{g}^{(d)}(x_n)^T \mathbf{\Sigma}^{(d)} \mathbf{g}^{(d)}(x_n)\right) q(\tau) d\tau \\
&= \int \mathcal{N}\left(\tilde{y_i} \mid \overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)} \varphi_i^{(d)}, \tau^{-1} + \sum_d \mathbf{g}^{(d)}(x_n)^T \mathbf{\Sigma}^{(d)} \mathbf{g}^{(d)}(x_n)\right) \mathrm{Ga}(\tau \mid a_N, b_N) d\tau \\
&\simeq \mathcal{T}\left(\tilde{y_i} \mid \overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)} \varphi_i^{(d)}, \left\{\frac{b_N}{a_N} \sum_d \mathbf{g}^{(d)}(x_n)^T \mathbf{\Sigma}^{(d)} \mathbf{g}^{(d)}(x_n)\right\}^{-1}, 2a_N\right)
\end{aligned}
$$

where $\mathcal{T}$ is a Student's t-distribution $\tilde{y_i} \mid \mathbf{y} \sim \mathcal{T}(\tilde{y_i}, \mathcal{S}_i, \nu_y)$ with its parameters given by

$$
\begin{aligned}
\tilde{y_i} &= \overset{D}{\underset{d=1}{\circledast}} \tilde{\mathbf{W}}^{(d)} \varphi_i^{(d)}, \qquad \nu_y = 2a_N \\
\mathcal{S}_i &= \left\{\frac{b_N}{a_N} \sum_d \mathbf{g}^{(d)}(x_n)^T \mathbf{\Sigma}^{(d)} \mathbf{g}^{(d)}(x_n)\right\}^{-1}.
\end{aligned}
$$

Thus, the predictive variance can be obtained by $\mathrm{Var}(y_i) = \frac{\nu_y}{\nu_y - 2} \mathcal{S}_i^{-1}$.

References

K. Batselier, Z. Chen, and N. Wong. Tensor Network alternating linear scheme for MIMO Volterra system identification. *Automatica*, 84:26–35, 2017.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. ISBN 978-0-387-31073-2.

J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

L. Cheng, Y. C. Wu, and H. V. Poor. Probabilistic Tensor Canonical Polyadic Decomposition with Orthogonal Factors. *IEEE Transactions on Signal Processing*, 65(3):663–676, 2017.

W. Chu and Z. Ghahramani. Probabilistic Models for Incomplete Multi-dimensional Arrays. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 89–96. PMLR, 2009.

T. Dao, C. D. Sa, and C. Ré. Gaussian quadrature for kernel features. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 6109–6119, Red Hook, NY, USA, 2017. Curran Associates Inc.

P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(72):2153–2175, 2005.

A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. Deep convolutional networks as shallow gaussian processes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

R. Guhaniyogi, S. Qamar, and D. B. Dunson. Bayesian Tensor Regression. *Journal of Machine Learning Research*, 18(79):1–31, 2017.

B. Hammer and K. Gersmann. A note on the universal approximation capability of support vector machines. *Neural Processing Letters*, 17(1):43–53, 2003.

R. A. Harshman. Foundations of the parafac procedure: Model and conditions for an 'explanatory' multi-mode factor analysis. Technical Report 16, UCLA Working Papers in Phonetics, 1970.

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

J. L. Hinrich and M. Morup. Probabilistic Tensor Train Decomposition. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, A Coruna, Spain, 2019. IEEE.

J. L. Hinrich, K. H. Madsen, and M. Mørup. The probabilistic tensor decomposition toolbox. *Machine Learning: Science and Technology*, 1(2), 2020.

P. D. Hoff. Multilinear tensor regression for longitudinal relational data. *The Annals of Applied Statistics*, 9(3), 2015.

P. D. Hoff. Equivariant and Scale-Free Tucker Decomposition Models. *Bayesian Analysis*, 11(3): 627–648, 2016.

P. Izmailov, A. Novikov, and D. Kropotov. Scalable gaussian processes with billions of inducing inputs via tensor train decomposition. In A. Storkey and F. Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 726–735, 2018.

P. J. H. Jørgensen, S. F. V. Nielsen, J. L. Hinrich, M. N. Schmidt, K. H. Madsen, and M. Mørup. Probabilistic PARAFAC2, 2018.

P. J. H. Jørgensen, S. F. V. Nielsen, J. L. Hinrich, M. N. Schmidt, K. H. Madsen, and M. Mørup. Analysis of Chromatographic Data using the Probabilistic PARAFAC2. *Proceedings of Second Workshop on Machine Learning and the Physical Sciences*, 2019.

S. A. Khan and S. Kaski. Bayesian multi-view tensor factorization. In T. Calders, editor, *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 8724 of *Lecture Notes in Computer Science*, pages 656–671. Springer, 2014.

S. A. Khan, E. Leppäaho, and S. Kaski. Bayesian multi-tensor factorization. *Machine Learning*, 105 (2):233–253, 2016.

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

K. Konstantinidis, Y. Xu, Q. Zhao, and D. Mandic. Variational bayesian tensor networks with structured posteriors. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3638–3642, 2022.

J. B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and Its Applications*, 18(2):95–138, 1977.

J. W. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2018. cs, stat.

D. J. C. MacKay. Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer, 1994.

M. Mutný and A. Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 9019–9030. Curran Associates Inc., 2018.

R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996.

R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

A. Novikov, M. Trofimov, and I. Oseledets. Exponential machines. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 66(No 6 (Special Section on Deep Learning: Theory and Practice)): 789–797, 2018.

I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS)*, pages 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc.

P. Rai, Y. Wang, S. Guo, G. Chen, D. Dunson, and L. Carin. Scalable Bayesian Low-Rank Decomposition of Incomplete Multiway Tensors. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1800–1808. PMLR, 2014.

P. Rai, Y. Wang, and L. Carin. Leveraging Features and Networks for Probabilistic Tensor Decomposition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), 2015.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.

A. Solin and S. Särkkä. Hilbert space methods for reduced-rank gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

F. Wesel and K. Batselier. Large-scale learning with fourier features and tensor decompositions. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 17543–17554, 2021.

F. Wesel and K. Batselier. Quantized fourier and polynomial features for more expressive tensor network models, 2024.

C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*, pages 682–688. MIT Press, 2001.

J. M. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 211–222. Society for Industrial and Applied Mathematics, 2010.

Z. Xu, F. Yan, and Y. Qi. Bayesian Nonparametric Models for Multiway Data Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2):475–487, 2015.

Y. Yang and D. B. Dunson. Bayesian Conditional Tensor Factorizations for High-Dimensional Classification. *Journal of the American Statistical Association*, 111(514):656–669, 2016.

R. Yu, G. Li, and Y. Liu. Tensor Regression Meets Gaussian Processes. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 482–490. PMLR, 2018.

Q. Zhao, L. Zhang, and A. Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9): 1751–1763, 2015a.

Q. Zhao, L. Zhang, and A. Cichocki. Bayesian Sparse Tucker Models for Dimension Reduction and Tensor Completion, 2015b.

Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S. ichi Amari. Bayesian Robust Tensor Factorization for Incomplete Multiway Data. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):736–748, 2016.