Journal of Machine Learning Research 22 (2021) 1-48 Submitted 7/18; Revised 8/21; Published 9/21

Matrix Product States for Inference in Discrete Probabilistic Models

Rasmus Bonnevie RABO@DTU.DK
Cognitive Systems, Department of Applied Mathematics and Computer Science
Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

Mikkel N. Schmidt MNSC@DTU.DK
Cognitive Systems, Department of Applied Mathematics and Computer Science
Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

Editor: Ryota Tomioka

Abstract

When faced with problems involving inference in discrete domains, solutions often involve appeals to conditional independence structure or mean-field approximations. We argue that this is insufficient for a number of interesting Bayesian problems, including mixture assignment posteriors and probabilistic relational models (e.g. the stochastic block model). These posteriors exhibit no conditional independence structure, precluding the use of graphical model methods, yet exhibit dependency between every single element of the posterior, making mean-field methods a poor fit. We propose using an expressive yet tractable approximation inspired by tensor factorization methods, alternately known as the tensor train or the matrix product state, and which can be construed of as a direct extension of the mean-field approximation to higher-order dependencies. We give a comprehensive introduction to the application of matrix product state in probabilistic inference, and illustrate how to efficiently perform marginalization, conditioning, sampling, normalization, some expectations, and approximate variational inference in our proposed model.

Keywords: variational inference, matrix product states, tensor trains, discrete models, symmetry

1. Introduction

Inference in discrete Bayesian probabilistic models has been studied intensely for decades. In terms of a graphical model, inference problems can be solved exactly based on dynamic programming such as the junction tree algorithm (Lauritzen and Spiegelhalter, 1988; Wainwright and Jordan, 2008); however, the computational cost scales exponentially in the so-called treewidth, intuitively a measure of graph connectedness, roughly implying that sparse graphs (few dependencies) are straightforward to perform inference on, while densely connected graphs can be computationally intractable. Unfortunately, for many hierarchical models such as mixture models and probabilistic relational models, marginalization of nuisance parameters leads to posteriors that are completely connected, meaning that algorithms which rely on local conditional independence fall short.

One solution has been to use Markov chain Monte Carlo (MCMC) sampling, and in many scenarios marginalization actually induces posteriors that are more easily traversed by MCMC algorithms as the nuisance parameters no longer have to be set appropriately for a particular configuration to be likely (Teh et al., 2007). But MCMC on discrete spaces also prohibits the use of efficient gradient-based samplers and Gibbs sampling and other MCMC methods often tend to get stuck in local modes of the posterior distribution.

Variational inference is a completely separate strategy which has been applied successfully to e.g. mixture inference (Bishop, 2006; Hughes and Sudderth, 2013). The idea is to form an analytic approximation of the posterior distribution by minimizing a statistical distance between some tractable family of distributions and the true posterior. One issue with variational inference we would like to highlight here is that the approximations are often quite limited in their expressiveness, and suffer more from mode collapse than even the Gibbs sampler, as we will illustrate.

We consider probabilistic joint models of the form $p(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{Y}$ is a set of observed random variables (discrete and/or continuous) and $\boldsymbol{X}$ is a set of latent discrete random variables. We assume that there is no exploitable conditional independence structure in the model. Let N denote the number of latent variables $\boldsymbol{X} = \{X_n\}_{n=1}^N$, each of which take values in a discrete space $\mathcal{X}_n$ of size K_n. We denote every set $\boldsymbol{x} = \{x_i\}_{n=1}^N$ where $x_n \in \mathcal{X}_n$ is a configuration of the random variables, and we note that there are $K^* = \prod_{n=1}^N K_n$ different discrete configurations.

Inference is the procedure of reasoning appropriately given observations, which in the Bayesian setting involves evaluating as well as computing marginals and expectations over the posterior $p(\boldsymbol{X}|\boldsymbol{Y})$. This is computationally intractable in most cases, since it involves evaluating the model evidence $p(\boldsymbol{Y})$: a sum with an exponential number of terms. Variational inference circumvents the problem by defining a variational approximation $q(\boldsymbol{x}; \boldsymbol{\theta})$ to the true posterior $p(\boldsymbol{x}|\boldsymbol{Y})$ and maximizing the so-called evidence lower bound objective (ELBO)

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L} = \mathbb{E}_q[\ln p(\boldsymbol{x}, \boldsymbol{Y})] - \mathbb{E}_q[\ln q(\boldsymbol{x}; \boldsymbol{\theta})] \,,$$

where the expectation is with respect to the variational distribution $q(\boldsymbol{x}; \boldsymbol{\theta})$. The optimal solution implies the approximation with the lowest KL divergence to the posterior (Wainwright and Jordan, 2008).

Recent innovations have extended the tractable classes of approximations for probabilistic models over continuous random variables to arbitrarily complex distributions (Kingma and Welling, 2014; Ranganath et al., 2014; Rezende and Mohamed, 2015). For discrete random variables approximations based on invertible discrete flows (Tran et al., 2019; Hoogeboom et al., 2019; Kuśmierczyk and Klami, 2020) is a promising avenue of research. However, for most discrete distributions it is uncommon to see approximations that are not mean-field, i.e., where the approximation factorizes completely as $q(\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{n=1}^N q(x_n; \boldsymbol{\theta}_n)$. In the discrete case, this leaves a rather constrained design space as each independent discrete factor can in all generality be modeled with a categorical distribution represented by a probability vector $\boldsymbol{\theta}_n$, where $q(x_n = k) = \theta_{n,k}$ and $\sum_{k=1}^{K_n} \theta_{n,k} = 1$, $\forall n$.

While this approximation has been used successfully to find clusterings (Hughes and Sudderth, 2013), topics (Teh et al., 2007), and communities (Xu et al., 2014), it does so in part by being exceedingly coarse. It is well-known that approximations found through variational inference tend to underestimate variance and are mode-seeking by nature (Minka, 2005), and in the discrete setting this often translates into a low-variance collapse unto a particular locally-optimal configuration $\boldsymbol{x}^*$.

2. Probability Tensor Decomposition

To get a sense for the low fidelity of the mean-field approximation, we will draw a connection between multivariate discrete distributions and tensors. We will follow Kolda and Bader (2009) in giving a brief outline of tensors and operations thereon. An N'th order (or N-way) tensor is a multidimensional array $\mathcal{T}$ of shape $K_1 \times \ldots \times K_N$ where the element indexed by $\mathcal{I} = (i_1, \ldots, i_N)$ is denoted by $\mathcal{T}_{\mathcal{I}}$ and takes values in $\mathbb{R}$ or $\mathbb{C}$. In this sense, vectors and matrices are also tensors of order 1 and 2, respectively. The rows and columns of matrices generalize to n-mode fibers for higher-order tensors, defined as the vector $\boldsymbol{v}_{i_1,\ldots,i_{n-1},i_{n+1},\ldots i_N}$ where the elements are taken from a slice of the tensor where every index but one is kept fixed, i.e. $[\boldsymbol{v}_{i_1,\ldots,i_{n-1},i_{n+1},\ldots i_N}]_{i_n} = \mathcal{T}_{i_1,\ldots,i_N}$. Matrix products are likewise subsumed by the n-mode product $\times_n$, which is an operation on a tensor $\mathcal{T}$ with a matrix $\boldsymbol{A}$ of shape $K_* \times K_n$ which applies to the n'th index alone,

$$[\mathcal{T} \times_n \boldsymbol{A}]_{i_1,\ldots,j_n,\ldots i_N} = \sum_{i_n=1}^{K_n} \mathcal{T}_{i_1,\ldots,i_n,\ldots,i_N} A_{j_n,i_n}. \tag{1}$$

This can be written in terms of standard linear algebra via matricizations of $\mathcal{T}$ on each mode n, denoted by $\boldsymbol{T}_{(n)}$: the matricization is the matrix found by stacking all of the tensor's n-mode fibers row-wise into a matrix of shape $K_n \times (\prod_{m\neq n} K_m)$, i.e. $[\boldsymbol{T}_{(n)}]_{i_n, f(i_1,\ldots,i_{n-1},i_{n+1},\ldots i_N)} = \mathcal{T}_{\mathcal{I}}$, with f being a bijective function mapping the fiber indices to a row index in some arbitrary order.[1] Then the above n-mode product can be expressed using matricizations and standard matrix products,

$$(\mathcal{T} \times_n \boldsymbol{A})_{(n)} = \boldsymbol{A}\boldsymbol{T}_{(n)}.$$

There are two notions of rank, each going hand-in-hand with a particular kind of tensor decomposition. The first is the tensor rank R, which is the minimal number of vector outer products (otherwise known as rank-one tensors) needed to sum to the tensor $\mathcal{T}$. It is related to the canonical polyadic (CP) decomposition of the form

$$\mathcal{T} = \sum_{r=1}^{R} \boldsymbol{v}_1^{(r)} \circ \ldots \circ \boldsymbol{v}_N^{(r)}, \tag{2}$$

where $\circ$ is the tensor outer product such that $[\mathcal{T} \circ \boldsymbol{v}]_{i_1,\ldots,i_{D+1}} = \mathcal{T}_{i_1,\ldots,i_D} v_{i_{D+1}}$. Second, there's the multilinear rank which is a vector $(r_1, \ldots, r_N)$, where r_n is equivalently the rank of $\boldsymbol{T}_{(n)}$ or the minimal number of rows in the matrix $\boldsymbol{U}_n$ of size $r_n \times K_n$, which features in the higher-order SVD (or Tucker) decomposition,

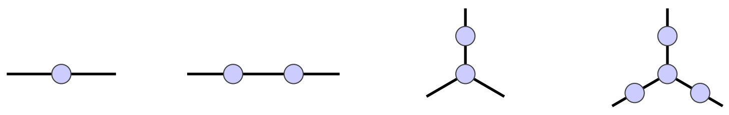
$$\mathcal{T} = \mathcal{C} \times_1 \boldsymbol{U}_1 \times_2 \ldots \times_N \boldsymbol{U}_N, \tag{3}$$

or, using index notation,

$$\mathcal{T}_{i_1,\ldots,i_N} = \sum_{j_1=1}^{r_1} \ldots \sum_{j_N=1}^{r_N} \mathcal{C}_{j_1,\ldots,j_N} [\boldsymbol{U}_1]_{i_1 j_1} \ldots [\boldsymbol{U}_N]_{i_N j_N}, \tag{4}$$

where $\mathcal{C}$ is the $r_1 \times \ldots \times r_N$ core tensor. By counting the number of rank-one terms in the Tucker decomposition, we get that $R \leq \prod_{n=1}^{N} r_n \leq \prod_{n=1}^{N} K_n$. It should be noted that the final upper bound is loose.

1. The stacking order is inconsequential, as long as it is used consistently.



(a) Matrix. (b) Matrix product. (c) N-mode product. (d) Tucker decomposition.

Figure 1: Examples of tensor network diagrams.

2.1 Tensor Network Diagrams

To get a better understanding of tensor operations, it helps to use tensor network diagrams to visualize the operations. The central idea of tensor network diagrams is to represent each tensor as a vertex with the number of outgoing edges reflecting the order of the tensor. Using this syntax, matrices have two outgoing edges (Fig. 1a) whereas vectors only have one, and higher-order tensors can have an arbitrary number of outgoing edges. Operations on the tensors are represented in the diagrams by connecting two edges to each other, which indicates that the edge is *contracted*: if two tensors $\mathcal{A}_{\mathcal{I},i_*}$ and $\mathcal{B}_{\mathcal{J},i_*}$ are contracted over the edge i_* (of dimension K_*), the resulting tensor takes the value,

$$\mathcal{C}_{\mathcal{I},\mathcal{J}} = \sum_{i_*=1}^{K_*} \mathcal{A}_{\mathcal{I},i_*} \mathcal{B}_{\mathcal{J},i_*}. \tag{5}$$

If we take two matrices $\boldsymbol{A}_{i,k}$ and $\boldsymbol{B}_{k,j}$ and contract them over the index k, the result is the matrix product $\boldsymbol{AB}$, as depicted graphically in Fig. 1b. Tensor contractions also include the n-mode product: A 2-mode product between a 3-tensor $\mathcal{A}_{ijk}$ and a matrix $\boldsymbol{B}_{j\ell}$ is identical to the contraction over index j, which can be graphically illustrated as in Fig. 1c.

In short, tensor network diagrams is a convenient graphical notation for expressing linear and multilinear algebra using the common language of tensors and tensor contractions, and can be used to depict any number of tensors and contractions. Tensor network diagrams can succinctly express many standard decompositions, e.g. the order-3 Tucker decomposition illustrated in Fig. 1d, where the three low-rank matrices $\boldsymbol{U}_n$ link to the shared order-3 core tensor $\boldsymbol{C}$.

2.2 Probability Tensors

To connect tensors to probabilistic models, consider that we can associate each configuration $\boldsymbol{x}$ with a posterior probability value $p(\boldsymbol{x}|\boldsymbol{Y})$, returning here to the notation used in the introduction. We can combine these values into a $K_1 \times \ldots \times K_N$ *probability tensor* $\mathcal{T}_{\boldsymbol{x}} = p(\boldsymbol{x}|\boldsymbol{Y})$, where the $\boldsymbol{x}$ subscript indicates that dimension n is indexed by the value of x_n, and the dependency on the constant observables $\boldsymbol{Y}$ is suppressed. This is a probability tensor in the sense that each element corresponds to a configuration, and it sums to 1, so if we vectorized it into a vector of length K^*, it could parameterize a categorical distribution over all possible configurations, similarly to the $\boldsymbol{\theta}_n$ parameters described previously.

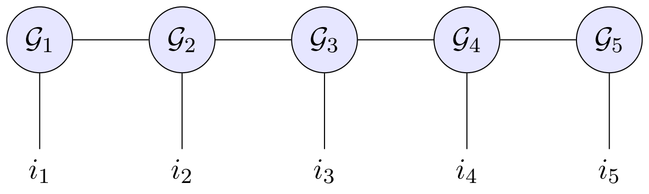


Figure 2: Tensor train as a diagram

Now, as a distribution, the mean-field approximation $q(\boldsymbol{x};\boldsymbol{\theta})$ also defines a probability tensor with elements

$$\hat{\mathcal{T}}_{\boldsymbol{x}} = \theta_{1,x_1} \ldots \theta_{N,x_N}. \tag{6}$$

In tensor parlance, this is a rank-one tensor of form $\hat{\mathcal{T}} = \boldsymbol{\theta}_1 \circ \ldots \circ \boldsymbol{\theta}_N$. Returning to the concept of tensor rank, we note that the approximation has the minimal rank possible, and is thus in some sense maximally simple. It seems optimistic to believe that the posterior probability tensors will have such simplistic structure, and Kolda and Bader (2009) go on to cite a result of a Monte Carlo experiment demonstrating that even for a small randomly-generated $2 \times 2 \times 2$ tensor, rank-one tensors occur with zero probability.[2]

2.3 Tensor Trains

So if we accept that approximation with a rank-one tensor is a flawed approach, what should we do instead? Recall that the sought tensor $\mathcal{T}$ has $K^* = \prod_{n=1}^{N} K_n$ elements, which grows exponentially with N, which rules out a naive representation. We already saw two structured representations, namely the CP representation of equation (2), and the Tucker representation in (3). Tucker unfortunately suffers from the same combinatorial explosion as $\mathcal{T}$ due to the existence of the tensor $\mathcal{C}$. CP on the other hand is convenient, but given that expressing $\mathcal{T}$ can require up towards K^* rank-one tensors, it might not be the most parsimonious representation.

Oseledets (2011) propose a different decomposition. Starting from the tensor $\mathcal{T}$, we can find a low-rank decomposition of the 1-mode unfolding as

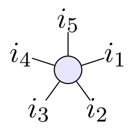
$$\boldsymbol{T}_{(1)} = \boldsymbol{U}_1 \boldsymbol{V}_1^\top, \tag{7}$$

where we choose $\boldsymbol{U}_1$ to be orthogonal (which is possible using SVD). Now, $\boldsymbol{V}_1$ will have shape $r_1 \times \prod_{n=2}^{N} K_n$ where r_1 is the rank of the low-rank decomposition used. To continue, we reshape $\boldsymbol{V}_1$ to have shape $r_1 K_2 \times \prod_{m=3}^{N} K_m$. We can then recursively define $\boldsymbol{U}_n$ as

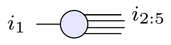
$$\boldsymbol{U}_n \boldsymbol{V}_n^\top = \text{Reshape}\left(\boldsymbol{V}_{n-1};\ \left(r_{n-1}K_n, \prod_{m=n+1}^{N} K_m\right)\right).$$

Written in terms of tensor reshapes and index-heavy notation like above, this decomposition can seem dense, but the algorithm is quite elegant when expressed using tensor network diagrams. Taking some higher-order tensor (Fig. 3a), we can represent matricizations and vectorizations as simple grouping of the indices as in Fig. 3b, conveniently abstracting away

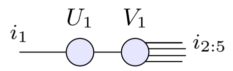
2. Note, though, that the posterior probability tensors are not random draws from the space of tensors.



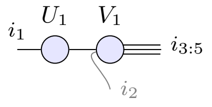
(a) Order 5 tensor



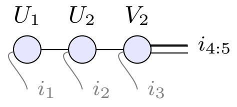
(b) Matricization of tensor



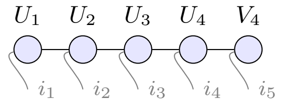
(c) Low-rank decomposition



(d) Matrix reshaped to have i_2 on the left side.



(e) SVD and reshape on V_1.



(f) Iterating yields TT

Figure 3: The tensor train decomposition algorithm illustrated using tensor network diagrams.

from the index ordering function we had to introduce previously. Using a singular value decomposition in practice, we can then low-rank decompose our matricized tensor like a regular matrix (Fig. 3c). The reshape step of the algorithm corresponds graphically to moving one edge from the right to the left in our diagram as done in Fig. 3d. In the example we move i_2, so that the left side of the right-most tensor in the diagram is indexed by i_2 and the edge introduced by the low-rank decomposition, while the right side has the remaining edges, except i_1 which is now only connected to the single orthogonal matrix U_1. Finally, we can repeat the SVD to introduce a corresponding low-rank factor for i_2 (Fig. 3e) and so on until the entire matrix has been decomposed into contractions of simple factors as in Fig. 3f.

Compared to CP and Tucker, it is less clear how to expand this into an algebraic expression. Looking at Fig. 3f again, we note that each U_n, while naturally a matrix-structured variable as it was derived using a low-rank decomposition like SVD, can quite naturally be thought of as having two legs on the left-hand side i.e. the leg that was bent over that we have labelled i_n and the edge connecting it to the previous matrix. If we reshape it to be an order-3 tensor with the two contracted edges and the edge i_n, we get the so-called core tensors $\mathcal{G}_n$ which in index notation are reshaped matrices

$$\text{Reshape}(\boldsymbol{U}_n;\ (r_{n-1}, K_n, r_n))$$

with $r_0 = r_N = 1$. Using the cores, we can elegantly express the individual indexed elements of the tensor as

$$\mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1]\mathcal{G}_2[i_2]\dots\mathcal{G}_N[i_N],$$

where we abuse notation slightly to define $[\mathcal{G}_n[k]]_{ij} = [\mathcal{G}_n]_{ikj}$ to be the 2nd mode matrix slices. This sequential construction of "carriages" linked together has led to naming the decomposition the tensor train (TT) decomposition.

Evaluation of the tensor train requires $N-1$ matrix products, which would normally cost $\mathcal{O}(r_n^3)$ a piece, but since the first and last core have vector-shaped slices, we can calculate the whole train using only matrix-vector products ($\mathcal{O}(r_n^2)$) by starting multiplication from the left or the right.

The central advantage of tensor trains is the number of operations that can be efficiently implemented directly on the representation (Oseledets, 2011):

Contraction If we want to sum out index i_n, we can write the tensor in terms of its indices and move the sum to the appropriate matrix core

$$\sum_{i_n=1}^{K_n} \mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1]\dots\left(\sum_{i_n=1}^{N}\mathcal{G}_n[i_n]\right)\dots\mathcal{G}_N[i_N].$$

Scaling Scaling every core $\mathcal{G}_n$ by α_n is equivalent to scaling the tensor train by $\prod_{n=1}^{N}\alpha_n$.

Addition As can be verified using standard linear algebra, the TT decomposition $\mathcal{C}_n$ of the addition of two TT-tensors with cores $\mathcal{A}_n$ and $\mathcal{B}_n$ can be found to be

$$\mathcal{C}_1[k] = \begin{pmatrix}\mathcal{A}_1[k] & \mathcal{B}_1[k]\end{pmatrix}, \quad \mathcal{C}_N[k] = \begin{pmatrix}\mathcal{A}_N[k] \\ \mathcal{B}_N[k]\end{pmatrix}, \tag{8}$$

$$\mathcal{C}_n[k] = \begin{pmatrix}\mathcal{A}_n[k] & \mathbf{0} \\ \mathbf{0} & \mathcal{B}_n[k]\end{pmatrix}, \quad n \neq 1 \wedge n \neq N. \tag{9}$$

Multiplication Finally, the TT decomposition $\mathcal{C}_n$ of the element-wise multiplication of two TT-tensors $\mathcal{A}$ and $\mathcal{B}$ with cores $\mathcal{A}_n$ and $\mathcal{B}_n$ is simply $\mathcal{C}_n[k] = \mathcal{A}_n[k]\otimes\mathcal{B}_n[k]$, where $\otimes$ is the Kronecker product, as follows from

$$\mathcal{A}_{\mathcal{I}}\mathcal{B}_{\mathcal{I}} = \operatorname{Tr}\mathcal{A}_{\mathcal{I}}\operatorname{Tr}\mathcal{B}_{\mathcal{I}} = \operatorname{Tr}[\mathcal{A}_{\mathcal{I}}\otimes\mathcal{B}_{\mathcal{I}}] = \operatorname{Tr}[(\mathcal{A}_1[i_1]\otimes\mathcal{B}_1[i_1])\dots(\mathcal{A}_N[i_N]\otimes\mathcal{B}_N[i_N])]. \tag{10}$$

These operations are sufficient for e.g. calculating the Frobenius norm of a tensor (Oseledets, 2011), but they also turn out to be extremely useful for probabilistic models as we will now show. In probability theory there are a number of operations that are essential, such as marginalization, normalization, and conditioning: These operations are fairly easy to carry out on a probability tensor in TT format:

Marginalization We can find any desired marginal distribution by applying contraction to the indices that we want to marginalize over. As a convenience, we define the marginal core matrices

$$\boldsymbol{M}_n = \sum_{i_n=1}^{K_n}\mathcal{G}_n[i_n],$$

so that marginalization can be written as

$$p(\boldsymbol{x}_{/n}) = \sum_{i_n=1}^{K_n}\mathcal{T}_{\mathcal{I}} = \mathcal{G}_1[i_1]\dots\boldsymbol{M}_n\dots\mathcal{G}_N[i_N],$$

where $\boldsymbol{x}_{/n}$ denotes the full set of random variables $\{x_n\}_{n=1}^{N}$, excluding the n'th element.

Normalization Given an unnormalized probability tensor (i.e. any non-negative tensor), we can easily calculate the normalizing constant of the tensor by using the contraction operation presented above on every index. As contraction is the same as marginalization, we can write the normalization constant using the marginal cores defined above

$$Z = \boldsymbol{M}_1 \dots \boldsymbol{M}_N.$$

Having computed the normalization constant, we can also normalize the tensor using the scaling operation.

Conditioning Conditional distributions are likewise easily computed by fixing the core slices corresponding to the observations, and marginalizing out all remaining indices to compute the normalization constant.

As a further note of interest, we can also compute a large class of expectations, namely all those where the quantity of interest can be written as another tensor train $\mathcal{Q}$. The expectation is then simply

$$\mathbb{E}[\mathcal{Q}_I] = \sum_{\mathcal{I}} \mathcal{T}_{\mathcal{I}} \mathcal{Q}_{\mathcal{I}}, \tag{11}$$

which is an element-wise product, followed by a complete contraction over all indices.

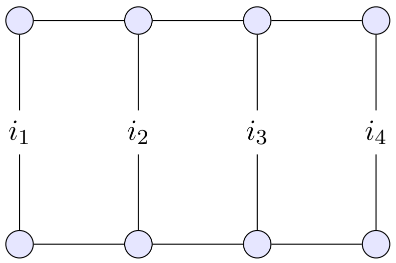
The main limitations of tensor trains are picking ranks, ordering the dimensions, and ensuring non-negativity. The rank of the tensor train is a fundamental hyperparameter which will effectively correspond to the expressiveness of the approximation. While most tensors seem to not be full rank, it makes sense to choose this parameter based predominantly on computational budget. While not pursued here, the ranks can also be compressed dynamically using the rounding procedure laid out in the original paper, which offers guarantees on approximation quality (Oseledets, 2011). There are no proper guidelines for ordering the dimensions, although the result saying that a perfect approximation exists holds no matter the ordering. It's also possible to use a tensor ring decomposition instead, which makes the tensor train invariant to cyclic permutations of the dimensions (Zhao et al., 2016). Non-negativity is the most crucial problem, as it is a necessary constraint which has to hold globally.

If all the cores are non-negative, the tensor will clearly be non-negative as well. From a tensor decomposition point of view it is however unlikely that the best representation of the tensor has only non-negative cores. Another construction that ensures non-negativity is a squaring, such that the tensor train $\hat{\mathcal{T}}$ itself models the square root of the probability tensor. By the algebra rules presented above, this implies a tensor train

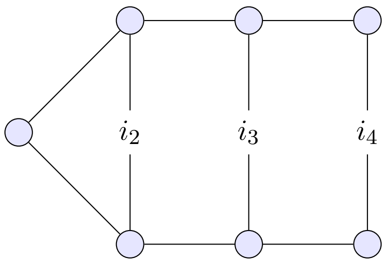
$$\mathcal{T}_{\mathcal{I}} = \hat{\mathcal{T}}_{\mathcal{I}}^2 = (\hat{\mathcal{G}}_1[i_1] \otimes \hat{\mathcal{G}}_1[i_1]) \dots (\hat{\mathcal{G}}_N[i_N] \otimes \hat{\mathcal{G}}_N[i_N]). \tag{12}$$

That is, the probability tensor can be guaranteed to be positive if each core matrix of the probability tensor can be represented by a Kronecker product of a matrix with itself,

$$\mathcal{G}_n[i_n] = \hat{\mathcal{G}}_n[i_n] \otimes \hat{\mathcal{G}}_n[i_n].$$



(a) Matrix Product State



(b) Contraction over i_1

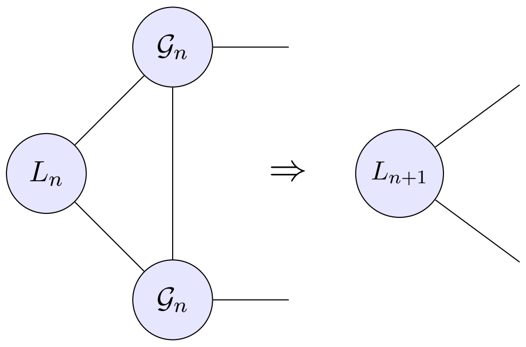


Figure 5: Iterative bubbling step

2.4 Matrix Product State

Interestingly, the tensor train decomposition has been developed independently within the quantum mechanics community under the name of matrix product states (see Schollwöck (2011) for a review of the history of its development). It falls within the larger umbrella of tensor networks, a general framework for constructing arbitrarily complex hierarchical tensor decompositions.

The matrix product state (MPS) literature contains a number of results not present in the tensor train literature, which motivates its introduction here. In quantum mechanics, the MPS $\mathcal{T}$ represents the quantum mechanical wave function of a combined quantum state of N particles where each particle x_n can be in one of K_n states. Following the probabilistic interpretation of quantum mechanics via Born's rule, the probability of the system being in a configuration $\boldsymbol{x}$ is exactly the square of the MPS, similarly to the squaring construction we employed above to ensure non-negativity. While an MPS is really synonymous with a tensor train, we will use the term to describe squared tensor trains as probabilistic models, in contrast with the more general tensor train. A problem with the squaring construction as presented, is that it incurs a bit of a performance hit since we have to operate with cores with squared ranks r_n^2 despite only being able to model the complexity of $\hat{\mathcal{T}}$, the square root of the tensor, with an effective rank of r_n. The literature on MPS proposes a solution for this, in the form of the *bubbling algorithm*, which can speed up the computation of contractions (and thus normalization, marginalization, and conditioning) significantly for particular cases (Bridgeman and Chubb, 2017; Robeva and Seigal, 2018).

Instead of following the derivation of multiplication in equation (10), consider the scenario where we want to contract a product of two tensor trains over i_1:

$$\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1]\dots\mathcal{A}_N[i_N]\mathcal{B}_1[i_1]\dots\mathcal{B}_N[i_N] = \mathcal{A}_N[i_N]^\top \dots \left(\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1]^\top \mathcal{B}_1[i_1]\right)\dots\mathcal{B}_N[i_N]. \quad (13)$$

Note that by transposition (possible since the quantity is scalar) we have gathered the factors depending on i_1 together. We can then analytically marginalize them out. Instead of being forced to instantiate Kronecker products of the cores, as would be necessary when calculating marginals of equation (12) so that a full contraction costs $\mathcal{O}(Nr^4)$, the bubbling algorithm only requires $\mathcal{O}(2Nr^3)$ operations. If we then want to also marginalize over i_2 we can write

$$\sum_{i_2=1}^{K_2} \mathcal{A}_2[i_2]^\top \left(\sum_{i_1=1}^{K_1} \mathcal{A}_1[i_1]^\top \mathcal{B}_1[i_1]\right) \mathcal{A}_2[i_2].$$

Returning to our squared tensor trains with cores $\mathcal{G}_n$, we introduce the left marginal matrices recursively as,

$$\boldsymbol{L}_1 = \sum_{i_1=1}^{K_1} \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1], \quad \boldsymbol{L}_n = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1} \mathcal{G}_n[i_n], \quad (14)$$

so the contraction over indices $i_1, \dots, i_m$ can be written as

$$\sum_{i_1,\dots,i_m} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathcal{G}_N[i_N]^\top \dots \mathcal{G}_{m+1}[i_{m+1}]^\top \boldsymbol{L}_m \mathcal{G}_{m+1}[i_{m+1}] \dots \mathcal{G}_N[i_N]. \quad (15)$$

This is helpful, but only for a rather limited set of contractions and marginals. Fortunately, we can define a similar series of *right* marginal matrices, by first noting that equation (15) can be written as

$$\mathrm{Tr}[\boldsymbol{L}_m \mathcal{G}_{m+1}[i_{m+1}] \dots \left(\mathcal{G}_N[i_N]\mathcal{G}_N[i_N]^\top\right) \dots \mathcal{G}_{m+1}[i_{m+1}]^\top].$$

If we then marginalize over first i_N, then i_{N-1} and so on, we can define the right marginal matrices as

$$\boldsymbol{R}_N = \sum_{i_N=1}^{K_N} \mathcal{G}_N[i_N]\mathcal{G}_N[i_N]^\top, \quad \boldsymbol{R}_n = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]\boldsymbol{R}_{n+1}\mathcal{G}_n[i_n]^\top,$$

and then finally, the contraction over $i_1, \dots, i_\ell$ and $i_u, \dots, i_N$ can be written as,

$$\sum_{i_1,\dots,i_\ell} \sum_{i_u,\dots,i_N} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathrm{Tr}[\boldsymbol{L}_\ell \mathcal{G}_{\ell+1}[i_{\ell+1}] \dots \mathcal{G}_{u-1}[i_{u-1}] \boldsymbol{R}_u \mathcal{G}_{u-1}[i_{u-1}]^\top \dots \mathcal{G}_{\ell+1}[i_{\ell+1}]^\top].$$

As a future convenience, let G_a^b denote the consecutive product of cores from $\mathcal{G}_a[i_a]$ to $\mathcal{G}_b[i_b]$, then we can also write the above statement as,

$$\sum_{i_1,\dots,i_\ell} \sum_{i_u,\dots,i_N} \hat{\mathcal{T}}_{\mathcal{I}}^2 = \mathrm{Tr}[\boldsymbol{L}_\ell G_{\ell+1}^{u-1} \boldsymbol{R}_u {G_{\ell+1}^{u-1}}^\top].$$

While this allows us to compute many marginals efficiently, a small caveat is that the efficiency drops once we start to consider non-consecutive marginals since we can only use the recursive formulas on the first and last indices. A final note is that we get a series of identities relating the left and right marginal matrices to the partition function,

$$Z = \mathrm{Tr}[\boldsymbol{L}_n \boldsymbol{R}_{n+1}] = \boldsymbol{L}_N = \boldsymbol{R}_1.$$

2.5 Canonical Representation

Another key feature employed in the MPS literature is the idea of a canonical set of cores. As presented, the tensor train is not a unique representation as applying a so-called gauge transform,

$$\tilde{\mathcal{G}}_n[i_n] = \boldsymbol{A}_n^{-1} \mathcal{G}_n[i_n] \boldsymbol{A}_{n+1}, \tag{gauge transform}$$

does not change the value of the corresponding tensor, i.e., the tensors represented by $\mathcal{G}_n$ and $\tilde{\mathcal{G}}_n$ are identical for all choices of $\boldsymbol{A}_n$. We can use this to strategically pick cores with desirable properties.

Left- and right-canonical matrix product states constitute two classical forms of canonicity (Schollwöck, 2011), defined as,

$$\sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \mathcal{G}_n[i_n] = \boldsymbol{I}, \quad \forall n, \tag{left-canonical}$$

$$\sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n] \mathcal{G}_n[i_n]^\top = \boldsymbol{I}, \quad \forall n. \tag{right-canonical}$$

Note that for left-canonical MPS's, this implies that $\boldsymbol{L}_1 = \boldsymbol{I}$, and by the definition in equation (14), all $\boldsymbol{L}_n = \boldsymbol{I}$. Similarly, for right-canonical MPS's it is true that all $\boldsymbol{R}_n = \boldsymbol{I}$. As an immediate consequence, $Z = \boldsymbol{L}_N = \boldsymbol{R}_1 = 1$ so that the tensor is automatically a probability tensor.

While the canonicity constraint might look complicated, it is actually easy to describe the set of cores for which it holds true. Defining stacked core matrices,

$$\boldsymbol{U}_n^{(L)} = \begin{pmatrix} \mathcal{G}_n[1] \\ \vdots \\ \mathcal{G}_n[K_n] \end{pmatrix}, \quad \boldsymbol{U}_n^{(R)} = \begin{pmatrix} \mathcal{G}_n[1]^\top \\ \vdots \\ \mathcal{G}_n[K_n]^\top \end{pmatrix}, \tag{16}$$

the constraints are simply ${\boldsymbol{U}_n^{(L)}}^\top \boldsymbol{U}_n^{(L)} = I$ and ${\boldsymbol{U}_n^{(R)}}^\top \boldsymbol{U}_n^{(R)} = I$ for left- and right-canonical MPS's, respectively. By inspection, the constraints are equivalent to the stacked matrices being column-orthogonal. Recalling the algorithm used to construct tensor trains using sequential SVD's from section 2.3, this condition is natural: There we defined the cores as reshapes of orthogonal matrices, and as it turns out, $\boldsymbol{U}_n = \boldsymbol{P}\boldsymbol{U}_n^{(L)}$ for a permutation $\boldsymbol{P}$, so that the cores returned by the algorithm define a left-canonical MPS. The final step of the algorithm returns a scalar coefficient, scaling the normalized tensor train appropriately.

Strict left/right-canonical form is only possible if we impose some constraints on the ranks r_n. For left-canonical matrices, $\boldsymbol{U}_n^{(L)}$ has shape $K_n r_{n-1} \times r_n$, so we need $r_n \leq K_n r_{n-1}$

to hold to ensure that the matrix can be column-orthogonal. If we assume $r_n = K_n r_{n-1}$ and $K = K_n$ for all n, this leads to the following progression of ranks

$$1, K, K^2 \dots K^2, K, 1.$$

Since the maximal rank grows exponentially, this is further evidence that we need to impose some additional low-rank assumption.

While canonicity is a computationally beneficial constraint to impose, it is not enough to ensure uniqueness. For a left/right-canonical MPS, we can still preserve both canonicity and the value of the MPS if we substitute $\mathcal{G}_n[i_n]$ for $\boldsymbol{Q}_n^\top \mathcal{G}_n[i_n]\boldsymbol{Q}_{n+1}$ for orthogonal matrices $\boldsymbol{Q}_n$. Schollwöck (2011) gives a good introduction to a unique representation alternately called the Vidal representation or the $\Gamma\Lambda$-representation as it defines (Vidal, 2003),

$$\mathcal{G}_n[i_n] = \boldsymbol{\Lambda}_{n-1}\boldsymbol{\Gamma}_n[i_n],$$

where $\boldsymbol{\Lambda}_n$ is a diagonal matrix and $\boldsymbol{\Gamma}_n$ is the same size as $\mathcal{G}_n[i_n]$. We additionally impose the constraint that $\mathcal{G}_n[i_n]$ is left-canonical,

$$\boldsymbol{I} = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]^\top \mathcal{G}_n[i_n] = \sum_{i_n=1}^{K_n} \boldsymbol{\Gamma}_n[i_n]^\top \boldsymbol{\Lambda}_{n-1}^2 \boldsymbol{\Gamma}_n[i_n].$$

There are of course many such possible decompositions. To make the representation unique, we note that,

$$\mathcal{T}_{\mathcal{I}} = \dots \boldsymbol{\Lambda}_{n-1} \underbrace{(\boldsymbol{\Gamma}_n[i_n]\boldsymbol{\Lambda}_n)}_{\tilde{\mathcal{G}}_n} \boldsymbol{\Gamma}_{n+1}[i_{n+1}] \dots,$$

where $\tilde{\mathcal{G}}_n$ describes an equivalent set of cores, defining the same tensor. It can then be shown that we can simultaneously require that this alternative representation corresponds to a right-canonical set of cores (Vidal, 2003; Schollwöck, 2011),

$$\boldsymbol{I} = \sum_{i_n=1}^{K_n} \mathcal{G}_n[i_n]\mathcal{G}_n[i_n]^\top = \sum_{i_n=1}^{K_n} \boldsymbol{\Gamma}_n[i_n]\boldsymbol{\Lambda}_n^2 \boldsymbol{\Gamma}_n[i_n]^\top.$$

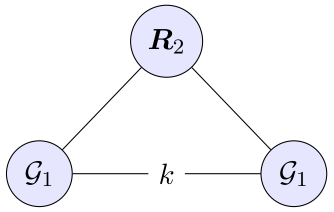
This is a particularly powerful representation as it allows us to shift effortlessly between left- and right-canonical core sets by simply collecting the $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ cores in different orders. In particular, we can employ mixed-canonical representations, where $\boldsymbol{L}_{n'} = \boldsymbol{I}$ for all $n' \leq n$ and $\boldsymbol{R}_{n'} = \boldsymbol{I}$ for all $n' > n$. A univariate marginal can then be computed in constant time as

$$p(x_n = i_n) = \mathrm{Tr}[\mathcal{G}_n[i_n]^\top \boldsymbol{L}_{n-1} \mathcal{G}_n[i_n] \boldsymbol{R}_{n+1}] = \mathrm{Tr}[\boldsymbol{\Gamma}_n[i_n]^\top \boldsymbol{\Lambda}_{n-1}^2 \boldsymbol{\Gamma}_n[i_n] \boldsymbol{\Lambda}_n^2].$$

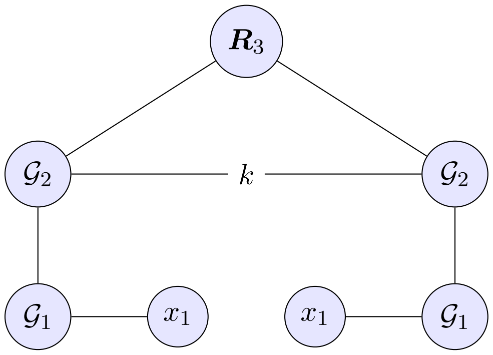
2.6 Sampling

An attractive feature in a probabilistic model is the ability to generate random samples. A direct approach to generate samples is to use an ancestral sampling routine based on the marginals and conditionals we have already discussed (Ferris and Vidal, 2012; Han et al., 2018). We will sample sequentially from the left, starting with x_1, then x_2, and so on. x_1 is simple to sample, as the marginal is readily available as

$$p(x_1 = k) = \mathrm{Tr}[\mathcal{G}_1[k]^\top \mathcal{G}_1[k] \boldsymbol{R}_2].$$



(a) Tensor network diagram for the marginal of $p(x_1 = k)$.



(b) Tensor network diagram for the conditional of $p(x_2 = k|x_1)$. The x_1 tensor is a one-hot vector of the sample.

Figure 6: Tensor networks of marginals and conditionals used in ancestral sampling.

Having sampled x_1, we can then go on to sample $x_2|x_1$ from the appropriate conditional distribution,

$$p(x_2 = k|x_1) \propto \mathrm{Tr}[\mathcal{G}_2[k]^\top \left(\mathcal{G}_1[x_1]^\top \mathcal{G}_1[x_1]\right) \mathcal{G}_2[k]\boldsymbol{R}_3],$$

and so on,

$$p(x_n = k|x_1, \dots x_{n-1}) \propto \mathrm{Tr}[\mathcal{G}_n[k]^\top \boldsymbol{C}_n \mathcal{G}_n[k]\boldsymbol{R}_{n+1}], \tag{17}$$

where the conditioning information of the past samples is summarized in a matrix

$$\boldsymbol{C}_n = \mathcal{G}_{n-1}[x_{n-1}]^\top \dots \mathcal{G}_1[x_1]^\top \mathcal{G}_1[x_1] \dots \mathcal{G}_{n-1}[x_{n-1}].$$

This sampling routine offers an excellent argument for why canonical forms can be useful: If we pick our MPS to be right-canonical, we can remove all of the right marginal matrices $\boldsymbol{R}_n = \boldsymbol{I}$ from the equations above, reducing the computational load.

3. Inference for the MPS

We consider a joint distribution $p(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{Y}$ is observed and $\boldsymbol{X} = \{X_n\}_{n=1}^N$ is a set of unobserved discrete variables, such that the joint distribution can be studied as a unnormalized probability tensor indexed by $\boldsymbol{X}$. Calculating the posterior distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ by Bayes' theorem,

$$p(\boldsymbol{X}|\boldsymbol{Y}) = \frac{p(\boldsymbol{X}, \boldsymbol{Y})}{p(\boldsymbol{Y})},$$

reduces to finding the evidence $p(\boldsymbol{Y})$ which is the normalization constant. If the domain of $\boldsymbol{X}$ is of moderate size, this is easily computed as the sum across all possible configurations, but this calculation suffers under a combinatorial explosion as N grows large.

We can instead consider approximations based on a finite number of observations of the probability tensor. As the tensor is unnormalized, the approximation problem is ill-posed. Consider, for instance, the scenario where all elements but one have been evaluated; if the final element is vanishingly small, the approximation is excellent, but if it has a value vastly larger than the observed elements, it can be arbitrarily poor.

Using an MPS as the approximate model alleviates this to a degree, as we limit our search to sufficiently regular probability tensors that can be written as a low-rank MPS. As such we hope to exploit that observing a minority of configurations will still inform us heuristically about the value of similar configurations.

While we could approximate the tensor directly, this approach is naive to the fact that the tensor represents a probability distribution, as witnessed by the fact that we would need to normalize the tensor post hoc. Instead, we will employ variational inference which minimizes a divergence on probability distributions. We use the Kullback-Leibler divergence (also known as the relative entropy), which is a information theoretic measure of the amount of information lost when using $q(\boldsymbol{X})$ to approximate $p(\boldsymbol{X}|\boldsymbol{Y})$,

$$\mathrm{KL}(q(\boldsymbol{X})\|p(\boldsymbol{X}|\boldsymbol{Y})) = \mathbb{E}_q\left[\ln \frac{q(\boldsymbol{X})}{p(\boldsymbol{X}|\boldsymbol{Y})}\right],$$

by solving the equivalent problem of maximizing the evidence lower bound (ELBO),

$$\ln p(\boldsymbol{Y}) \geq \mathcal{L} = \mathbb{E}_q\left[\ln \frac{p(\boldsymbol{X},\boldsymbol{Y})}{q(\boldsymbol{X})}\right].$$

This formulates the approximation as a maximization of an expectation over the MPS distribution, and is invariant with respect to the unknown normalization constant $p(\boldsymbol{Y})$. If q is parametric with parameters θ, which we denote q_θ for now, then the most common gradient estimators are (Ranganath et al., 2014; Kingma and Welling, 2014)

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\boldsymbol{X}\sim q}\left[\left(\ln \frac{p(\boldsymbol{X},\boldsymbol{Y})}{q(\boldsymbol{X})}\right)\nabla \ln q_\theta(\boldsymbol{X})\right] = \tag{18}$$
(score estimator)

$$\mathbb{E}_{\epsilon\sim q_0}\left[\nabla_\theta \ln \frac{p(h(\epsilon,\theta),\boldsymbol{Y})}{q(h(\epsilon,\theta))}\right]. \tag{19}$$
(reparametrization)

The score estimator is alternately known as the black-box gradient estimator or REINFORCE (Ranganath et al., 2014), while the reparametrization estimator depends on finding a function $h(\epsilon,\theta)$ and distribution q_0 such that $\hat{\boldsymbol{X}} = h(\epsilon,\theta)$ for $\epsilon \sim q_0$ is identically distributed to $\boldsymbol{X} \sim q_\theta(\boldsymbol{X})$.

The reparametrization estimator is often considered superior as it has empirically lower variance in many practical settings. Constructing a differentiable reparametrization for a discrete distribution is however impossible, but it has recently been shown that the gradient can be approximated with some fidelity using a relaxed differentiable version of the discrete distribution.

3.1 Differentiable MPS

As we saw in section 2.6, ancestral sampling from the MPS involves sampling from categorical distributions. To reparametrize a categorical random variable $k \sim \mathrm{Categorical}(\boldsymbol{p})$ with probability vector $\boldsymbol{p}$, we can employ the Gumbel-max trick which provides a non-differentiable reparametrization (Gumbel, 1954)

$$w_k = p_k + G_k, \quad k = \underset{k}{\mathrm{argmax}}(w_k), \tag{20}$$

where $G_k \sim \text{Gumbel}(0,1)$ are i.i.d. standard Gumbel random variables. Here, the argmax is the non-differentiable component, and the Gumbel variables take the role of the non-parametric random noise ϵ. To relax this non-differentiable component, we substitute it with

$$\text{onehot}(\operatorname*{argmax}_k w_k) \approx \text{softmax}(\boldsymbol{w}/T),$$

with the fidelity of the approximation increasing as the temperature $T \to 0$. This approximation is alternately known as Gumbel-softmax (Jang et al., 2017) or the concrete distribution (Maddison et al., 2017).

While this allows us to (approximately) reparametrize every step of the ancestral sampling process, it does not take into account that the categorical distribution of x_n,

$$p(x_n = k | x_1, \dots x_{n-1}) \propto \text{Tr}[\mathcal{G}_n[k]^\top \boldsymbol{C}_n \mathcal{G}_n[k] \boldsymbol{R}_{n+1}], \tag{21}$$

as seen in equation (17), depends conditionally on the exact samples $x_m, m < n$ from the past, via the recursively-defined conditioning tensor,

$$\boldsymbol{C}_n = \mathcal{G}_{n-1}[x_{n-1}]^\top \boldsymbol{C}_{n-1} \mathcal{G}_{n-1}[x_{n-1}].$$

To get around this, we can define a relaxed conditioning matrix,

$$\hat{\boldsymbol{C}}_n = \sum_{k=1}^{K} \hat{x}_{nk} \mathcal{G}_{n-1}[k]^\top \hat{\boldsymbol{C}}_{n-1} \mathcal{G}_{n-1}[k], \tag{22}$$

which is based on a separate sequence of $\hat{\boldsymbol{x}}_n = \text{onehot}(\hat{x}_n)$ continuous random variables, which we design to be Gumbel-softmax relaxations of the conditional sampling equation in equation (21),

$$z_{nk} = \log \text{Tr}[\mathcal{G}_n[k]^\top \hat{\boldsymbol{C}}_n \mathcal{G}_n[k] \boldsymbol{R}_{n+1}] \tag{23}$$

$$\hat{\boldsymbol{x}}_n = \text{softmax}\left(\frac{1}{T}(\boldsymbol{z}_n + \boldsymbol{G}_n)\right). \tag{24}$$

where we have swapped the proper conditioning matrix $\boldsymbol{C}$ for the relaxed version and where $\boldsymbol{G}_n$ is a vector of stacked i.i.d. Gumbel variables. As the conditioning matrix is a convex combination, we can also impose a mixture interpretation of the approximate conditional,

$$p(\hat{x}_n = k | \hat{x}_1, \dots \hat{x}_{n-1}) \propto \sum_{m=1}^{K} \hat{x}_{nk} \, \text{Tr}[\mathcal{G}_n[k]^\top \left(\mathcal{G}_{n-1}[m]^\top \hat{\boldsymbol{C}}_{n-1} \mathcal{G}_{n-1}[m]\right) \mathcal{G}_n[k] \boldsymbol{R}_{n+1}], \tag{25}$$

which is a mixture of the different conditionals that would arise based on the sampled value of x_n, weighted by the normalized Gumbel-softmax variables $\hat{x}_n$. By continuing to expand the conditioning matrices, we can get a mixture over all possible pasts $\hat{x}_1, \dots, \hat{x}_{n-1}$.

Note that in the low-temperature limit, this approaches the correct conditional sampling steps. We name the sequential procedure generating the $\hat{x}_n$ variables the differentiable MPS (dMPS), as every step of it is differentiable.

3.2 Unbiased Gradient Estimation

While using the dMPS as a direct substitute for the MPS during inference will often work fine, the consecutive approximations mean that we are no longer solving the original problem, but some facsimile thereof.

To improve this approximation, we can make use of recently proposed efficient unbiased gradient estimators for objectives involving discrete random variables, with the REBAR estimator and its generalization RELAX at the forefront (Tucker et al., 2017; Grathwohl et al., 2018). The RELAX estimator is unbiased and takes the form

$$\nabla_\theta \mathbb{E}_{q(x;\theta)}[f(x)] = \mathbb{E}\big[(f(x) - \mathbb{E}_{q(z|x;\theta)}[c_\phi(z)])\nabla_\theta \ln q(x;\theta)\big] + \tag{26}$$

$$\nabla_\theta \mathbb{E}_{q(z;\theta)}[c_\phi(z)] - \mathbb{E}_{q(z;\theta)}\big[\nabla_\theta \mathbb{E}_{q(z|b;\theta)}[c_\phi(z)]\big] \tag{27}$$

where f is a loss function or other scalar function of interest, c_ϕ is a parametric control variate, $x \sim q(x;\theta)$ is the discrete variable sampled from the parametric probabilistic model of interest q, and z is any random variable; ideally one strongly correlated with the original variable x. The estimator typically works best when $q(z;\theta)$ and $q(z|b;\theta)$ can be assumed to be reparametrizable.

In the original papers, z is picked to be a continuous reparametrization of x in the sense that $x = H(z)$ for some non-differentiable transfer function $H(\cdot)$. They then recommend setting $c_\phi(z) = f(\tilde{H}(z))$ where $\tilde{H}(z)$ is a differentiable approximation of H. RELAX further advocates augmenting c_ϕ as,

$$c_\phi(z) = f(\tilde{H}(z)) + c'_\phi(z),$$

where c' is a neural network or other high-capacity model (Grathwohl et al., 2018).

To design a RELAX estimator for the MPS, we must find a continuous variable $\boldsymbol{z}_n$ that correlates with each of the $\boldsymbol{x}_n = \text{onehot}(x_n)$ discrete variables in our model. It would be natural if we could use the differentiable relaxation $\hat{\boldsymbol{x}}_n$, but this is not immediately correlated with $\boldsymbol{x}_n$—in fact, the two define independent sequences of random variables. To correlate the two, we take $\boldsymbol{x}_n$ to be generated via a Gumbel-max reparametrization as in equation (20), and we then tie the Gumbel noise used in each true sampling step to that used for the matching relaxed sampling steps in equation (24). The corresponding coupled graphical model is depicted in Fig. 7. This approach is strongly inspired by one presented in the appendices of the original REBAR paper (Tucker et al., 2017).

Conditioning on $\boldsymbol{x}$, we can sample the Gumbel noise $\boldsymbol{G}_n$ conditioned on that information (see appendix A for details), and using that we can run the coupled dMPS forward. Given the continuous random variables $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_n\}_{n=1}^N$, we will assume our control variate is,

$$c_\phi(\hat{\boldsymbol{X}}) = \ln p(\hat{\boldsymbol{X}}, \boldsymbol{Y}) - \ln q(\hat{\boldsymbol{X}}) + c'_\phi(\hat{\boldsymbol{X}}),$$

This is not strictly correct as p and q are defined over discrete domains, but often there is a direct way to relax both. For an MPS q, we can relax it by doing a weighted marginalization,

$$\hat{\boldsymbol{L}}_1 = \sum_{i_1=1}^{K_1} \hat{x}_{1i_1} \mathcal{G}_1[i_1]^\top \mathcal{G}_1[i_1], \quad \boldsymbol{L}_n = \sum_{i_n=1}^{K_n} \hat{x}_{ni_n} \mathcal{G}_n[i_n]^\top \hat{\boldsymbol{L}}_{n-1} \mathcal{G}_n[i_n]. \tag{28}$$

This corresponds to standard indexing when the $\hat{\boldsymbol{x}}_n$ vectors are one-hot.

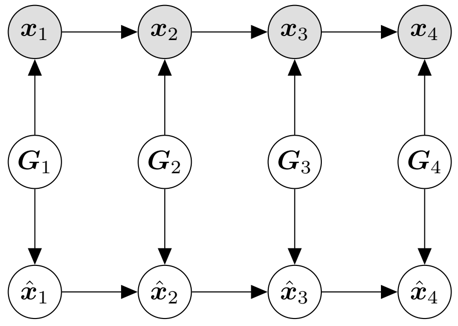


Figure 7: Coupled sampler for RELAX reparametrization. We can sample the continuous variables $\hat{\boldsymbol{x}}_i$ conditioned on the discrete $\boldsymbol{x}_i$.

3.3 Differentiable Normal Forms

Following the previous chapter, we can apply gradient-based optimization to the cores $\mathcal{G}_n[k]$, but we are left with some challenges, such as how to handle normalization. If we constrain the parameters to left/right-canonical cores, we not only get automatic normalization, but also reduce the issue of non-identifiability. We saw in section 2.5 that the defining constraint of left/right-canonical cores was equivalently stated in terms of an orthogonal matrix as in equation (16). By inverting this argument, we can take any column-orthogonal matrix and then define the cores (or their transposes, for right-canonicity) of an MPS to be equal to the blocks of the auxiliary orthogonal matrix. The respective canonical constraint will then be automatically maintained.

Given this, we need a differentiable parametrization of orthogonal matrices. There are in general two overall strategies for constructing orthogonal matrices: i) map a latent quantity through a map that has the orthogonal matrices as its image, or ii) construct a general orthogonal matrix by multiplying members of some set of elementary matrices (Shepard et al., 2015). The drawback of the former approach is that it tends to involve complex matrix operations like matrix inverses and matrix exponentials, while the latter results in long chains of repeated operations.

The Cayley transform (Shepard et al., 2015) says that

$$\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{A})(\boldsymbol{I} + \boldsymbol{A})^{-1}$$

is orthogonal, when $\boldsymbol{A}$ is skew-symmetric, i.e., $\boldsymbol{A}^\top = -\boldsymbol{A}$. $\boldsymbol{I} + \boldsymbol{A}$ is always invertible, and is often well-conditioned as it has eigenvalues $1 + i\lambda_k$, where $i\lambda_k$ are the eigenvalues of the skew-symmetric matrix.

Alternatively, to avoid the matrix inverse, we can express an arbitrary orthogonal matrix $\boldsymbol{Q} \in O(k)$ as the product of K Householder reflections (Sun and Bischof, 1995),

$$\boldsymbol{Q} = \prod_{k=1}^{K} H(\boldsymbol{v}_k), \quad H(\boldsymbol{v}_k) = \boldsymbol{I} - 2\frac{\boldsymbol{v}_k\boldsymbol{v}_k^\top}{\boldsymbol{v}_k^\top\boldsymbol{v}_k}, \quad \boldsymbol{v}_k \neq \boldsymbol{0}.$$

The complexity of the representation is $\mathcal{O}(K^3)$, so equal to the Cayley transform, but does not involve matrix inverses. On the other hand, the algebraic inverses can also lead to numerical issues, although we empirically observe that this is not common.

4. Symmetry

Clustering and mixture models are often described in terms of a random label x_n determining which cluster or mixture component each data point belongs to. As such, the individual label spaces of all the variables x_n can be identified with the same canonical label space of $K \equiv K_1 = \ldots = K_N$ discrete labels.

Typically, the labels are indistinguishable in the prior leading to both the prior and the posterior possessing a *relabeling symmetry*: given any permutation map $\sigma(\cdot)$ on the K labels, the configuration given by $\hat{x}_n = \sigma(x_n)$ is exactly as probable as the original configuration x_n (so $p(\boldsymbol{x}) = p(\hat{\boldsymbol{x}})$ for distribution p). In other words, for any particular partition of the data, the common label assigned to the elements in each set is inconsequential, as long as it encodes the same partition.

4.1 Marginals under Relabeling Symmetry

Mathematically, we can write distributions with relabeling symmetry as

$$p(\boldsymbol{x}) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \frac{\chi(\boldsymbol{x} \in \mathcal{L}_K(S))}{K!},$$

where $\sum_S \omega_S = 1$, χ is the indicator map which is 1 when its condition holds, and $\mathcal{S}_K(\mathcal{I})$ is the set of partitions of the set $\mathcal{I}$ into K non-empty parts, where each partition element S is a set of disjoint subsets of $\mathcal{I}$, with their union being $\mathcal{I}$. $\mathcal{L}_K(S)$ is the set of $K!$ labellings $\boldsymbol{x}$ of the partition S using K labels.

Relabeling symmetry has strong effects on the marginals of the distribution, as they by definition are uniform. Conditioning on a partition S we can factorize the distribution as

$$p(\boldsymbol{x}|S) = \sum_{k=1}^{K} \frac{\chi(x_j = k, j \in A)}{K} \frac{\chi(\boldsymbol{x}_{\mathcal{I}/A} \in \mathcal{L}_{K-1}(S/\{A\}))}{(K-1)!} \chi(x_j \neq k, j \notin A),$$

which holds for any choice of $A \in S$. If we choose $A = N(n, S)$ where $N(n, S)$ is the set of indices grouped together with index n under partition S, then we can compute

$$p(x_n = k) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \sum_{\boldsymbol{x}_{/n}} p(\boldsymbol{x}|S) = \sum_{S \in \mathcal{S}_K(\mathcal{I})} \omega_S \frac{\chi(x_j = k, j \in N(n, S))}{K} = \frac{1}{K},$$

where the second equality is true as we marginalize exactly over $p(\boldsymbol{x}_{\mathcal{I}/A}|S/\{A\})$ for $K-1$ labels, and the final equality is true by design.

Using the same type of arguments, we can find that the probability table of the bivariate marginal can be written as a constant plus diagonal matrix

$$p(x_n = k, x_m = \ell) = \begin{cases} \alpha_{nm}, & k = \ell, \\ \frac{1 - K\alpha_{nm}}{K^2 - K}, & k \neq \ell, \end{cases}$$

where $K\alpha_{nm}$ is the co-occurrence (co-clustering) probability that $x_n = x_m$. In other words, the distribution only distinguishes between whether the two points in question are clustered together, or apart.

4.2 Tensor Trains and Relabeling

In tensor language, we can say that a probability tensor has relabeling symmetry if its underlying distribution has it, and the relabeling symmetry means that the tensor is unchanged under a simultaneous and identical permutation along each mode, i.e.,

$$\mathcal{T} = \mathcal{T} \times_1 \boldsymbol{P}_\sigma \times_2 \ldots \times_N \boldsymbol{P}_\sigma$$

has to hold for each permutation σ, where $\boldsymbol{P}_\sigma$ is the matrix representation of σ, where $P_{ij} = 1$ if $\sigma(j) = i$, and zero otherwise. Adopting a general result from Kolda and Bader (2009), we can vectorize to get the equivalent matrix expression

$$\left(\boldsymbol{P}_\sigma^{\otimes N} - \boldsymbol{I}\right) \operatorname{vec}(\mathcal{T}) = 0,$$

where we use $\boldsymbol{P}_\sigma^{\otimes N} = \overbrace{\boldsymbol{P}_\sigma \otimes \ldots \otimes \boldsymbol{P}_\sigma}^{\text{N times}}$ to denote a Kronecker power. So the vectorized tensor has to live in the intersection of the null spaces of all of these massive $K^N \times K^N$ matrices. This statement can be made a bit more compact by noting that the set of all permutations can be constructed from a smaller subset of generating permutations. A classical choice is the set of all transpositions $i \leftrightarrow j$, swapping elements i and j, while the most compact consists of just two elements: a transposition of the two first elements $0 \leftrightarrow 1$, and the cycle permutation $\sigma(i) = i + 1 \pmod K$ that performs a circular shift of every index (Conrad, 2013; Miller, 1901).

Permutations on the coordinates of a tensor in a TT format are straightforward to apply, giving rise to a permuted tensor which can also be represented in TT format, with the cores being a reordering of the original cores: $\hat{\mathcal{G}}_n[k] = \mathcal{G}_n[\sigma(k)]$. It is of interest to consider whether we can encode knowledge about the posterior, such as the relabeling symmetry, directly into the MPS approximation as this might reduce both redundancy in representation and the risk of degenerate solutions.

In all generality, there is a trivial construction for making a tensor (or other function-like object) invariant to a finite closed group of transformations $\sigma \in \Pi$, which is to construct a new tensor by averaging over all the group elements

$$\hat{\mathcal{T}}_{\mathcal{I}} = \sum_{\sigma \in \Pi} \mathcal{T}_{\sigma(\mathcal{I})}.$$

This approach has been used previously in the kernel literature to make invariant kernels (Haasdonk and Burkhardt, 2007), and can be applied in the TT setting as well, as we have rules for adding tensor trains together. The problem with this approach arises when the cardinality $|\Pi|$ is large, as the rank grows linearly with the number of terms in the sum, and many groups of interest, including the set of permutations, possess a large number of elements. As such it is more attractive if we can find representations that directly encode the symmetries.

Limiting ourselves to $K = 2$, it is fairly simple to build tensor trains with symmetries using structured cores. This case, while minimally complex in some sense, is of interest due to its tight relationship to the concepts of bits and bit-strings. Huckle et al. (2013) consider a number of bit-string symmetries, including bit-shift, reverse, and relabeling, the latter of which they name the bit-flip symmetry. For $K = 2$, it is characterized by the single swap permutation $0 \leftrightarrow 1$. They further show that if

$$\mathcal{G}_n[1] = \boldsymbol{U}_n \mathcal{G}_n[0] \boldsymbol{U}_{n+1} \tag{29}$$

with $\boldsymbol{U}$ being a set of involutions where $\boldsymbol{U}_n^2 = \boldsymbol{I}$ and $\boldsymbol{U}_{N+1} = \boldsymbol{U}_0 = 1$, we can inspect the value of the tensor at index $\mathcal{I}$,

$$\mathcal{T}_{\mathcal{I}} = \ldots (U_n \mathcal{G}_n[i_n] U_{n+1})(U_{n+1} \mathcal{G}_{n+1}[i_{n+1}] U_{n+2}) \ldots = \tag{30}$$

$$\ldots \mathcal{G}_n[\bar{i}_n] \mathcal{G}_{n+1}[\bar{i}_{n+1}] \ldots = \mathcal{T}_{\bar{\mathcal{I}}}, \tag{31}$$

to see that the value at $\mathcal{I}$ is equal to that at index $\bar{\mathcal{I}}$ where the bar indicates the application of the bit-flip/swap permutation. The authors go on to show that for any tensor train with relabeling symmetry, there exists a representation where the cores follow the above relation (Huckle et al., 2013, Theorem 3.8). Note that the result is not air-tight: cores of the form above lead to the symmetry, and for every tensor train with the symmetry, there exists a set of cores with the properties above, but this does not directly exclude different sets of cores describing the same tensor, but without the property. Indeed, in the proof of theorem 3.8, they have to double the assumed rank to prove that the tensor train has a set of cores obeying equation (29). In the next section we will pursue a more general theory of symmetry. Our approach is inspired by the physics literature, where versions of MPS have been developed that are invariant to various symmetries, in particular of the continuous and spatial variety (Singh et al., 2010, 2011; Weichselbaum, 2012).

4.3 Representation Theory

In section 2.5, we noted that each tensor does not have a unique representation as a tensor train, and that a family of gauge transforms,

$$\mathcal{G}_n[k] \to \boldsymbol{A}_n^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1},$$

leave the implicit tensor invariant, as the $\boldsymbol{A}_n$ matrices cancel. If we conjecture that this is a necessary condition, such that the equivalence of two equal-sized tensor trains implies that they are gauge transforms of each other, we get the consequence that for tensor trains invariant to permutation, the permutation must be acting as a gauge transform (Bridgeman and Chubb, 2017). In other words,

$$\mathcal{G}_n[\sigma(k)] = \boldsymbol{A}_{n,\sigma}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma}. \tag{32}$$

We note that $\boldsymbol{A}_{n,\sigma}$ cannot depend on k, by the definition of the gauge transform. We can then also consider consecutive applications of multiple transforms, e.g.,

$$\mathcal{G}_n[\sigma_2(\sigma_1(k))] = \boldsymbol{A}_{n,\sigma_2}^{-1} \boldsymbol{A}_{n,\sigma_1}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma_1} \boldsymbol{A}_{n+1,\sigma_2} = \boldsymbol{A}_{n,\sigma_2 \circ \sigma_1}^{-1} \mathcal{G}_n[k] \boldsymbol{A}_{n+1,\sigma_2 \circ \sigma_1}.$$

The last equality states that the matrix representation for the composite map $\sigma_2 \circ \sigma_1$ should be equivalent to the product of the separate matrix representations for σ_1 and σ_2. If we assume that the set of transforms we are interested in form a finite group G under function composition, then the set of matrices obeying this relationship are exactly the matrix representations of said group. As permutations form groups, we can apply the above idea to our cases of interest. In the matrix product state setting, the squaring procedure means that the tensor train does not have to be strictly invariant; each matrix $\boldsymbol{A}_{n,\sigma}$ can be multiplied by any root of unity factor $(\alpha_{n,\sigma})^2 = 1$. We will assume that this scaling factor is 1 for simplicity.

Any finite group is isomorphic to a subgroup of the symmetric group $S(K)$ by Cayley's theorem. Every element $\sigma \in S(K)$ is a permutation on K elements, and we can define the so-called permutation representation $\rho: S(K) \to \mathbb{R}^{K\times K}$, consisting of the permutation matrices $\rho(\sigma) = \boldsymbol{P}_\sigma$. Specifically, this assumes a correspondence between element k and canonical basis vector $\boldsymbol{e}_k$, so that $\boldsymbol{P}_\sigma \boldsymbol{e}_k = \boldsymbol{e}_{\sigma(k)}$ (where we use $\sigma(k)$ to mean application of the permutation operation). Similarly, we can define another equivalent matrix representation by changing the basis as $\tilde{\boldsymbol{P}}_\sigma = \boldsymbol{Q}\boldsymbol{P}_\sigma\boldsymbol{Q}^{-1}$. We can furthermore construct matrix representations of size $nK \times nK$ by using the direct sum $\oplus$ as

$$(\boldsymbol{Q}_1\boldsymbol{P}_\sigma^{(1)}\boldsymbol{Q}_1^{-1}) \oplus (\boldsymbol{Q}_2\boldsymbol{P}_\sigma^{(2)}\boldsymbol{Q}_2^{-1}) = \begin{pmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{P}_\sigma^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_\sigma^{(2)} \end{pmatrix} \begin{pmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2 \end{pmatrix}^{-1},$$

where the resulting matrix is likewise a representation. While we have maintained an explicit basis above, note that this disappears under the gauge transform equivalence, so we will assume that $\boldsymbol{Q} = \boldsymbol{I}$ going forward.

If we assume that our maximum TT rank is a multiple of K, all of the intermediate ranks r_n will be multiples of K as well, and we can use direct sums of the standard permutations matrices to form representations $\boldsymbol{A}_{n,\sigma} = \boldsymbol{I} \otimes \boldsymbol{P}_\sigma$ at every site. Note that their inverse is given by their transpose.

To finalize the representation, we can select K elements $\sigma_k \in S(K)$ where $\sigma_k(1) = k$, to generate the cores

$$\mathcal{G}_n[k] = (\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k})\mathcal{G}_n[1](\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k})^\top,$$

implying that the representation is determined by the choice of matrix representation and a single "free" $\mathcal{G}_n[1]$. From equation (32), a constraint on $\mathcal{G}_n[1]$ is imposed, as it should be invariant to σ where $\sigma(1) = 1$. For the standard permutation representation presented above, the invariant subspace under this subgroup of permutations is the span of $\boldsymbol{e}_1$, and $\boldsymbol{1}$ which is invariant to all permutations. To ensure that the block-wise column- and row-space is the subspace spanned by these vectors, we can write

$$\mathcal{G}_n[1] = (\boldsymbol{I} \otimes \boldsymbol{V})\boldsymbol{B}_n(\boldsymbol{I} \otimes \boldsymbol{V})^\top, \quad \boldsymbol{V} = \begin{pmatrix} \boldsymbol{e}_1 & \frac{1}{\sqrt{K}}\boldsymbol{1} \end{pmatrix}, \quad \boldsymbol{B}_n \in \mathbb{R}^{\frac{2}{K}r_n \times \frac{2}{K}r_{n+1}}.$$

Alternatively, we can use a projection matrix to the same effect, but the above representation is less redundant.

4.4 Left-Canonical Form for the Relabeling Symmetric MPS

Due to the design, we can construe the parameter matrix as a block matrix of 2×2 blocks, $\boldsymbol{B}_n = \sum_{i,j} \boldsymbol{e}_i \boldsymbol{e}_j^\top \otimes \boldsymbol{B}_{nij}$, and use that to consider the inner product

$$\mathcal{G}_n[k]^\top \mathcal{G}_n[k] = (\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k} \boldsymbol{V}) \boldsymbol{B}_n^\top (\boldsymbol{I} \otimes \boldsymbol{\Omega}) \boldsymbol{B}_n (\boldsymbol{I} \otimes \boldsymbol{P}_{\sigma_k} \boldsymbol{V})^\top = \tag{33}$$

$$\sum_{ijrs} (\boldsymbol{e}_j \boldsymbol{e}_i^\top \boldsymbol{e}_r \boldsymbol{e}_s^\top \otimes \boldsymbol{V}_k \boldsymbol{B}_{nij}^\top \boldsymbol{\Omega} \boldsymbol{B}_{nrs} \boldsymbol{V}_k^\top) = \tag{34}$$

$$\sum_{js} \boldsymbol{e}_j \boldsymbol{e}_s^\top \otimes \boldsymbol{V}_k \left(\sum_{r=1}^{r_n/K} \boldsymbol{B}_{nrj}^\top \boldsymbol{\Omega} \boldsymbol{B}_{nrs} \right) \boldsymbol{V}_k^\top, \tag{35}$$

where $\boldsymbol{V}_k$ is the same as $\boldsymbol{V}$, with $\boldsymbol{e}_1$ swapped for $\boldsymbol{e}_k$ and $\boldsymbol{\Omega} = \boldsymbol{V}^\top \boldsymbol{V}$. If we standardize with respect to $\boldsymbol{\Omega}$ by setting $\tilde{\boldsymbol{B}}_n \leftarrow (\boldsymbol{I} \otimes \boldsymbol{\Omega}^{1/2}) \boldsymbol{B}_n$, we can write the left-canonicity condition from section 2.5 as

$$\boldsymbol{I} = \sum_{k=1}^{K} \mathcal{G}_n[k]^\top \mathcal{G}_n[k] = \sum_{js} \boldsymbol{e}_j \boldsymbol{e}_s^\top \otimes \sum_{k=1}^{K} \boldsymbol{V}_k \left(\sum_{r=1}^{r_n/K} \tilde{\boldsymbol{B}}_{nrj}^\top \tilde{\boldsymbol{B}}_{nrs} \right) \boldsymbol{V}_k^\top.$$

This condition translates into the block-wise statement that for $j = s$, the right Kronecker factor should be the identity matrix, and for $j \neq \ell$ it should be the zero matrix.

Let $\boldsymbol{M} = \sum_{r=1}^{r_n/K} \tilde{\boldsymbol{B}}_{nrj}^\top \tilde{\boldsymbol{B}}_{nrs}$. Splitting the products with $\boldsymbol{V}_k$ into rank-one elements yields

$$\sum_{k=1}^{K} \boldsymbol{V}_k \boldsymbol{M} \boldsymbol{V}_k^\top = \sum_{k=1}^{K} \left(M_{11} \boldsymbol{e}_k \boldsymbol{e}_k^\top + \frac{M_{22}}{K} \boldsymbol{1}\boldsymbol{1}^\top + \frac{1}{\sqrt{K}} \left(M_{12} \boldsymbol{e}_k \boldsymbol{1}^\top + M_{21} \boldsymbol{1} \boldsymbol{e}_k^\top \right) \right) = \tag{36}$$

$$M_{11} \boldsymbol{I} + \left(M_{22} + \frac{M_{12} + M_{21}}{\sqrt{K}} \right) \boldsymbol{1}\boldsymbol{1}^\top, \tag{37}$$

This immediately gives us the condition that $M_{11} = 1$ for diagonal blocks $j = s$ and $M_{22} = 0$ when $j \neq s$. Inspecting $\tilde{\boldsymbol{B}}_n$, this means that all columns with odd index should form an orthonormal basis $\boldsymbol{Q}_{\text{odd}}$. For all blocks it should furthermore hold that

$$M_{22} + \frac{1}{\sqrt{K}} (M_{12} + M_{21}) = 0,$$

which we can express in matrix form as

$$\boldsymbol{Q}_{\text{even}}^\top \boldsymbol{Q}_{\text{even}} + \frac{1}{\sqrt{K}} (\boldsymbol{Q}_{\text{even}}^\top \boldsymbol{Q}_{\text{odd}} + \boldsymbol{Q}_{\text{odd}}^\top \boldsymbol{Q}_{\text{even}}) = \boldsymbol{0} \Rightarrow \tag{38}$$

$$\boldsymbol{H}^\top \boldsymbol{H} + \frac{1}{\sqrt{K}} \boldsymbol{H}^\top + \frac{1}{\sqrt{K}} \boldsymbol{H} + \bar{\boldsymbol{H}}^\top \bar{\boldsymbol{H}} = \boldsymbol{0}, \tag{39}$$

where we get the last equation by defining $\boldsymbol{Q}_{\text{even}} = \boldsymbol{Q}_{\text{odd}} \boldsymbol{H} + \bar{\boldsymbol{Q}}_{\text{odd}} \bar{\boldsymbol{H}}$, where $\bar{\boldsymbol{Q}}_{\text{odd}}$ is the orthogonal subspace of $\boldsymbol{Q}_{\text{odd}}$, and $\boldsymbol{H}$ and $\bar{\boldsymbol{H}}$ are appropriately shaped coefficient matrices. By completing the square, we can transform the condition into

$$(\sqrt{K} \boldsymbol{H} + \boldsymbol{I})^\top (\sqrt{K} \boldsymbol{H} + \boldsymbol{I}) = \boldsymbol{I} - K \bar{\boldsymbol{H}}^\top \bar{\boldsymbol{H}}. \tag{40}$$

Taking the SVD of $\bar{\boldsymbol{H}} = \boldsymbol{LSW}^\top$ we can multiply by $\boldsymbol{W}^\top$ on both sides to get a diagonal matrix on the right-hand side,

$$\boldsymbol{W}^\top(\sqrt{K}\boldsymbol{H} + \boldsymbol{I})^\top(\sqrt{K}\boldsymbol{H} + \boldsymbol{I})\boldsymbol{W} = \boldsymbol{I} - K\boldsymbol{S}^2. \tag{41}$$

Equation (40) has an inner product on the left-hand side, so equation (40) is implicitly solved when

$$\boldsymbol{U} \equiv (\sqrt{K}\boldsymbol{H} + \boldsymbol{I})\boldsymbol{W}(\boldsymbol{I} - K\boldsymbol{S}^2)^{-\frac{1}{2}},$$

is an orthogonal matrix. Via algebra, a family of solutions can be found as

$$\boldsymbol{H} = \frac{1}{\sqrt{K}}\left(\boldsymbol{U}(\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}}\boldsymbol{W}^\top + \begin{pmatrix}\boldsymbol{U} & \bar{\boldsymbol{U}}\end{pmatrix}\begin{pmatrix}\boldsymbol{C}_1 \\ \boldsymbol{C}_2\end{pmatrix}\bar{\boldsymbol{W}}^\top - \boldsymbol{I}\right),$$

where for $\boldsymbol{U}$ and $\boldsymbol{W}$, the matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{W}}$ are orthogonal to their counterparts, and span the orthogonal complement of bases $\boldsymbol{U}$ and $\boldsymbol{W}$, respectively. If we use $\boldsymbol{U}_*$ and $\boldsymbol{W}_*$ to denote the concatenation of the two complementary bases into a full basis (square orthogonal matrix),

$$\boldsymbol{H} = \frac{1}{\sqrt{K}}\left(\boldsymbol{U}_*\boldsymbol{C}_*\boldsymbol{W}_*^\top - \boldsymbol{I}\right), \quad \bar{\boldsymbol{H}} = \boldsymbol{LSW}^T, \quad \boldsymbol{C}_* = \begin{pmatrix}(\boldsymbol{I} - K\boldsymbol{S})^{\frac{1}{2}} & \boldsymbol{C}_1 \\ \boldsymbol{0} & \boldsymbol{C}_2\end{pmatrix},$$

fulfills the projected condition from equation (41) for arbitrary orthogonal $\boldsymbol{U}$, $\boldsymbol{W}$, $\boldsymbol{L}$ and diagonal $\boldsymbol{S}$ with $|S_{ii}| < \frac{1}{K}$. Plugging the result back into the original constraint from equation (40), we get the final condition

$$\boldsymbol{I} = \boldsymbol{C}_*^\top\boldsymbol{C}_* + \begin{pmatrix}K\boldsymbol{S}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}\end{pmatrix} = \begin{pmatrix}\boldsymbol{I} & (\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}}\boldsymbol{C}_1 \\ \boldsymbol{C}_1^\top(\boldsymbol{I} - K\boldsymbol{S}^2)^{\frac{1}{2}} & \boldsymbol{C}_1^\top\boldsymbol{C}_1 + \boldsymbol{C}_2^\top\boldsymbol{C}_2\end{pmatrix},$$

which reduces to two simplifying conditions: $\boldsymbol{C}_1 = \boldsymbol{0}$ and $\boldsymbol{C}_2$ has to be orthogonal.

For $\boldsymbol{S} = \boldsymbol{0}$, there is no weight on the orthogonal subspace, and

$$\boldsymbol{H} = \frac{1}{\sqrt{K}}\left(\boldsymbol{UW}^\top - \boldsymbol{I}\right),$$

where $\boldsymbol{QW}^\top$ is an arbitrary orthogonal matrix.

Just like with the original left/right-canonical forms, we cannot expect the above to work with arbitrary choices of rank: the identity matrix has full-rank, and by adding K rank-r matrices we can get at best a rank Kr matrix. Problematically, the above design is intrinsically of lower rank than it should be: the coefficient matrix $\boldsymbol{B}_n$ is only rank $\min(\frac{2}{K}r_n, \frac{2}{K}r_{n+1})$. Multiplying by K, we see that for the initial regime where $r_n < r_{n+1}$, we have to ensure $2r_n \geq r_{n+1}$. Maximizing the possible rank by setting $2r_n = r_{n+1}$, we get the rank progression

$$1, K, 2K, 2^2K, \ldots, 2^nK, \ldots K, 1.$$

Since this maximal rank is smaller than the maximal rank of the unconstrained problem, it might be the case that this is not simply due to the relabeling symmetry constraints, but that there exists symmetric tensors that are not expressible using the above representation. Indeed, this appears to be the case empirically. A simple remedy is to augment the size of the core to the maximal rank achievable in the unconstrained setting by zero padding the relabeling symmetric representation, and then add the same constant matrix to all core slices. While this appears to remove the low-rank problem, we have been unable to find a parametric canonical form for this augmented representation, which is left as future work.

5. Relationship to Probabilistic and Graphical Models

Whereas we have transformed the probabilistic model into a probability tensor and then approximated it with the MPS in the sense of a tensor approximation, it is possible to do it the other way around. In fact, there is a duality between tensor networks (the generalization of MPS's to more complex hierarchies of tensor products) and probabilistic graphical models (Robeva and Seigal, 2018). This dual view lends additional insights: For example, an efficient ordering of tensor contractions when computing e.g. a marginal or the normalization constant can be found by using the classical junction-tree algorithm on the dual graphical model (Robeva and Seigal, 2018). Prior to the explicit description of the duality, it was also shown how graphical models (or the corresponding decomposed log-densities) could be mapped to tensor networks (Novikov et al., 2014).

These two results highlight that we can in some sense characterize the MPS as a graphical model for which inference is particularly efficient. We find that MPS's bear resemblance to observable operator models (Jaeger, 2000) as well as weighted finite automata and the class of rational stochastic languages (Balle et al., 2015) which are formally distributions over all sequences Σ^* using some alphabet of states Σ where the distribution of $\boldsymbol{x} = (i_1, \dots, i_N) \in \Sigma^*$ is computable as

$$p_{\mathrm{RSL}}(\boldsymbol{x}) = \boldsymbol{v}_0^\top \boldsymbol{A}_{i_1} \dots \boldsymbol{A}_{i_N} \boldsymbol{v}_\infty$$

for some set of matrices $\boldsymbol{A}_k$. The MPS is tremendously similar, except that it is a distribution over sequences of length N only and allows for the state matrices $\boldsymbol{A}$, including dimensionality, to vary with the index. If we truncate the model above to only be over length N sequences, we can identify it with an MPS with the translation invariance property, where $\mathcal{G}_m[k] = \mathcal{G}_n[k]$ for all m, n (Schollwöck, 2011). Just like with the tensor trains, it is difficult to determine if the model, for some choice of parameters, assigns positive probability to all sequences, which lead to squaring constructions similar to the MPS to be explored in the literature (Bailly, 2011).

A final interesting link comes from the most prominent member of the above model class being the Hidden Markov model (HMM), which can be converted into the above form by setting

$$\boldsymbol{A}_i = \mathrm{Diag}(\boldsymbol{O}_{s,:})\boldsymbol{T} \tag{42}$$

for transitions $T_{ij} = p(z_{n+1} = j | z_n = i)$, latent states $z_n \in \Sigma$, and emission matrix $O_{ij} = p(x_n = j | z_n = i)$ where $\boldsymbol{O}_{s,:}$ denotes the s'th row (Jaeger, 2000). We highlight this as the observable operator model is a generalization of the HMM, which is limited by equation (42) constraining the shape of each $\boldsymbol{A}_k$ matrix (Jaeger, 2000). For instance, an HMM can never have negative elements in its transition matrices.

6. Experiments

In this section we will demonstrate the functionality of the matrix product state model when used as a variational approximation for tensor-shaped distributions. Throughout, we will use a stochastic block model (Nowicki and Snijders, 2001) posterior as our approximation target. It is a model for community detection in networks, which is related to other clustering problems. In particular, we assume an observed binary adjacency matrix $\boldsymbol{Y} \in \{0,1\}^{N \times N}$ for an undirected graph over N vertices. The stochastic block model for K communities is then,

$$\begin{aligned}
\boldsymbol{w} &\sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K), && && \text{(cluster proportions)} \\
x_i &\sim \text{Categorical}(\boldsymbol{w}), && \forall i \in 1, \ldots, N, && \text{(cluster assignments)} \\
\eta_{k\ell} &\sim \text{Beta}(a_{k\ell}, b_{k\ell}), && \forall k \in 1, \ldots, K, \ell \in k, \ldots, K, && \text{(link probabilities)} \\
\boldsymbol{Y}_{ij} &\sim \text{Bernoulli}(\eta_{x_i, x_j}), && \forall i \in 1, \ldots, N, j \in i+1, \ldots, N. && \text{(observed links)}
\end{aligned}$$

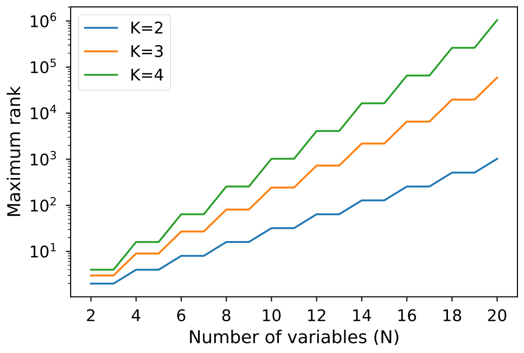
Here, $\eta_{k,\ell}$ describes the probability of a connection between community k and ℓ, with $\ell \geq k$, and $\{\alpha_k\}_{k=1}^K$, $\{a_{k\ell}\}_{k,\ell=1}^K$, and $\{b_{k\ell}\}_{k,\ell=1}^K$ are hyperparameters. We consider the label symmetric setting $\alpha_k = \alpha, a_{k\ell} = a, b_{k\ell} = b \forall k, \ell \in 1, \ldots, K$. This model cannot be written as a probability tensor in its current form, as it contains continuous variables $\boldsymbol{w}$ and $\eta_{k\ell}$. Fortunately, the model is in the conjugate exponential family, so we can integrate out $\boldsymbol{w}$ and $\eta_{k\ell}$, leaving a posterior density over the discrete x_i.

$$P(\boldsymbol{X}|\boldsymbol{Y}) \propto B(\alpha_1 + n_1, \ldots, \alpha_K + n_k) \prod_{k\ell} B(m_{k\ell} + a, \bar{m}_{k\ell} + b), \tag{43}$$

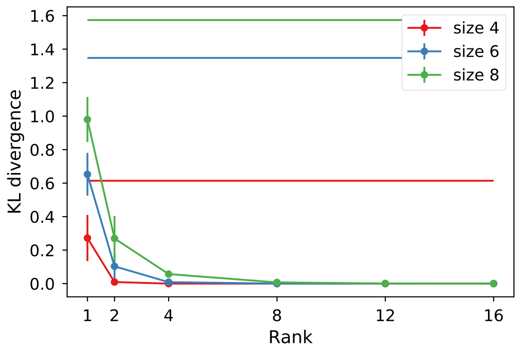
where B is the (multivariate) Beta function, n_k denotes the number of observations in cluster k (computed from $\boldsymbol{X}$), and $m_{k\ell}$, and $\bar{m}_{k\ell}$ denotes the number of edges and non-edges respectively between nodes in cluster k and ℓ (computed from $\boldsymbol{X}$ and $\boldsymbol{Y}$). If we let each x_i index a dimension of a tensor, we get a $K \times \ldots \times K$ (unnormalized) posterior probability tensor with N modes. This tensor will be our approximation target.

6.1 Influence of Rank

The rank of the matrix product state is the most significant tuning parameter, both in terms of modeling capacity and computational complexity. Using the incremental SVD algorithm defined in section 2.3, we are guaranteed to always be able to perfectly approximate any tensor, if only we use a sufficiently high rank. In the worst case, each iteration involves the SVD of a matrix with maximal rank, i.e. a matrix of size $K^i \times K^{N-i}$ in the i'th step, until we hit step $\lfloor N/2 \rfloor$, at which point the effective rank will start to decrease. The maximal rank is attained at this tipping point, so the maximal rank required to express a tensor with N modes of length K is $K^{\lfloor N/2 \rfloor}$. We plot this in Fig. 8a for different values of K. This



(a) Upper bound on rank



(b) KL divergence for varying rank

Figure 8: (a) The upper bound on the rank of the true TT decomposition of any tensor with N modes of dimensionality K. (b) The true KL divergence computed on three small tractable models (Erdős-Rényi random graphs with 4, 6 and 8 vertices) with approximations of varying rank. Errorbars denote standard deviation over 10 random graphs. Horizontal lines denote best mean-field solution.

is smaller than the number of elements in the tensor, but unfortunately scales in the same exponential manner. As such, low-rank assumptions are a necessity.

To give a brief demonstration of the model's potential efficacy, we generated small Erdős-Rényi random graphs with 4, 6 and 8 vertices. These are sufficiently small for us to compute the true gradient and true posterior, making it possible to calculate the actual KL divergence between the posterior and the approximation. We found the locally optimal approximation using an off-the-shelf BFGS optimization routine and 10 random restarts, and selected the solution with the smalles KL divergence. We did this for 10 random instances for each vertex count. Fig. 8b shows the average KL divergence at the best run, with the errorbars denoting the standard deviation across different random graphs. While the tensors in question here are very small with only 16, 64 and 256 elements respectively,

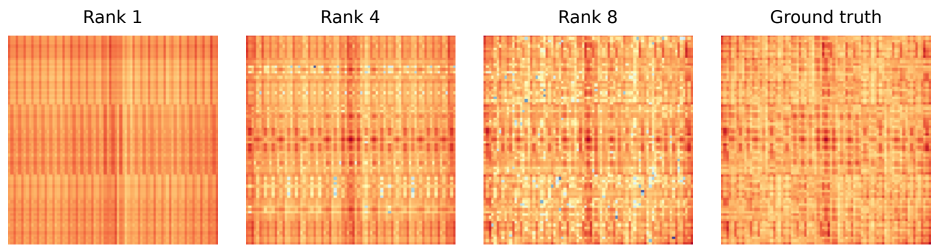


Figure 9: Mosaics (tiled heatmap visualization) of approximations of the posterior distribution when using varying rank. The underlying model is a stochastic block model with three communities on a small network with eight vertices. Structural complexity increases with rank, slowly approximating the complexity of the true tensor.

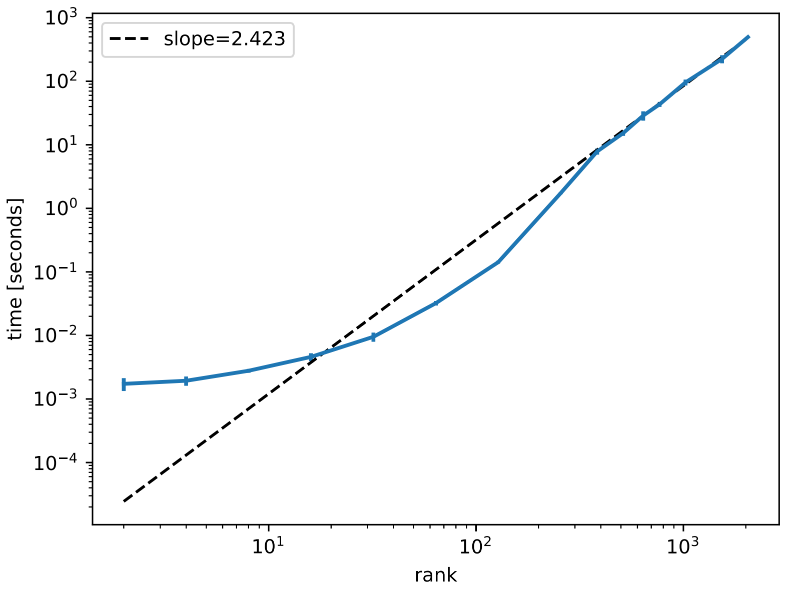
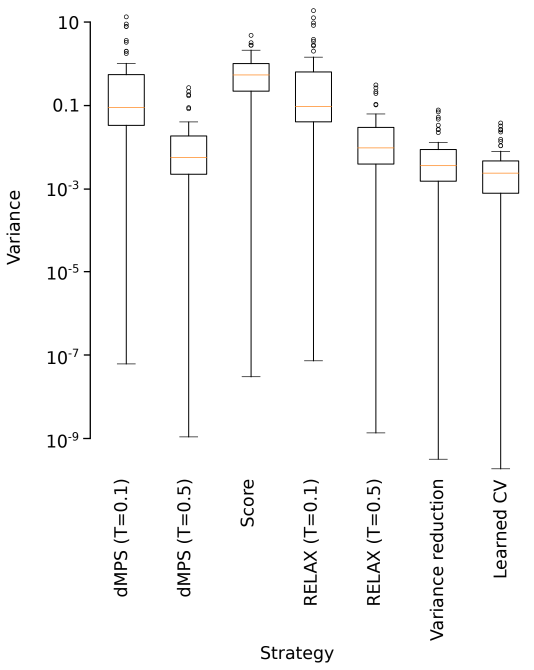


Figure 10: The time it takes to draw one sample (in seconds) as a function of the rank. Reference line is a regression on the last five points.

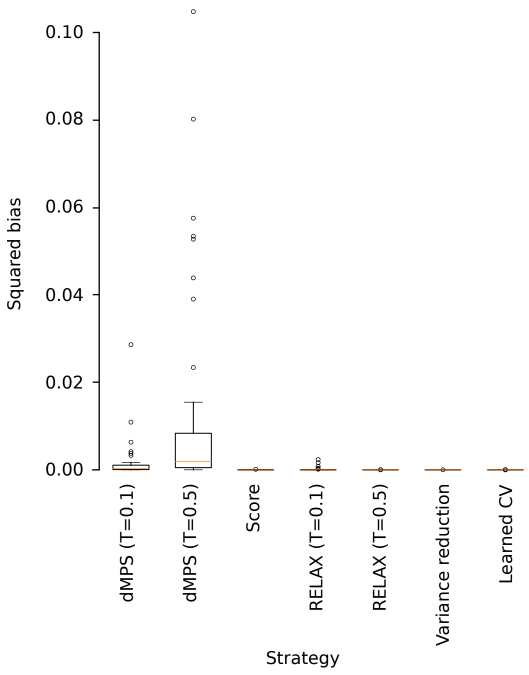
the maximal ranks are 4, 8, and 16, but in this experiment the approximation already appears quite trustworthy at around half that rank. Note that this result might depend a lot on how the random graph is generated, especially for larger graphs.

For further demonstration, we fitted models of varying rank using BFGS (as above) and unfolded the resulting tensors into a matrix. Naive unfoldings of a tensor have a tendency to hide its structural regularity, in much the same way as a random permutation on e.g. an adjacency graph can make even very regular graphs appear irregular. To form a structured unfolding, we can group the random variables (for graph clustering, one for each vertex) into two equally-sized ordered lists, which we then to use to index into the rows and columns of the matrix, respectively. We associate the i'th element x_i of the first list with the i'th digit of a base K number, i.e. $r = \sum_{i=0}^{N/2-1} x_i K^i$, which we can then use as a row index; we can do the same for the second list and the column index. In this way, we can map any configuration of the discrete random variables to an element in the matrix. We call the resulting matrix the mosaic of the tensor. We show mosaics of differently ranked tensor approximations in Fig. 9, compared to the true posterior tensor in the setting where we have $N = 8$ vertices and $K = 3$ communities.

The trade-off is of course that high rank impacts the performance of the model. We take an order-24 matrix product state representing a probability distribution over 24 binary random variables and calculate the time it takes to draw one 24-dimensional sample, and repeat the timing 10 times for different settings of the rank; the maximum rank for an order-24 tensor with binary states employing canonical cores is 2048, so we check various ranks between 2 and 2048. The timing results are illustrated in Fig. 10 which hints at a super-quadratic growth rate, although the exact growth rate is somewhat obscured by transient effects.



(a) Variance of the gradient estimators.



(b) Most of our methods are unbiased, except for inference using the dMPS directly.

Figure 11: Estimated squared bias and variance to true gradient with 20 000 gradient samples, each with 100 gradient draws per estimate. We register the estimated bias and variance for each parameter—the boxplots summarize the distribution of the estimates over the parameters. We use T to denote temperature used in Gumbel-softmax.

6.2 Variance Reduction

We have detailed a number of ways to estimate the gradients and control their variance. Keeping within the small N paradigm where the true gradient is tractable, we will demonstrate that the gradients exhibit different characteristics even at this scale. In particular, we take the graph to be an $N = 4$ cycle graph and look for $K = 2$ communities. We initialized randomly, and took 20 BFGS steps to get into the basin of attraction. Empirically, we observed that the gradients at random initializations were be extremely unstable, but we observed that the gradient algorithms tend to escape this regime quickly.

Fig. 11 shows the variance and squared bias of various estimators and control variates, including results for two temperatures of the Gumbel softmax. We took a 100-sample Monte Carlo gradient estimator as our gradient estimator, and resampled the estimator 20 000 times to produce the statistical summaries. We used an MPS with a canonical core using the Householder implementation which has 52 parameters. We calculated the summaries for each partial gradient individually, letting the box plots describe the distribution of the quantities across parameters. By using an unbiased estimate of the gradient variance along an unbiased estimate of the squared residual with respect to the true gradient, we can get an unbiased estimate of the squared bias by using the bias-variance trade-off identity.

The score estimator, our baseline, stands out as the highest variance estimator. The dMPS estimators employ a direct differentiation of a dMPS and are the simplest estimators in our arsenal. We note that for both temperature settings we achieve a reduction in variance, but the reduction is particularly significant for $T = 0.5$ which is in the same league as the best estimators. This is offset by its significant bias, and there we see the inverse relationship: the lower temperature estimator has significantly smaller bias. So the dMPS estimators seem to exhibit a very explicit bias-variance trade-off.

Comparing estimators of equal temperature to each other, the two static RELAX estimators (without optimization over the variance-reducing parameters) match the dMPS estimators in terms of variance, but as with all other RELAX-based estimators we see that they are unbiased. To be clear, these estimators use the dMPS as control-variate, without any added parametric component, which goes some way in explaining the correlation with that approach.

The models labeled *variance reduction* performs stochastic optimization on the variance by using the variance gradient estimator described in the REBAR and RELAX papers (Tucker et al., 2017; Grathwohl et al., 2018). We optimize for 40 000 steps using the AMSgrad algorithm which has improved convergence properties compared to Adam (Reddi et al., 2018).

The most advanced estimator we investigate is the *learned CV* model where we extend RELAX by using the control variate,

$$\nu(\ln p(\boldsymbol{x}, \boldsymbol{z}) - \ln q_{\mathrm{MPS}}(\boldsymbol{z}) + \alpha \hat{f}(\boldsymbol{z})),$$

where the two first terms make up the dMPS-based control variate, while $\hat{f}$ is a flexible function (a neural network) with everything scaled by a coefficient ν (Grathwohl et al., 2018). Due to the structure of our problem, we settle for using an MPS with canonical cores of maximum rank 2 instead, scaled by α. This works quite well, yielding the best unbiased estimator of the whole selection. It might be possible to improve upon this further by using a higher-rank MPS in the control variate, but there are certainly some complexity trade-offs.

A few empirical observations about training the variance reduced estimators are worth mentioning. That RELAX is better with high-temperature estimators goes against the intuition that we should strive for a control variate that is as close as possible to the true objective; apparently the increased variance we incur from a low temperature is too high a price. In fact, when we optimize for the temperature, we often see it increasing to values of 1 or higher, which is significantly above what we would otherwise expect based on model-fitting intuitions and the literature on Gumbel-softmax where temperatures around 0.5 are usually recommended (Maddison et al., 2017; Jang et al., 2017). The scalar weight ν multiplied onto the control variate is on the other hand quite stable, and almost always converges to 1. The few times ν diverged, it was usually preceded by the temperature dropping to a very low value, causing high variance gradients and unstable training. In general, we advocate keeping both the temperature and the scaling parameter from dropping too close to 0.

6.3 Biased vs Unbiased Gradients

The earliest of the recent bout of papers on gradient-based learning of discrete distributions hinged mostly on relaxations such as the Gumbel-softmax trick we employed to form the dMPS, which naturally introduced bias into the inference (Maddison et al., 2017; Jang et al., 2017). Although they had some success in their applications, we will make the case that the bias can be quite harmful.

In particular, we ran a dMPS versus an MPS trained with RELAX gradients and a learned control variate on an $N = 9$ graph (specifically the first 9 nodes of Zachary's karate network) taking $K = 2$. We used canonical cores, a gradient estimator based on 1 000 samples, and optimized using AMSgrad. We tracked the true ELBO, which is tractable due to the size of the graph, as well as the differentiable and stochastically estimated version of the ELBO targeted by the dMPS. Both of these learning curves appear in Fig. 12. Note that similar behavior was noted in several experimental runs, with this one picked for illustrative purposes.

We notice in particular that while the dMPS appears to perform exceedingly well according to its own biased metric, its behaviour with respect to the true objective is concerning, as it dips to its lowest point around iteration 500, before rebounding and leveling off at a higher level, around iteration 800. Most importantly, this seems to indicate that the biased objective is running counter to the true objective in some subtle way.

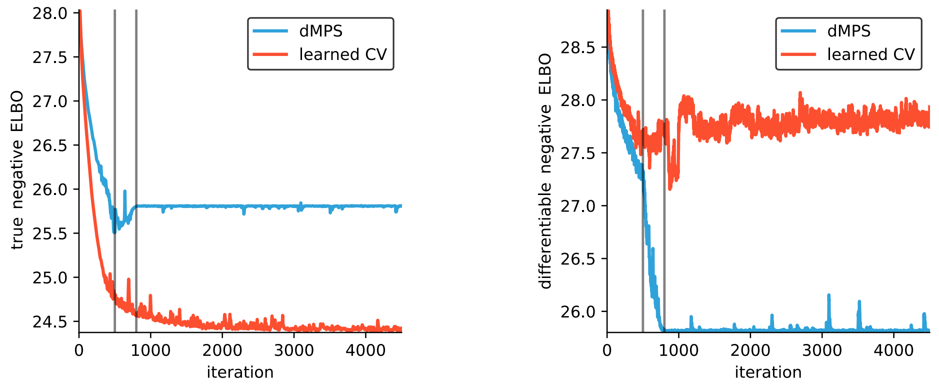
We can pick this observation apart by taking the entire 512 element posterior probability tensor, and compare it to the approximate posteriors. We plot the true probability against the approximation for all tensor elements individually in the plots in Fig. 13. The two first subfigures correspond to the iterations marked with vertical lines in Fig. 12, and the third subfigure is at convergence. We note that at iteration 500 the approximation is somewhat sound, with the approximation roughly in the right range of values and a good fit to the highest value mode. Then at iteration 800, we see that pretty much all of the probability mass is concentrated at the single highest mode. The unbiased gradient meanwhile finds a reasonable approximation, especially with respect to the more significant high probability states.

6.4 Experiments on Zachary's karate network

In this section we approximate the posterior of a stochastic block model applied to the venerable karate graph (Zachary, 1977). This is a 34 vertex social graph with $K = 2$ gold standard communities. We use uniform priors, $\alpha_k = 1$, $a_{k\ell} = 1$ and $b_{k\ell} = 1$ since informed priors can have a large impact on the posterior's concentration.

The undirected graph has a total of $\binom{34}{2} = 561$ possible edges, and we assume that we only observe 400 of these vertex pairings, leaving a test set of 161 to allow us to evaluate the model by its prediction on held-out data.

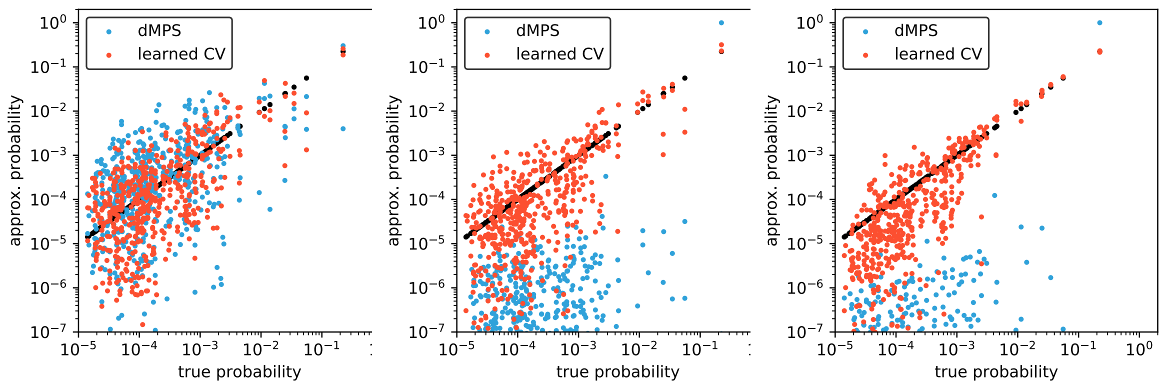
As a baseline, we also solved it using a mean-field approximation, employing KL-corrected bounds (Hensman et al., 2012) and an out of the box L-BFGS optimization to efficiently find local optima. We ran it 500 times to establish the global optimum with some certainty.



(a) The true ELBO (not stochastically estimated).

(b) The stochastic loss function employed by the dMPS.

Figure 12: Learning curves for two MPS models: Differentiable matrix product state (dMPS) and RELAX with control variates (learned CV) with biased and unbiased gradients respectively. We report the true loss, and the implicit loss of the biased method. Horizontal lines correspond to snapshots at iteration 500 and 800 in Fig. 13.



(a) Iteration 500.

(b) Iteration 800.

(c) At convergence.

Figure 13: Difference between the estimated q and the true posterior at all 512 positions of the probability tensor for differentiable matrix product state (dMPS) and RELAX with control variates (learned CV). Black points correspond to the true posterior.

6.4.1 Initialization and Warm Starts

A central issue with graphs of any meaningful size is that due to the combinatorial explosion of possible states, they can be quite sensitive to initialization. One advantage of our problem is that we can in many cases easily find local optima or efficiently computable approximations, in particular in the settings where we can calculate the analytical ELBO for a mean-field model. To each mean-field solution $q_i(\boldsymbol{X}) = \prod_{n=1}^{N} q_{i,n}(\boldsymbol{x}_n)$, we can associate a tensor $\mathcal{T}_i$ which is the outer product of the marginal probability vectors, as in equation (2).

We could combine all the mean-field solutions in a mixture $\bar{\mathcal{T}} = \frac{1}{S}\sum_{i=1}^{S}\mathcal{T}_i$, and compute the norm

$$\left\langle \mathcal{T} - \bar{\mathcal{T}}, \mathcal{T} - \bar{\mathcal{T}} \right\rangle = \langle \mathcal{T}, \mathcal{T} \rangle + \frac{1}{S^2}\sum_{i,j=1}^{S} \langle \mathcal{T}_i, \mathcal{T}_j \rangle - \frac{2}{S}\sum_{i=1}^{S} \langle \mathcal{T}, \mathcal{T}_i \rangle ,$$

which is computable as all of the inner products are expectations per equation (11). Unfortunately, due to the high dimensionality of the tensors, this problem appears to be numerically problematic as distances break down due to the curse of dimensionality.

As a light proxy, we propose using the sum of log expectations,

$$\sum_{i=1}^{S} \log \langle \mathcal{T}, \mathcal{T}_i \rangle .$$

Intuitively, if $\mathcal{T}_i$ is close to a one-hot encoding, maximizing the expectation $\langle \mathcal{T}, \mathcal{T}_i \rangle$ encourages the model to put as much mass as possible on that index. Summing over the expectations with respect to all states aims to then find a compromise that puts mass on all the indices, but if the expectations are just summed the optimal solution will be to put maximum mass on the index with highest weight. The logarithm tries to balance this by making it disproportionately disadvantageous to put zero mass on any of the states. This target is amenable to off-the-shelf optimizers like BFGS. Since there is still some chance of collapsing onto the S states, we recommend performing early stopping after 30 steps.

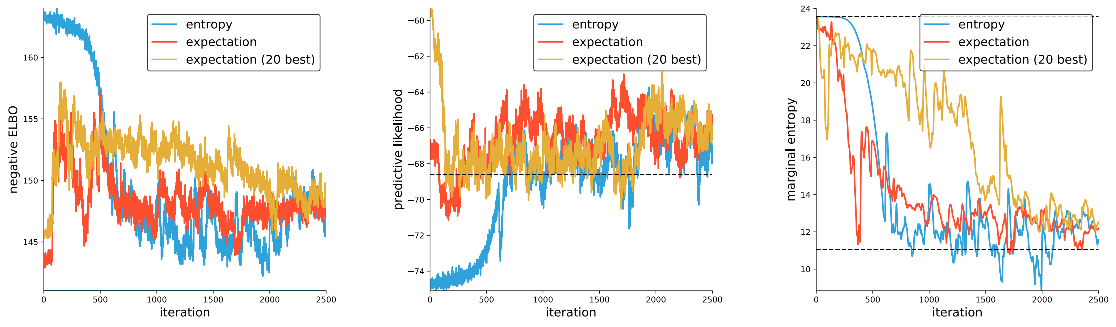
Another alternative to random initialization is to start in the maximum entropy state. The entropy of the entire MPS is difficult to calculate, but since we can compute the marginals efficiently we can use the marginal entropy as a proxy, encouraging that no label preferred a priori.

In Fig. 14 we visualize the learning curves generated by initializing using these two strategies. We also consider a boosted version of the expectation-based initialization, where we only maximize the objective with respect to the 20 entries with highest predictive likelihood.

In Fig. 14a we have inverted the ELBO to get a minimization problem, and we see that all curves implement a decreasing behavior as expected, although the expectation strategy behaves oddly in the initial transient phase. The initial state of the entropy initialization is not surprisingly a rather poor initialization with respect to the ELBO, but it seems to decrease to the level of the others gracefully.

The entropy initialization likewise under-performs on predictive likelihood in Fig. 14b.The best predictive likelihood achieved by any of the mean-field optima is -59 which is achieved by the boosted method, which was specifically designed to be close to that state, but we see that this is not a stable point and the boosted strategy dives down to the same level as the other curves. This is in line with the mean-field solution with the best bound having a rather mediocre predictive performance, so this is indicative of a model mismatch.

From the marginal entropy plot in Fig. 14c, we see that all of the states start in fairly entropic solutions, despite having very different empirical behavior. We also see that the inference procedure drives all of them towards low entropy configurations, with the lower



(a) ELBO for the different initialization strategies.

(b) Predictive likelihood on test set. Line denotes the mean-field model with best ELBO.

(c) Marginal entropy. Lines denote maximum possible entropy (top) and entropy where each marginal puts 0.9 on a single entry.

Figure 14: Learning curves for three different initialization strategies based on maximizing the marginal entropy (labelled entropy), maximizing the sum of log expectations overall (expectation) or for the 20 entries with highest predictive likelihood (expectation 20 best).

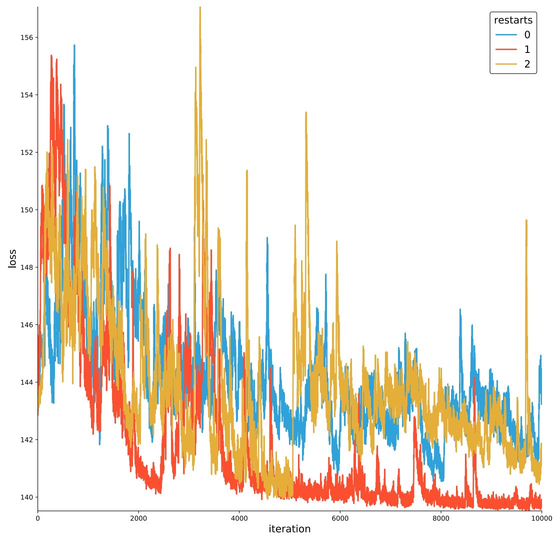


Figure 15: Loss as a function of iteration number for three restarts of rank 16 MPS on Zachary's karate network.

reference line being the entropy of a mean-field model where every marginal puts 0.9 probability mass on a single entry, and divides the rest evenly. This is discouraging, as we had hoped the flexibility of the MPS would allow us to find solutions with high marginal entropy.

6.4.2 Induced Covariance

Next, we ran a rank 16 MPS with canonical cores, warm starts using the expectation strategy, and AMSgrad with a 0.01 learning rate. We plot three restarts in Fig. 15. We note that run 1 surpasses the other two runs by a fair margin, so we selected that for further inspection. As a rule, we generally observe quite noisy transient phases during optimization, but eventually it levels out.

We have no good easily computable metric to evaluate the degree to which dependencies have been encoded in the approximate posterior. To get a local measure we will consider the covariance matrix of each one-hot categorical variable pair $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n'}$, with element (k, k') of the covariance matrix given by,

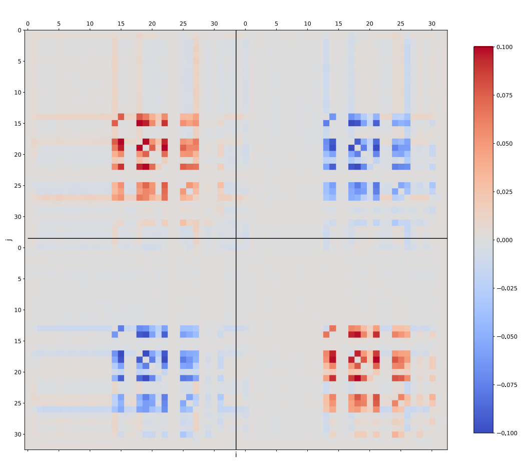
$$\operatorname{Cov}(x_{nk}, x_{n'k'}) = \mathbb{E}[x_{nk}x_{n'k'}] - \mathbb{E}[x_{nk}]\,\mathbb{E}[x_{nk'}].$$

If $k = k'$, and the respective covariance element is positive, this means that those two points are likely to get sorted into the same cluster. We can take this a step further by computing the co-location matrix,

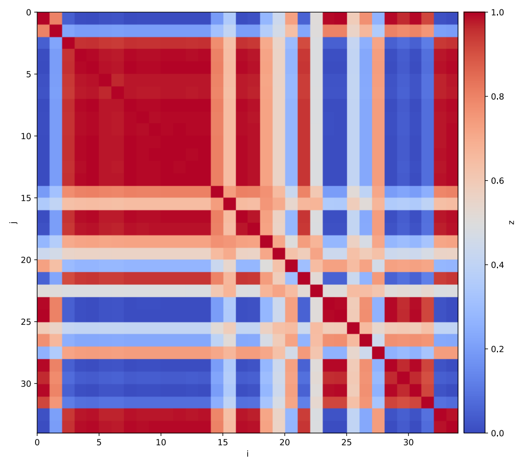
$$\mathbb{E}\Big[\boldsymbol{X}\boldsymbol{X}^\top\Big],$$

where $[\boldsymbol{X}]_{nk} = x_{nk}$. Given any sample instantiation $\hat{\boldsymbol{X}}$, the quantity $\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^\top$ gives a binary matrix where each element is 1 if two elements share a label in $\hat{\boldsymbol{X}}$. Taking the expectation then gives us the co-location probability of how likely two elements are clustered on average. Both of these quantities can be estimated. The expectations here are computable using the tensor train framework, as we can compute the marginal distribution of the two variables, or use the tower property to express it in terms of a conditional expectation. It is often simpler to just sample it though, which is what we will do going forward, using 10 000 samples.

We visualize the covariance matrix in Fig. 16a. The blocks are indexed by label, so the upper left block is covariance between x_{i0} and x_{j0}, the upper right block is between elements x_{i0} and x_{j1}, and so on. For saliency, we have deducted the corresponding covariance matrix of the marginal distribution, which is only non-zero on the diagonals of each of these blocks. So all of the observed covariance is due to the higher-rank modeling. We note that one group exhibits relatively high positive correlation, making it likely that these elements group together. When we look to the corresponding co-location matrix in Fig. 16b, we see that there are some elements that almost always get clustered together, but we also see that the elements for which we observed high covariance are the elements that are somewhat uncertainly labeled. The strong blocks correspond to elements that have highly concentrated marginals, giving them an almost certain assignment. This is a consequence of most of the local optima we find concentrating around a single symmetry mode. The uncertain elements then get randomly assigned to each of these blocks, but the covariance tells us that they are assigned as a group. This is not the case for mean-field models, where all uncertain elements must be randomly assigned without consideration for other elements, which encourages these rank-1 mean-field solutions to collapse unto a specific hard clustering.



(a) Covariance matrix for restart 1. Blocks corresponds to labels. Marginal covariance has been deducted.



(b) Co-location matrix for restart 1.

Figure 16: Covariance and co-location matrices for restart 1, estimated using 10 000 samples.

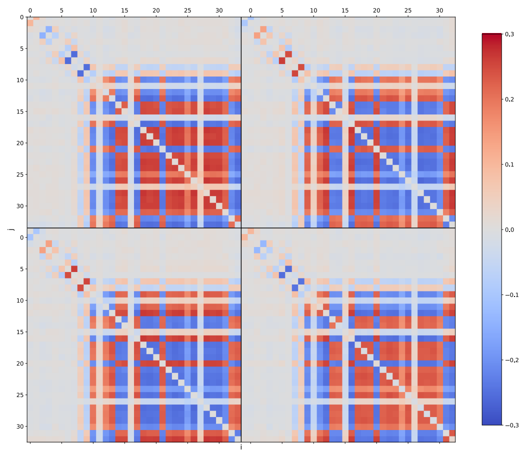
6.4.3 Permutation Invariance

We now run a similarly sized MPS with a set of permutation-invariant cores. As we are working with $K = 2$ we can use the bit-flip symmetry core (Huckle et al., 2013), although we will have to explicitly normalize it as we do not have a canonical self-normalizing version of that core available. After 20 000 iterations, we get the covariance and co-location plots of Fig. 17.

Contrasted with the canonical core set from before, the covariances exhibited in Fig. 17a are a lot stronger; note that the color scale has been extended to cover the new range of values. Compared to Fig. 17b, we also note that the co-location is near-identical to that of a covariance block, which is of course a direct consequence of the permutation-invariant representation modeling everything through covariance alone. Another consequence is that the diagonal blocks of the covariance matrix are now identical, since the two labels are interchangeable. More distressingly, we note that the upper left corner seems to hardly be modeled at all, which is especially odd as this part of the model was assigned labels with very high certainty. This could be an issue with model capacity, but it could also be a local optima problem: if we want to put the points of the upper left corner in one cluster, it might be problematic if the K clusters have already been "spent" clustering the lower left corner in a partition that is not consistent with the remaining points. Either way, it should be clear that permutation-invariant cores have a large effect on the produced posterior approximations and that the choice of core leads to qualitative differences.

6.5 Ambiguity and comparisons with sampling methods

The most common strategy for approximating discrete models is to employ sampling methods. While methods like Hamiltonian Monte Carlo are seeing widespread use, they rely


(a) Covariance matrix for the permutation-invariant core. Blocks corresponds to labels. Marginal covariance has been deducted.

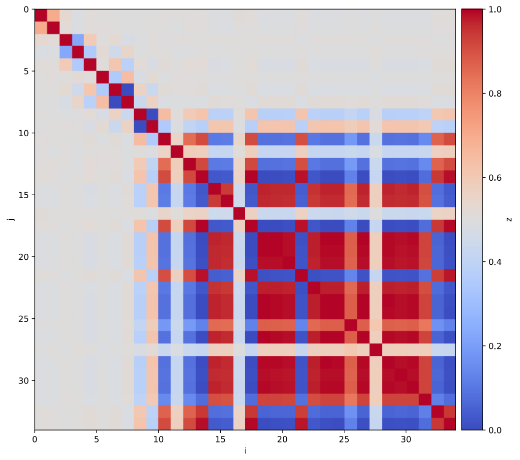

(b) Co-location matrix for the permutation-invariant core.

Figure 17: Covariance and co-location matrices for the permutation-invariant kernel, estimated using 10 000 samples. Note that the color scale for covariance has been extended.

on continuous sample spaces, making them inapplicable in the discrete setting; instead, discrete sampling methods are often variants of the Gibbs sampler, which can be both fast and effective in many practical situations. Unfortunately, it can also have exponential mixing time in others and can get stuck in local configurations. Alternatives like split/merge samplers (Jain and Neal, 2004) have been employed to help with these issues, at the cost of added complexity, but navigating a large combinatorial space successfully remains a matter of some luck (Albers et al., 2013; De Sa et al., 2015).

Taking the karate graph as our test case again, we run a collapsed Gibbs sampler for better mixing (Liu, 1994) and compare it against MPS models of rank 1, 2, 4 and 8. We run each for 15000 steps and average over 5 random restarts. We run the Gibbs sampler for 1000 warmup iterations and then for 15000 iterations, using all non-warmup samples to calculate a log-predictive likelihood; we repeat 10 times and average to get our benchmark.

The learning trajectories of the MPS's are depicted in Fig. 18, where we first note that they all eventually surpass the benchmark. Second, the lower-rank models actually terminate at a higher likelihood than the higher-rank models, although this is likely just a consequence of the larger more complex models being harder to optimize, a hypothesis supported by the generally slower convergence and more erratic trajectory of the high-rank models.

As a further challenge, we propose an ambiguous community detection problem illustrated by a graph with 4 vertices, one for each community, arranged in a square pattern as depicted in Fig. 19. Each community has a high probability of 0.8 of a self-connection, and is well-connected with its neighbours with probability 0.5 of a connection. The community diagonally across from it in the square is weakly connected with probability 0.2. If we

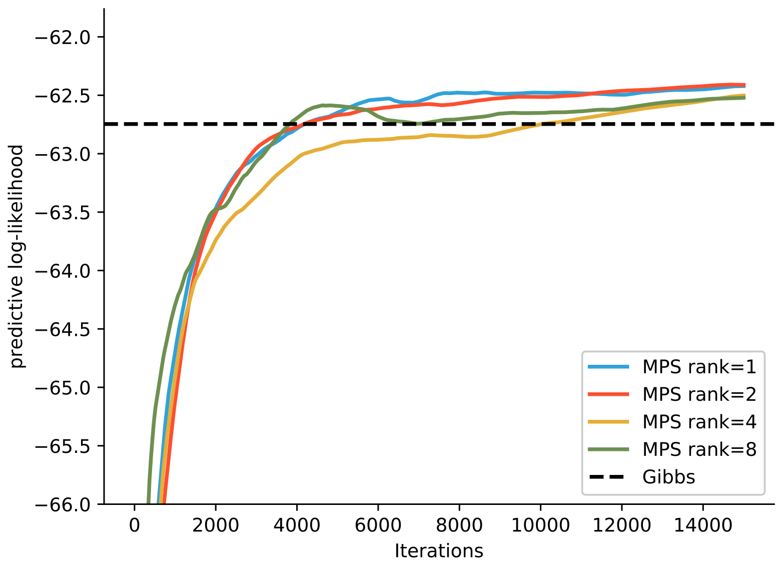


Figure 18: MPS models, averaged over 5 restarts, competing against a benchmark on the karate network set by averaging over the log-predictive likelihood of 10 Gibbs sampler runs.

consider a misspecified scenario where we look for only 2 communities, then the problem becomes ambiguous as any split into 2 times 2 neighbours is equivalent. We generate data from this model artificially, taking exactly 10 points from each community and searching for a graph that is as close as possible to having the correct edge statistics to rule out statistical flukes.

Running a collapsed Gibbs sampler on this new case, we take 1000 warm-up samples followed by 10000 samples that we then use to compute co-location plots like before. Eight such runs are illustrated in Fig. 20a. We have laid out the communities in clockwise order with respect to the graph in Fig. 19 so that adjacent blocks are also neighbours with 0.5 probability of connecting. These runs demonstrate that the Gibbs sampler does get stuck in local configurations as there are evidently at least two qualitatively different patterns, with runs 5 and 6 exemplifying one of them. Run 4 demonstrates some uncertainty, but never connects the second and third community (in the center, with negative (blue) correlation).

Our method, illustrated in Fig. 20b, is unfortunately not in a much better position, and also seems to get stuck in one of the symmetric modes. This example used a model of rank 8 which should be able to capture multiple modes, at least partially, but the stochastic optimization routine used in this paper invariably converges to a local mode. As we know that the MPS can indeed model any non-negative tensor, including the ambiguous graph model presented here, this failure demonstrates that there are remaining challenges with optimization. One option is that the optimization employed is flawed. It is recognized that stochastic gradient descent can have an intrinsic regularizing effect, which might bias it towards certain modes and solutions. Since we evaluate the loss using points sampled from

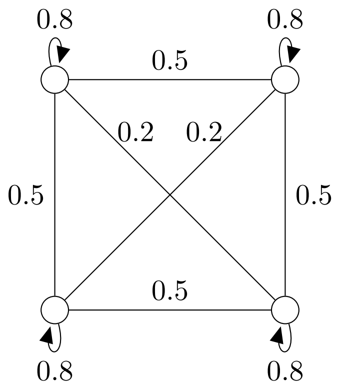
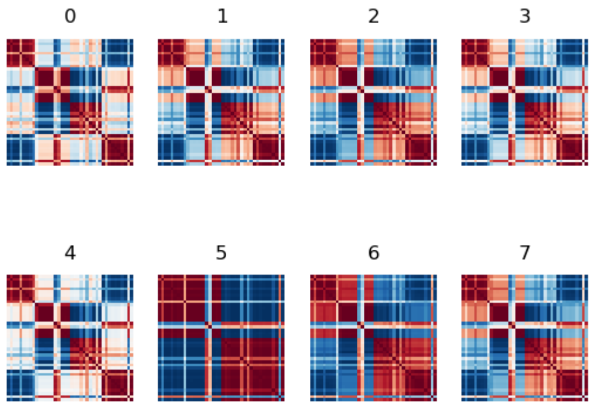
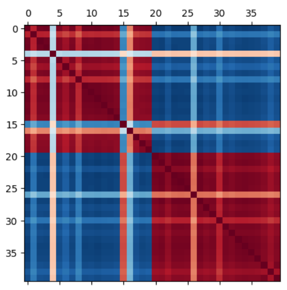


Figure 19: The ambiguous graph community structure. Vertices depict communities, edge labels denote connection probabilities.



(a) Gibbs sampling



(b) MPS

Figure 20: Co-location diagrams for 8 runs of 10000 samples (1000 warmup) of a Gibbs sampler on the ambiguous graph problem. White is 0.5, red is above, blue is below. Note the two distinct patterns, reflecting that the sampler is not mixing adequately. The MPS approach likewise collapses to a mode.

the model itself, it seems plausible that SGD might ignore points with near-zero probability of being sampled under the initial q, leading to SGD-induced mode collapse.

7. Related Work and Further Reading

As matrix product states have seen wide-spread use in quantum mechanics, there is a large literature on the topic. Unfortunately, for readers outside the physics community, the presentation can be impenetrable, which is one of the motivating reasons behind the present manuscript. For general introductions to the topic, we recommend Schollwöck (2011) which covers the mechanics and mathematics well and is quite pedagogical. The thoroughly illustrated introduction by Bridgeman and Chubb (2017) complements it nicely. For some more esoteric and physics-oriented introductions, interested readers might want to consult Perez-García et al. (2007); Orús (2014b); Biamonte and Bergholm (2017) or the review on applications to many-body quantum systems by Cirac et al. (2020).

While we have focused on solving non-linear variational inference problems, the physics community has developed good solutions for solving rayleigh quotients and linear objectives. The central workhorse is the DMRG algorithm (White, 1993) that solves the objective by optimizing over one core tensor at a time, which reduces the optimization to an eigenvector problem; see also (Hubig et al., 2015; Stokes and Terilla, 2019). In the extended two-site DMRG (Holz et al., 2012) each step optimizes over two cores jointly, which makes it possible to apply SVD to deduce the appropriate rank of the core tensors dynamically. The introductions Bridgeman and Chubb (2017) and Schollwöck (2011) both offer excellent introductions to the DMRG algorithm. The ideas from the DMRG algorithm can also be applied to tensor trains (Holtz et al., 2012).

In the tensor literature, the tensor train has also been growing in popularity, starting with its publication in Oseledets (2011). The literature on tensor trains has proven less relevant for this presentation, due to the more algebraic focus. We note that other results from the physics literature have started to percolate over, such as the tensor ring decomposition (Zhao et al., 2016), which is known as a MPS with periodic boundary condition in quantum mechanics (Schollwöck, 2011).

As noted previously, we are not the first to consider the relationship between matrix product states and probabilistic models. Arguably, this relationship is a bit of a false dichotomy, as its application in quantum mechanics means that it has always had a statistical interpretation, by virtue of the way a quantum mechanical wave function is related to a probability distribution by way of Born's rule. The more explicit connections to probabilistic models, and graphical models in particular, were pioneered by Novikov et al. (2014) and culminated recently in a duality result (Robeva and Seigal, 2018).

There have also been a several ventures into applying tensor trains and matrix product states in machine learning applications, including generative modeling and density estimation (Han et al., 2018; Cheng et al., 2019), anomaly detection (Wang et al., 2020), image classification (Stoudenmire and Schwab, 2016; Selvan et al., 2020; Cheng et al., 2021), language modeling (Pestun and Vlassopoulos, 2017), sequence/time series analysis (Miller et al., 2020; da Costa et al., 2021), and optimization (Alcazar and Perdomo-Ortiz, 2021). Theoretical results on the generalization error (Bradley et al., 2020) and the expressive power (Glasser et al., 2019) in tensor networks are also being developed. Finally we highlight the relation of MPS models to sum-product networks (Poon and Domingos, 2011; Sanchez-Cauce et al., 2021; Peharz et al., 2019) which generalizes all tractable probabilistic graphical models.

8. Conclusion

In this paper, we have attempted to lay the groundwork for how matrix product states can be used as an approximate model in variational inference. Part of our contribution is that this text should serve as a pedagogical gateway for people in the machine learning community interested in this topic, by offering an introduction to the topic, relevant references to the existing literature, and translating some of the physics-oriented results into a notation that is more relatable to machine learning researchers.

The main challenge of integrating the MPS into a variational inference setting is finding ways to estimate gradients in a robust manner. As a first step, this requires finding a differentiable representation for the MPS, which is a bit different from other papers where the model is often learned by way of repeated application of singular value decompositions like in the tensor train algorithm we originally described in section 2.3 (Oseledets, 2011), or the density-matrix renormalization group (DMRG) method popular in physics (Schollwöck, 2011). We note that Han et al. (2018) propose a hybrid approach, combining the iterative procedure of the DMRG, gradient steps, and SVD. This is one of the things that could be pursued in future work, although we were worried that local updates might make the model more likely to converge to local modes. This worry is motivated by the passing similarity with Gibbs sampling, which can become trapped due to its inability to make global changes (Jain and Neal, 2004).

There are some remaining mysteries when it comes to representations. We consider a differentiable $\Gamma\Lambda$-representation to be an important target for future research as it would make marginalization operations exceedingly expeditious, and allow direct parametric control over the form of the marginals. Further exploration of symmetry representations is another thing we believe would be fruitful; although modeling symmetry modes wastes some of the model's capacity, we conjecture that factoring in symmetry constraints reduces the search space and thus helps with exploring the model space. One could also attempt to find representations that only concentrate on a single symmetry mode, but we have not looked into this. We found that many of our representations depended on orthogonal matrices with their own differentiable parametric forms, where we have one unresolved issue as the orthogonal matrices do not form a connected manifold, i.e. there does not exist a parametric representation capable of modeling all orthogonal matrices with both positive and negative determinant (Shepard et al., 2015). A final missing piece is the opportunity to use complex valued representations which is standard in the physics community. One might hope that using complex numbers translates directly into added model capacity, simply by virtue of the increase in parameters, without increasing the rank. Additionally, since we only need to model the square root of the true probability tensor, allowing complex values gives us an infinite number of alternative solutions; when the tensor train is real-valued, we can flip signs in the elements of the tensor train without affecting the square. With complex-valued tensor trains we can multiply any element with any root of unity without affecting it. On the other hand, the constructive argument means that it always suffices to use real-valued tensors.

Optimization remains a challenge. While we found that our stochastic gradients performed admirably considering the difficulty of the problem, more work could be put into finding good warm start procedures based on e.g. locally optimal mean-field solutions like we used. Another avenue would be to extend the coordinate-ascent updates of DMRG and Han et al. (2018) to the full variational problem. Finally, we think there could be merit in pursuing Riemannian optimization as the tensor trains span a sub-manifold of the space of all tensors, and the gauge invariance means that we are often using highly redundant parameterizations. Some progress has already been made in this direction (Steinlechner, 2016). Numerical issues also remain problematic, as we are forced to work in the non-log domain to exploit the MPS structure. Whether this can be fully circumvented by technical means is an open question. One could also imagine hybrid models where we also model the log-probability with an MPS in conjunction with the normalized model.

The MPS methods are designed to scale well, although the $\mathcal{O}(R^3)$ scaling in the maximum rank eventually gets prohibitive. Scaling in N is linear for many operations, which is a big difference from most other factorization schemes. The trade-off is that few of the MPS and TT operations come cheaper than $\mathcal{O}(N)$, e.g. both evaluation and sampling scale as $\mathcal{O}(N)$. We did a full-fledged implementation in Tensorflow, parallelizing where possible, but performance could still be quite slow. Additionally, running it on a GPU often made the whole thing slower, despite most operations being standard linear algebra routines. It is possible that the large computational graphs and the long chains of small matrix operations is a poor fit for Tensorflow. Also, from inspection, the main bottleneck appears to be the existing implementation of Einstein summation, which has not been fully optimized in Tensorflow yet. In summary, it should be possible to make a high-performance versions of the MPS.

There is also room for exploring some of the more advanced architectures from the tensor networks literature. The tensor ring (Zhao et al., 2016), known as an MPS with periodic boundary conditions in the physics literature (Schollwöck, 2011), is a straightforward extension of the tensor train which makes the tensor invariant to cyclic reordering of the cores, but it lacks a canonical representation, as well as the efficient normalization scheme of equation (13), forcing the explicit computation of Kronecker products of the cores. Even more advanced architectures like PEPS and MERA could also prove useful, although few of the analytical formulas available for the MPS carry over (Orús, 2014b,a). Another interesting avenue is the idea of implementing the cores as their own tensor trains, possibly allowing the extension of the MPS ideology to larger ranks (Hübener et al., 2010). Note that tensor trains can also be used to speed up standard matrix multiplications considerably (Oseledets, 2011; Novikov et al., 2015).

Appendix A. REBAR for categoricals

The current best method for constructing a low-variance unbiased discrete gradient estimator is the REBAR estimator which constructs a control variate with coupled randomness. If we denote the discrete random variable b and the cost function f, then the target is to estimate $\nabla \mathbb{E}[f(b)]$. Given a source of randomness $p(\epsilon)$ and a continuous reparametrization $z = g(\epsilon, \theta)$ such that $b = H(z)$ after passing through a gating function H, REBAR builds a control variate using a dummy reparametrization $z' \sim p(z|b)$ likely to have high correlation with the original.

If b is a categorical variable $\mathrm{Cat}(\boldsymbol{\alpha})$, a natural reparametrization follows via the Gumbel-max trick. The procedure is simply:

$$\begin{aligned} \epsilon_i &\sim \mathcal{U}[0,1], \\ z_i &= \ln \alpha_i - \ln(-\ln(\epsilon_i)) \sim \mathrm{Gumbel}(\ln \alpha_i, 1), \\ b &= \arg\max(\boldsymbol{z}). \end{aligned}$$

Going the other way and sampling from $p(\boldsymbol{z}|b)$ is slightly more complicated. It turns out that

$$\begin{aligned} z_b &\sim \mathrm{Gumbel}\left(\ln \sum_{i=1}^{N} \alpha_i, 1\right), \\ z_i | z_b &\sim, \mathrm{TruncatedGumbel}\left(\ln \alpha_i, 1, z_i \leq z_b\right), \quad \forall i \neq b. \end{aligned}$$

The Gumbel has pdf and cdf

$$f_G(x; \mu, 1) = \exp(-(x-\mu) - \exp(-(x-\mu))), \tag{44}$$

$$F_G(x; \mu, 1) = \exp(-\exp(-(x-\mu))). \tag{45}$$

so the truncated cdf is conveniently just

$$F_{TG}(x; \mu, 1, x \leq t) = \frac{F_G(x; \mu, 1)}{F_G(t; \mu, 1)}.$$

Finding a reparametrization then just becomes a question of applying the inverse transform sampler of the truncated Gumbel to a uniform variable u, and as

$$F_G^{-1}(p; \mu, 1) = \mu - \ln(-\ln(p)),$$

we have

$$x = F_{TG}^{-1}(u) = F_G^{-1}(F_G(t)u) = \mu - \ln(-\ln F_G(t) - \ln u).$$

If we instead want to reparametrize in terms of a Gumbel variable z, we can use that $u = F_G(z)$ (running the inverse transform sampler in reverse) and then substitute in to get

$$x = \mu - \ln(-\ln F_G(t) - \ln F_G(z)) = \mu - \ln\left(e^{-(t-\mu)} + e^{-(z-\mu)}\right).$$

which can finally be reduced to $x = -\ln\left(e^{-t} + e^{-z}\right)$.

References

K. J. Albers, A. L. A. Moth, M. Mørup, and M. N. Schmidt. Large scale inference in the infinite relational model: Gibbs sampling is not enough. In *Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, Sept. 2013. doi: 10.1109/MLSP.2013.6661904.

J. Alcazar and A. Perdomo-Ortiz. Enhancing combinatorial optimization with quantum generative models, Jan. 2021. arXiv:2101.06250.

R. Bailly. Quadratic weighted automata: Spectral algorithm and likelihood maximization. *Asian Conference on Machine Learning (ACML)*, 20:147–162, 2011.

B. Balle, P. Panangaden, and D. Precup. A canonical form for weighted automata and applications to approximate minimization. In *Symposium on Logic in Computer Science (LICS)*, volume 2015-July, 2015. doi: 10.1109/LICS.2015.70.

J. Biamonte and V. Bergholm. Tensor networks in a nutshell, July 2017. arXiv:1708.00006.

C. Bishop. *Pattern recognition and machine learning.* Springer, New York, 2006.

T.-D. Bradley, E. M. Stoudenmire, and J. Terilla. Modeling sequences with quantum states: a look under the hood. *Machine Learning: Science and Technology*, 1(3), 2020. doi: 10.1088/2632-2153/ab8731.

J. C. Bridgeman and C. T. Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22): 223001, May 2017. doi: 10.1088/1751-8121/aa6dc3.

S. Cheng, L. Wang, T. Xiang, and P. Zhang. Tree tensor networks for generative modeling. *Physical Review B*, 99(15), 2019. doi: 10.1103/PhysRevB.99.155131.

S. Cheng, L. Wang, and P. Zhang. Supervised learning with projected entangled pair states. *Physical Review B*, 103(12), 2021. doi: 10.1103/PhysRevB.103.125117.

I. Cirac, D. Perez-Garcia, N. Schuch, and F. Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, and theorems, Nov. 2020. arXiv:2011.12127.

K. Conrad. Generating sets. Technical report, University of Conneticut, Department of Mathematics, 2013. URL `kconrad.math.uconn.edu/blurbs/grouptheory/genset.pdf`.

M. N. da Costa, R. Attux, A. Cichocki, and J. M. T. Romano. Tensor-train networks for learning predictive modeling of multidimensional data, Mar. 2021. arXiv:2101.09184.

C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré. Rapidly mixing gibbs sampling for a class of factor graphs using hierarchy width. In *Advances in Neural Information Processing Systems.* 2015.

A. J. Ferris and G. Vidal. Perfect sampling with unitary tensor networks. *Physical review. B, Condensed matter*, 85(16):165146, Apr. 2012. doi: 10.1103/PhysRevB.85.165146.

I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and I. Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. In *Advances in Neural Information Processing Systems*, 2019.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations (ICLR)*, 2018.

J. E. Gumbel. Statistical theory of extreme values and some practical applications. *National Bureau of Standards Applied Mathematical Series*, 33, 1954.

B. Haasdonk and H. Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine learning*, 68(1):35–61, July 2007. doi: 10.1007/s10994-007-5009-7.

Z. Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3), 2018. doi: 10.1103/PhysRevX.8.031012.

J. Hensman, M. Rattray, and N. D. Lawrence. Fast variational inference in the conjugate exponential family. In *Advances in Neural Information Processing Systems*. 2012.

S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal of Scientific Computing*, 34(2): A683–A713, Jan. 2012. doi: 10.1137/100818893.

S. Holz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2): A683–A713, 2012. doi: 10.1137/1008188.

E. Hoogeboom, J. W. Peters, R. van den Berg, and M. Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, 2019.

R. Hübener, V. Nebendahl, and W. Dür. Concatenated tensor network states. *New journal of physics*, 12(2):025004, Feb. 2010. doi: 10.1088/1367-2630/12/2/025004.

C. Hubig, I. P. McCulloch, U. Schollwöck, and F. A. Wolf. A strictly single-site dmrg algorithm with subspace expansion. *Physical Review B*, 91:155115, 2015. doi: 10.1103/PhysRevB.91.155115.

T. K. Huckle, K. Waldherr, and T. Schulte-Herbrüggen. Exploiting matrix symmetries and physical symmetries in matrix product states and tensor trains. *Linear and Multilinear Algebra*, 61(1):91–122, Jan. 2013. doi: 10.1080/03081087.2012.663371.

M. C. Hughes and E. Sudderth. Memoized online variational inference for dirichlet process mixture models. In *Advances in Neural Information Processing Systems*. 2013.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural computation*, 12(6):1371–1398, June 2000. doi: 10.1162/089976600300015411.

S. Jain and R. M. Neal. A Split-Merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of computational and graphical statistics*, 13(1):158–182, 2004. doi: 10.1198/1061860043001.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, Apr. 2017.

D. P. Kingma and M. Welling. Auto-Encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, 2009. doi: 10.1137/07070111X.

T. Kuśmierczyk and A. Klami. Reliable categorical variational inference with mixture of discrete normalizing flows, June 2020. arXiv:2006.15568.

S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B*, 50(2):157–224, 1988. doi: 10.1111/j.2517-6161.1988.tb01721.x.

J. S. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, Sept. 1994. doi: 10.1080/01621459.1994.10476829.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

G. A. Miller. On the groups generated by two operators. *Bulletin of the American Mathematical Society*, 7(10):424–426, 1901.

J. Miller, G. Rabusseau, and J. Terilla. Tensor networks for probabilistic sequence modeling, Mar. 2020. arXiv:2003.01039.

T. Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005.

A. Novikov, A. Rodomanov, A. Osokin, and D. Vetrov. Putting MRFs on a tensor train. In *International Conference on Machine Learning (ICML)*, pages 811–819, Jan. 2014.

A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*. 2015.

K. Nowicki and T. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001. doi: 10.1198/016214501753208735.

R. Orús. Advances on tensor network theory: symmetries, fermions, entanglement, and holography. *The European physical journal. B*, 87(11):280, Nov. 2014a. doi: 10.1140/epjb/e2014-50502-9.

R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158, 2014b. doi: 10.1016/j.aop.2014.06.013.

I. Oseledets. Tensor-Train decomposition. *SIAM Journal of Scientific Computing*, 33(5): 2295–2317, Jan. 2011. doi: 10.1137/090752286.

R. Peharz, A. Vergari, K. Stelzner, A. Molina, X. Shao, M. Trapp, K. Kersting, and Z. Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2019.

D. Perez-García, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(5-6):401–430, 2007. doi: 10.26421/qic7.5-6-1.

V. Pestun and Y. Vlassopoulos. Tensor network language model. Oct. 2017. arXiv:1710.10248.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *International Conference on Computer Vision (ICCV)*, 2011. doi: 10.1109/ICCVW.2011.6130310.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Artificial Intelligence and Statistics (AISTATS)*, 2014.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015.

E. Robeva and A. Seigal. Duality of graphical models and tensor networks. *Information and Inference: A Journal of the IMA*, June 2018. doi: 10.1093/imaiai/iay009.

R. Sanchez-Cauce, I. Paris, and F. J. D. Vegas. Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. doi: 10.1109/TPAMI.2021.3061898.

U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of physics*, 326(1):96–192, Jan. 2011. doi: 10.1016/j.aop.2010.09.012.

R. Selvan, S. Ørting, and E. B. Dam. Locally orderless tensor networks for classifying two- and three-dimensional medical images. *Journal of Machine Learning for Biomedical Imaging. Special Issue: Medical Imaging with Deep Learning*, 2020. arXiv:2009.12280.

R. Shepard, S. R. Brozell, and G. Gidofalvi. The representation and parametrization of orthogonal matrices. *The journal of physical chemistry. A*, 119(28):7924–7939, July 2015. doi: 10.1021/acs.jpca.5b02015.

S. Singh, R. N. C. Pfeifer, and G. Vidal. Tensor network decompositions in the presence of a global symmetry. *Physical review. A*, 82(5):050301, Nov. 2010. doi: 10.1103/PhysRevA.82.050301.

S. Singh, R. N. C. Pfeifer, and G. Vidal. Tensor network states and algorithms in the presence of a global U(1) symmetry. *Physical Review B: Condensed Matter and Materials Physics*, 83(11):115125, 2011. doi: 10.1103/PhysRevB.83.115125.

M. M. Steinlechner. *Riemannian optimization for solving high-dimensional problems with low-rank tensor structure*. PhD thesis, École polytechnique fédérale de Lausanne, 2016.

J. Stokes and J. Terilla. Probabilistic modeling with matrix product states. *Entropy*, 21 (12), 2019. doi: 10.3390/e21121236.

E. M. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, 2016.

X. Sun and C. Bischof. A Basis-Kernel representation of orthogonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1184–1196, 1995. doi: 10.1137/S0895479894276369.

Y. W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*. 2007.

D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems*, 2019.

G. Tucker, A. Mnih, C. J. Maddison, D. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, 2017.

G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902, Oct. 2003. doi: 10.1103/PhysRevLett.91.147902.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, Jan. 2008. doi: 10.1561/2200000001.

J. Wang, C. Roberts, G. Vidal, and S. Leichenauer. Anomaly detection with tensor networks, June 2020. arXiv:2006.02516.

A. Weichselbaum. Non-abelian symmetries in tensor networks: A quantum symmetry space approach. *Annals of Physics*, 327(12), 2012. doi: 10.1016/j.aop.2012.07.009.

S. R. White. Density-matrix algorithms for quantum renormalization groups. *Physical review. B, Condensed matter*, 48(14):10345–10356, Oct. 1993. doi: 10.1103/PhysRevB.48.10345.

Z. Xu, Y. Ke, and Y. Wang. A fast inference algorithm for stochastic blockmodel. In *International Conference on Data Mining (ICDM)*, pages 620–629, 2014.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977. doi: 10.1086/jar.33.4.3629752.

Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. Tensor ring decomposition. June 2016. arXiv:1606.05535.