Bayesian Tensor Regression Of Random Fields

Max Kirstein
Corporate Research Robert-Bosch GmbH
31102 Hildesheim, Germany
max.kirstein@de.bosch.com

Martin Eigel
Weierstrass Institute Berlin
10117 Berlin, Germany
eigel@wias-berlin.de

Dimitrios Bariamis
Corporate Research Robert-Bosch GmbH
31102 Hildesheim, Germany
dimitrios.bariamis@de.bosch.com

Abstract

In this work in progress we take a look at a tensor model in the Bayesian setting, which is applied to the approximation of a stochastic function. Specifically, we learn a random field by stochastically optimising a coefficient tensor in the tensor-train format by variational inference. That is, we minimise a Kullback-Leibler divergence based loss functional. The task is motivated by recent work in the machine learning community and the popular field of uncertainty quantification, where this problem arises from partial differential equations with uncertain data. The model is implemented in a deep probabilistic programming framework. It is thus possible to use it as a drop in replacement in other deep probabilistic architectures such as Bayesian neural networks.

1 Introduction

Our central interest in this work in progress is function approximation of random fields by means of tensor models which have made an impact in different fields [22, 1, 2]. Additionally, we would like to point out further research directions for tensor models as a technique for high-dimensional function approximation. In this sense we illustrate a problem, which is well known and extensively studied in the uncertainty quantification community of numerical analysis and might be of considerable interest for machine learning applications.

Random fields, the generalisation of stochastic processes to arbitrary domains, are ubiquitous in the natural and engineering sciences. There they are often featured as randomised input quantities for partial differential equations, e.g. when there is uncertainty about the properties of a certain medium in a diffusion problem.

Learning a random field translates to a high dimensional regression problem, where two input domains need to be specified. Usually the physical domain is discretised as an equidistant spatial grid and the parameter domain is randomly sampled according to an assumed probability distribution.

In our case we use a linearised tensor network model in conjunction with a polynomial chaos basis to learn the stochastic function. This approach enables us to capitalise on the non-linear properties of the basis as well as on the multi-linear properties of a tensor product space.

In contrast to the classical approach of function approximation and learning theory, where data is assumed to follow an unknown distribution but model parameters and all related quantities are treated as point estimates, we consider a Bayesian tensor model. That is, all involved quantities are random variables, which permits us to reason about uncertainty involved in function approximation. This approach, together with a tensor model, is closely related to the notion of Bayesian neural networks [18]. Furthermore, low-rank tensor formats have shown to be highly capable of approximating stochastic fields, see e.g. [3].

2 Related work

In the mathematics and engineering communities concerned with uncertainty quantification, tensor representations of random fields have gained interest in the last ten years. Most research in this area employs the inherent properties of tensors to directly represent the stochastic function in question. However, this is done deterministically. Thus, no direct uncertainty information can be gained. Some works in this area are [3], where the computation of hierarchical decomposition of tensors with the resulting properties and their applications in differential equations are discussed. Another work is [8], where the authors reduce the complexity of a stochastic finite element method by means of tensor decompositions. Furthermore, in [9] a tensor model is used to learn a direct functional representation of a probability density.

In the machine learning community, a lot of tensor methods are applied to unsupervised learning as means to reduce dimensionality e.g. in [24] where tensors are used to represent different data aspects. There has also been work linking tensor decompositions to convolutional neural network models [5]. More recently, there has also been an interest in utilising tensor decompositions for high dimensional supervised learning problems. In [22] the authors also construct a tensor product space and employ low-rank decompositions in order to solve a classification problem of handwritten digits. Training of the the resulting model is performed by a variant of the density matrix renormalization group (DMRG) algorithm [25]. Authors of [12] show the connection between certain types of probabilistic graphical models. Additionally they show that combining different types of tensor decompositions can improve model expressibility. Contributions, which are similar to our work can be found in [7], where the same ansatz as in [22] is used. However, instead of DMRG stochastic optimisation is used for learning. Additionally, a new framework is presented, which enables tensor decompositions to become drop in replacements for standard neural network layers. Furthermore, in [16] a Bayesian perspective is also taken. However, the tensor decomposition is extended to include specific data for the problem at hand.

3 Problem formulation

3.1 Function spaces, Karhunen-Loeve expansion and tensor model

In the following we take the same approach and rely on the same formalism as [10]. However, we make a distinction later on as we learn the tensor by Bayesian means. A more thorough derivation can be found e.g. in [6].

We start from a spatial domain with $x \in D \subset \mathbb{R}^d$, $d = 1, 2, 3$ and a stochastic domain $y \in \Gamma \subset \mathbb{R}^S$, $S \in \mathbb{N}$. The goal is to reconstruct a random function / random field f in a Bochner space, i.e. $f \in L^2(\Gamma, \rho; \mathcal{X}) = \mathcal{X} \otimes \mathcal{Y} := \mathcal{V}$, with $\mathcal{X} \subseteq L^2(D)$, $\mathcal{Y} \subseteq L^2(\Gamma, \rho)$ with probability distribution ρ on Γ.

A straight forward way to represent a random field is to use the Karhunen-Loève expansion. In order to formulate an affine field we further assume $\Gamma = [-1, 1]^S$ and $\rho = \mathcal{U}(\Gamma)$ to be the uniform distribution. This leads to

$$f(x, y) = a_0(x) + \sum_{s=1}^{S} a_s(x) y_s, \tag{1}$$

for $x \in D$ and $y \in \Gamma$, where the set $\{a_0, \ldots, a_S\}$ is an orthonormal basis of the space $L^2(D)$. Notice, that we have truncated the (normally infinite) expansion to S terms.

Up until now we have given continuous formulations for the function spaces we are working in. However, in order to move to computation we have to define a finite dimensional subspace for $\mathcal{V}$. Thus, let $\{\phi_n\}$ for $n = 1, \ldots, N$ be basis functions on a regular grid of D with N distinct nodes. Now, we can define $\mathcal{X}_N = \operatorname{span}\{\phi_1, \ldots, \phi_N\} \subseteq \mathcal{X}$ to be a finite dimensional subspace of $\mathcal{X}$. In order to formulate a subspace for $\mathcal{Y}$ we assume a product structure, i.e. $\mathcal{Y} = L^2(\Gamma, \rho) = \bigotimes_{s=1}^{S} L^2([-1, 1], \tilde{\rho})$.

This permits us to use a tensor product basis of polynomials up to degrees $\{q_1, \dots, q_S\}$, $q_s \in \mathbb{N}$ given by

$$\Psi_\alpha = \bigotimes_{s=1}^{S} \psi_{\alpha_s}, \tag{2}$$

with multi-indices $\alpha \in \mathcal{I} \subseteq \times_{s=1}^{S}[q_s]$. Therefore, we can define $\mathcal{Y}_\mathcal{I} = \operatorname{span}\{\psi_{\alpha_1} \otimes \dots \otimes \psi_{\alpha_S}\} \subseteq \mathcal{Y}$. Finally, this results in the finite dimensional space $\mathcal{V}_{N,\mathcal{I}} = \mathcal{X}_N \otimes \mathcal{Y}_\mathcal{I} \subseteq \mathcal{V}$.

Therefore, we can represent every function in $\mathcal{V}_{N,\mathcal{I}}$ by a linear combination of its basis functions

$$\hat{f}(x^{(j)}, y) = \sum_{n=1}^{N} \sum_{\alpha \in \mathcal{I}} W(n, \alpha) \phi_n(x^{(j)}) \Psi_\alpha(y), \tag{3}$$

for all nodes $x^{(j)} \in D$, $j = 1, \dots, N$ on the spatial grid and a random sample $y \in \Gamma$ from the stochastic domain.

A closer look at the coefficient of the linear combination reveals a tensor $W \in \mathcal{W} := \mathbb{R}^{N, q_1, \dots, q_S}$. When faced with a high dimensional stochastic grid, i.e. with large S and high degree polynomials, W will become too large for computation due to the curse of dimensionality. Therefore we employ a low-rank tensor decomposition, in order to reduce complexity. Namely, we use the tensor-train decomposition [20]. Thus, we can write

$$W(n, \alpha) = A_0(n) A_1(\alpha_1) \cdots A_S(\alpha_S), \tag{4}$$

with the factor matrices $A_0(n) \in \mathbb{R}^{r_0}$ for $n = 1, \dots, N$, $A_i(\alpha_i) \in \mathbb{R}^{r_{i-1}, r_i}$, for $\alpha_i = 1, \dots, q_i$ and $A_S(\alpha_S) \in \mathbb{R}^{r_{s-1}}$, for $\alpha_S = 1, \dots, q_S$. The ranks are defined by $r_0, \dots, r_{S-1} \in \mathbb{N}$. Using the ranks as a criterion, we can define our final model class as the set of functions $\mathcal{M} := \{\hat{f}_W \in \mathcal{V}_{N,\mathcal{I}} : W \in \mathcal{M}_{\leq \mathbf{r}}\}$ with the set of tensors which permit a tensor-train decomposition of rank(vector) of at most $\mathbf{r} = [r_0, \dots, r_{S-1}]$.

3.2 Bayesian framework

For a brief introduction into the Bayesian framework we follow [23, 19]. We assume W to be a $\mathcal{W}$-valued random variable with Borel σ-algebra $\mathcal{B}(\mathcal{W})$ and prior probability distribution $\mathbb{P}_0$. Furthermore, the random field is a $\mathcal{G} := \mathbb{R}$-valued random variable with Borel σ-algebra $\mathcal{B}(\mathcal{G})$ and realisations $g := f(x, y)$. We choose a Markov kernel to construct the likelihood function and probability distribution of the random field. That is, $\mathbb{P}_g \colon \mathcal{B}(\mathcal{G}) \times \mathcal{W} \to [0, 1]$. From this, we can formulate the joint distribution $\mathbb{P}_{W,g}(B) = \int_\mathcal{W} \int_\mathcal{G} \mathbb{1}_{\mathbb{E}[W,g]} \, \mathbb{P}_g(\mathrm{d}g, w) \mathbb{P}_0(\mathrm{d}w)$ for $B \in \mathcal{B}(\mathcal{W}) \times \mathcal{B}(\mathcal{G})$. With the joint distribution in place, we can give the posterior distribution by the Radon-Nikodym derivative

$$\frac{\mathrm{d}\mathbb{P}_W}{\mathrm{d}\mathbb{P}_0} = Z^{-1} \exp -\mathbb{P}_g(g, w), \tag{5}$$

with $Z = \mathbb{E}_{\mathbb{P}_0}[\exp(-\mathbb{P}_g(y, w))]$. Further, to actually make predictions in conjunction with uncertainty information we construct the posterior predictive distribution, i.e. in order to generate a prediction g^* we sample $\mathbb{P}_{g^*}(C) = \int_\mathcal{W} \mathbb{P}_g(g^* \in C, w) \, \mathbb{P}_W(\mathrm{d}w)$, with $C \in \mathcal{B}(\mathcal{G})$.

3.3 Tensor-train prior

First, we assume each of the elements of the core tensors in Equation 4 to be i.i.d. random variables. Thus, we have

$$\begin{aligned}
A_0 &\sim \mathbb{P}_{0,A_0} := \prod_{k_0=1}^{r_0} \prod_{n=1}^{N} \mathbb{P}_{0,k_{i-1},n} \\
A_i &\sim \mathbb{P}_{0,A_i} := \prod_{k_{i-1}=1}^{r_{i-1}} \prod_{\alpha_i=1}^{q_i} \prod_{k_i=1}^{r_i} \mathbb{P}_{0,k_{i-1},\alpha_i,k_i} \\
A_S &\sim \mathbb{P}_{0,A_S} := \prod_{k_{S-1}=1}^{r_{S-1}} \prod_{\alpha_S=1}^{q_S} \mathbb{P}_{0,k_{S-1},\alpha_S}.
\end{aligned} \tag{6}$$

A sensible choice of prior is of course the Gaussian distribution, because it will automatically regularise our model towards smooth solutions [23]. Another choice, can be found in [13], where a Gamma hyper-prior is used to capture some of the couplings between tensor-train cores. Unfortunately, this still neglects the overall structure and properties of the tensor by assuming univariate distributions for every tensor entry. However, a fitting prior distribution for the tensor-train format is subject to active research. Thus, the prior will be properly adjusted in the near future.

3.4 Summarising remark

In summary our model enables us to perform regression in a high dimensional product function space by employing distinct basis functions. Furthermore, the tensor-train decomposition circumvents the curse of dimensionality resulting from the tensor-product space ansatz in the stochastic space. We extend the capabilities of this model to include uncertainty quantification by moving from the classical function approximation framework to a Bayesian framework, where point estimates for all quantities are changed in favor of probability distributions.

Therefore, our contribution lies in random field reconstruction by means of a Bayesian tensor model, which is motivated by problems from uncertainty quantification in the numerical analysis community. Additionally, we would like to bring both the Bayesian and tensor model research areas closer together e.g. by raising awareness for the need of proper prior distributions for tensor models, in order to fully capture the intricate dependencies.

4 Model training

In contrast to other work involving the tensor-train format, we do not employ tensor decomposition specific learning algorithms like the DMRG [25] or the alternating linear scheme (ALS) [15], but rather use stochastic optimisation to train our model. However, because we are working in the Bayesian regime, optimising the tensor-train elements directly is not possible. Instead, we operate on parameterised probability distributions.

4.1 Variational inference

Due to the complexity of the normalisation constant Z in Equation 5, we are not able to compute the posterior analytically. Therefore, we need to resort to an approximation for the posterior, which we can gain by optimising a Kullback-Leiber divergence based loss functional. However, the chosen approximation is usually less complex and therefore less expressive. Maximising the functional called evidence lower bound (ELBO) is equivalent to minimising the Kullback-Leibler divergence between the posterior and an approximate distribution, which can be done without the computation of any normalisation constants.

The components for variational inference to work include the posterior distribution $\mathbb{P}_W$ of tensor W in the tensor-train format, the prior $\mathbb{P}_0$, a so-called variational distribution $\mathbb{Q}_W$ and the likelihood $\mathbb{P}_g$. In this setting we are trying to optimise over a set of probability distributions (which are part of a Radon measure space), where we want to enforce a separability condition. This is included to reduce the complexity of the variational distribution. Further information on the construction of and optimisation on the mentioned set can be found in [11].

The loss functional ELBO is given by: $\text{ELBO}(\mathbb{Q}_W) := D_{\text{KL}}(\mathbb{Q}_W||\mathbb{P}_0) - \mathbb{E}_{\mathbb{Q}_W}[\log \mathbb{P}_g]$. Because, we specified the negative ELBO, we minimise over the objective

$$L(\mathbb{Q}_W) = -\int_\Gamma \text{ELBO}(\mathbb{Q}_W)\ \rho(\mathrm{d}y), \tag{7}$$

which is the standard formulation in statistical learning and can therefore be treated in much the same way. That is, by employing stochastic optimisation. Additionally, in practice we resort to optimising over parameterised densities, which results in an analogue objective. However, the actual distribution $\mathbb{Q}_W$ is exchanged with parameters governing a probability density function. This enables us to exploit automatic differentiation variational inference [17].

4.2 Variational distribution

A parametric family for the variational distribution often used in the machine learning community is a fully factorised Gaussian over all elements of W in tensor-train format. This is analogous to the i.i.d. assumption used for the prior measure, however a Gaussian is specifically employed. Nonetheless, using a Gaussian effectively doubles the amount of parameters in the model to be estimated. Fortunately, due to dimensionality reduction brought on by the tensor-train format we are still able to work with the model.

5 Numerical simulations

5.1 Explicit setting

Here we state the details of the numerical simulations carried out, in order to examine the models behaviour. Additionally, we use a Bayesian neural network as baseline comparison. All simulations were carried out on a workstation with an NVIDIA GeForce GTX Titan X using the PyTorch [21] and Pyro [4] frameworks to implement all models. Furthermore, training Bayesian neural networks has been identified as a notoriously difficult task. However, in contrast to Bayesian tensor models, this has already driven a considerable amount of research. Therefore, we limit ourselves to the same basic training strategies we already employ for the tensor based model in order to be able to compare them properly.

We compute on an equidistant spatial interval with $N = 50$ nodes, i.e. we choose $x \in D \subset \mathbb{R}$. Furthermore, we let $S \in \{2, 5, 10\}$. As basis a_s for $s = 1, \ldots, S$ we use a slowly decaying Fourier type

$$a_s(x) = \gamma s^{-2} \cos\left(2\pi\beta(s)x\right), \tag{8}$$

with $\beta(s) = s - k(s)(k(S) + 1)/2$ and $k(s) = \lfloor -1/2 + \sqrt{1/4 + 2s} \rfloor$. Furthermore, we set $a_0(x) = 0$, enforcing a zero mean random field. Because the target function is analytically known, we are free to simulate as many realisations as needed. Therefore, in order to observe increasingly accurate approximations, we generate $M \in \{1000, 2000, 3000, \ldots, 10000\}$ realisations and partition into training (80%), validation (10%) and test (10%) sets.

For the polynomial basis Ψ_α of the discrete stochastic space $\mathcal{Y}_\mathcal{I}$ we take individual ψ_{α_s} to be Chebychev polynomials of a degree up to 3. The tensor-train ranks $[r_0, \ldots, r_{S-1}]$ are allowed to change with the dimensionalty of the stochastic space, i.e. for higher dimensional spaces we take corresponding ranks. This allows our model to grow in complexity according to the problem. That is, for $S \in \{2, 5, 10\}$ we have 636, 2550 and 9900 total parameters according to ranks $\mathbf{r} \in \{[3, 3], [6, \ldots, 6], [11, \ldots, 11]\}$. Both, the polynomial degree and rank hyper-parameters are determined by grid search on the validation set. The neural models use two affine linear hidden layers with rectified linear units and matching number of parameters. The number of hidden layers is determined by hyper-parameter optimisation on the validation set. However, the depth of the neural model is tightly constrained by the number of parameters. Additionally, models with more than two hidden layers require projection based skip connections [14] which introduces further parameters. Also, we were not able to train models with more than 5 hidden layers (and therefore significantly more parameters) under the standard training employed here.

Specifics for the Bayesian framework are all taken to be Gaussian. That is, we have a Gaussian likelihood $\mathbb{P}_g$ with hyper-parameter $\sigma_g^2 = 1 \times 10^{-2}$. This is accompanied by prior distributions $\mathbb{P}_{0,k_0,n} = \mathbb{P}_{0,k_{i-1},\alpha_i,k_i} = \mathbb{P}_{0,k_{S-1},\alpha_S} = \mathcal{N}(\mu, \sigma^2)$, with $\mu = 0$ and $\sigma^2 = 1 \times 10^{-1}$. Furthermore, we draw 50 samples from the posterior predictive to estimate its mean and variance on the validation and test set. Whereas, we only use 10 during training. The Bayesian neural network models employ the same likelihood function, prior distribution and hyper-parameters. Grid searching different combinations did not result in better performance on the validation set.

There are a number of first order optimisation algorithms to choose from. In preliminary experiments, the YOGI algorithm [26] converged faster with less tweaking of hyperparamters compared to stochastic gradient descent with momentum and the ADAM algorithm. Therefore, we chose YOGI with an initial step size of 1×10^{-2} for all shown experiments.

In order to determine the approximation quality we use the relative root mean squared error (RMSE)

$$\mathcal{E}(f, \hat{f}) := \left(\mathbb{E}_\rho\left[\left\|f - \mathbb{E}_{\mathbb{P}_{g^*}}\left[\hat{f}\right]\right\|^2_{L^2(D)} / \|f\|^2_{L^2(D)}\right]\right)^{1/2}.$$

5.2 Preliminary results

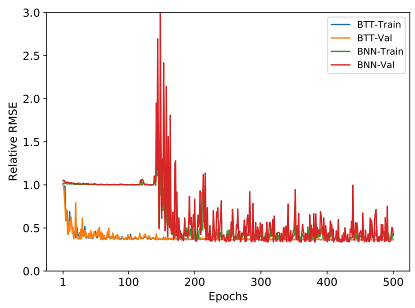
In general, moving to a higher dimensional stochastic space leads to a more complex optimisation task, due to more tensor-train cores. Therefore, we have a larger number of parameters as well as more couplings due to tensor contractions. This can be observed as an exemplary case in Figure 1 by the oscillating behaviour of the relative RMSE during training in higher dimensions. More data of course facilitates optimisation, thus for a larger number of data points training converges faster and is less erratic. Furthermore, we see additional details of the training procedure in two of the sub-figures. In Figure 1c we notice a plateau which drops during the course of one epoch. The drop is most likely due to a change in learning rate by the adaptive YOGI algorithm. In addition, Figure 1e shows epochs with constant relative RMSE. Such uniform behaviour indicate ridges or valleys in the optimisation landscape which the optimisation algorithm is not (yet) able to escape. That is, all proposed parameter values lead to equal contribution in the error computation. Also in the exemplary case of Figure 1, we see the learning dynamics for the Bayesian neural network models under the same data requirements. In general, we observe more variations in the dynamics across all combinations of S and M. However, they are especially pronounced in the small data regime, i.e. $M = 1000$. Furthermore, Figure 1a, Figure 1b and Figure 1c show the same ridge or valley like behaviour we see in the Bayesian tensor-train model in Figure 1e. Interestingly, in Figure 1b a drop in relative RMSE happens for both models even though for the Bayesian neural network the drop is delayed. Additionally, these drops are absent in the higher dimensional problems. This might indicate that optimisation of the Bayesian tensor-train model is easier in comparison to the neural model. In conclusion, we can say that the Bayesian tensor based model not only converges faster, but also to areas of the loss landscape which provide lower relative RMSE when compared to a Bayesian neural network model under a standard training regime.

Moreover, convergence of the relative RMSE is given in Figure 2. We presume that the absence of a prior as well as a variational distribution, which is tailored to the specific needs of a tensor-train model has a larger impact than expected. For Figure 2a, we can see that it takes 3000 data points for the error to actually go down to an acceptable level. From this point on increasing the amount of data does not change much. Moving from $S = 2$ to $S = 5$ dimensions in the stochastic domain imposes difficulties on learning. This is eminent in Figure 2c, where the convergence oscillates after an initial drop at 2000 data points. In Figure 2e, we observe a more or less steady convergence. However, more data points are needed and the error does not reduce as much as for the lower dimensional cases. Additionally, most Bayesian models cannot provide the same amount of accuracy as their deterministic counter parts due to the inherent stochasticity. Also this is the reason for the slight inconsistency noticeable in the convergence plots of Figure 2. Nevertheless, they enable us to quantify uncertainty inherent in data and model alike.

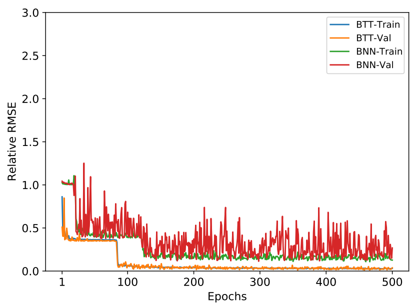
Convergence of the Bayesian neural network model is given in the right column of Figure 2. First, the relative RMSE steadily remains larger over the course of increased data points than the relative RMSE of the Bayesian tensor model. Second, for all values of S convergence oscillates around a fixed value. Third, the oscillations are more pronounced in comparison. We attribute this behaviour in part to the difficulty of the optimisation task, but also in part to the constrained model size. In contrast the tensor based model operates in a high dimensional space spanned by generic Chebychev polynomials, whereas the neural model additionally needs to learn a proper basis for both spatial and stochastic space. However, due to the parameter constraint posed by the need for comparison, the Bayesian neural network is not able to express the stochastic function underlying the data.

6 Discussion

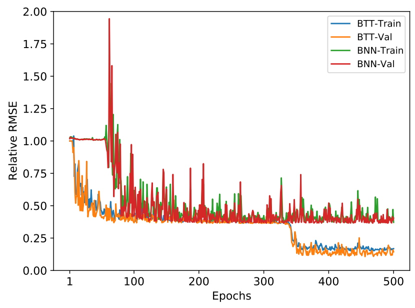
We presented a Bayesian tensor model in the tensor-train format for approximation of a stochastic function. The model was learned by variational inference, i.e. stochastic optimisation over a set of probability distributions. As prior measures for the models tensor-variate coefficient we chose univariate Gaussians. Furthermore, we also used a family of univariate Gaussians as our feasible set. The preliminary results already indicate the potential of Bayesian tensor models for function



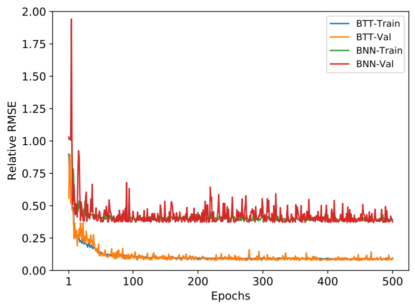
(a) $S = 2$, $M = 1000$



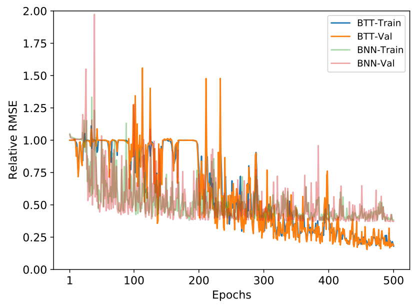
(b) $S = 2$, $M = 10000$



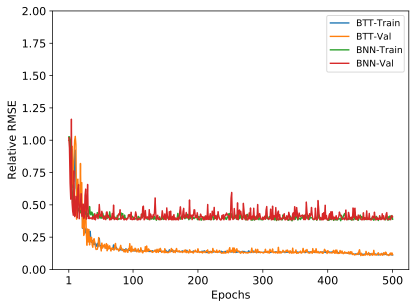
(c) $S = 5$, $M = 1000$



(d) $S = 5$, $M = 10000$



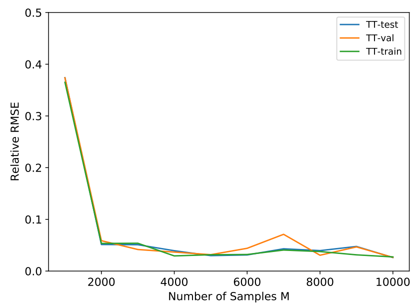
(e) $S = 10$, $M = 1000$



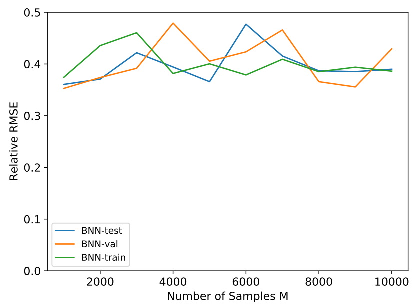
(f) $S = 10$, $M = 10000$

Figure 1: Relative RMSE of Bayesian tensor-train (BTT) and Bayesian neural network (BNN) model for increasing stochastic dimension S and amount of samples / data points M over the course of training.

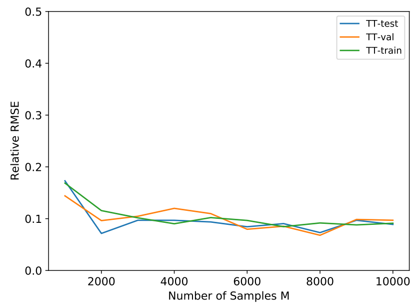
approximation. Observing the presented experiments further reveals that the prior, in addition to the family of distributions as search space, has to be chosen judiciously in order to effectively regularise training. This can be further improved upon by explicitly accounting for the tensor / tensor-train structure and its intricate interdependencies. Therefore, this leads us to further pursue this research direction in the future. Moreover, the reconstruction is a first step towards solving more involved partial differential equations by the same Bayesian tensor ansatz. Additionally, Bayesian tensor network models are closely connected to Bayesian neural networks, which have proved to be important for high stakes machine learning applications and have received considerable attention from researchers over the last years. In order to further underline this connection we compared both models under the same amount of parameters and training regime. We conclude that in a low to medium number of parameters setting the Bayesian tensor-train based model is superior to a Bayesian neural network for our task. Finally, we hope that this work will encourage other researchers to make a contribution to this (in our opinion very interesting) line of work.



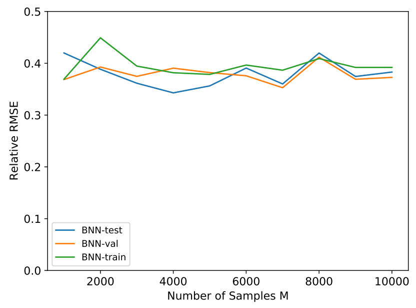
(a) Tensor model, $S = 2$



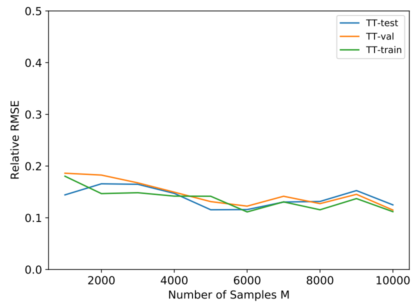
(b) Bayesian NN, $S = 2$



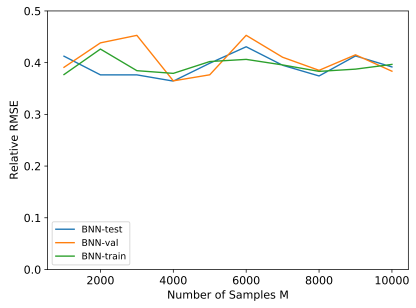
(c) Tensor model, $S = 5$



(d) Bayesian NN, $S = 5$



(e) Tensor model, $S = 10$



(f) Bayesian NN, $S = 10$

Figure 2: Relative RMSE progression over the amount of samples / data points M for different stochastic dimensions. Left column shows tensor model, right column Bayesian neural network model.

Broader Impact

In general the application of the Bayesian framework positively influences our model in a way that it provides us with information a deterministic model is not able to. That is, quantifying uncertainty in the models parameters lets us directly reason about uncertainty in model predictions. This is especially favorable for all downstream decision process which require careful validation. Examples of these tasks arise e.g. in medical and financial applications, automated driving or production of critical machine systems. Furthermore, tensor models and its decompositions could work as drop in replacements in deep architecture models when interpretability is required, due to its white-box nature. However, machine learning is a discipline with inherent duality. Meaning that most methods can be employed to have beneficial effects as well as deeply negative effects on our lives. The work we have shown here is certainly no exception in this case.

Acknowledgments and Disclosure of Funding

Finally, we would like to thank Manuell Marschall for providing valuable input and fruitful discussions for this work.

References

[1] M. Ali and A. Nouy. Approximation with Tensor Networks. Part I: Approximation Spaces. *arXiv e-print*, page arXiv:2007.00118, 2020.

[2] M. Ali and A. Nouy. Approximation with Tensor Networks. Part II: Approximation Rates for Smoothness Classes. *arXiv e-print*, page arXiv:2007.00128, 2020.

[3] M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor Networks and Hierarchical Tensors for the Solution of High-Dimensional Partial Differential Equations. *Foundations of Computational Mathematics*, 16:1423–1472, 2016.

[4] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 20:1–6, 2019.

[5] N. Cohen and A. Shashua. Convolutional Rectifier Networks as Generalized Tensor Decompositions. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of Machine Learning Research*, volume 48, pages 955–963, 2016.

[6] S. Dolgov, B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Polynomial Chaos Expansion of Random Coefficients and the Solution of Stochastic Partial Differential Equations in the Tensor Train Format. *SIAM/ASA Journal on Uncertainty Quantification*, 3:1109–1135, 2015. doi: https://doi.org/10.1137/140972536.

[7] S. Efthymiou, J. Hidary, and S. Leichenauer. TensorNetwork for Machine Learning. *arXiv e-prints*, page arXiv:1906.06329, 2019.

[8] M. Eigel, M. Pfeffer, and R. Schneider. Adaptive Stochastic Galerkin FEM with Hierarchical Tensor Representations. *Numerische Mathematik*, 136:765–803, 2017. doi: https://doi.org/10.1007/s00211-016-0850-x.

[9] M. Eigel, M. Marschall, and R. Schneider. Sampling-free Bayesian Inversion with Adaptive Hierarchical Tensor Representations. *Inverse Problems*, 34(3):035010, 2018. doi: 10.1088/1361-6420/aaa998.

[10] M. Eigel, R. Schneider, P. Trunschke, and S. Wolf. Variational Monte Carlo - Bridging Concepts of Machine Learning and High-Dimensional Partial Differential Equations. *Advances in Computational Mathematics*, 45:2503–2532, 2019. doi: https://doi.org/10.1007/s10444-019-09723-8.

[11] A. Fraysse and T. Rodet. A Measure-Theoretic Variational Bayesian Algorithm for Large Dimensional Problems. *SIAM Journal on Imaging Sciences*, 7(4):2591–2622, 2014.

[12] I. Glasser, N. Pancotti, and J. I. Cirac. From probabilistic graphical models to generalized tensor networks for supervised learning. *arXiv e-prints*, page arXiv:1806.05964, 2018.

[13] C. Hawkins and Z. Zhang. Bayesian Tensorized Neural Networks with Automatic Rank Selection, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: https://10.1109/CVPR.2016.90.

[15] S. Holtz, T. Rohwedder, and R. Schneider. The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 34(2):683—-713, 2012.

[16] R. Hu, G. K. Nicholls, and D. Sejdinovic. Large Scale Tensor Regression using Kernels and Variational Inference. *arXiv e-prints*, page arXiv:2002.04704, 2020.

[17] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic Differentiation Variational Inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

[18] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, 1996.

[19] J. A. Nelder and R. W. M. Wedderburn. Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972.

[20] I. V. Oseledets. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33: 2295–2317, 2011. doi: https://doi.org/10.1137/090752286.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[22] E. M. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4799–4807. Curran Associates, Inc., 2016.

[23] A. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010. doi: https://doi.org/10.1017/S0962492910000061.

[24] M. A. O. Vasilescu and D. Terzopoulos. Multilinear Analysis of Image Ensembles: TensorFaces. In *ECCV*, 2002.

[25] S. R. White. Density Matrix Formulation for Quantum Renormalization Groups. *Physical Review Letters*, 69(19):2863–2866, 1992.

[26] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive Methods for Nonconvex Optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9793–9803. Curran Associates, Inc., 2018.