TENSOR FACTORIZED DENSITY ESTIMATION

Eskild Børsting Sørensen (s163918), Johannes Emme Jørgensen (s144003)
Jonas Vestergaard Jensen (s162615), Peter Mørch Groth (s164049)

Technical University of Denmark
DTU Compute

ABSTRACT

When performing density estimation, a major issue often arises due to the curse of dimensionality. For this reason, an exploration of how a joint distribution can be factorized using tensor decomposition procedures is explored. The results reveal that for the regularized parametric models based on tensor decompositions such as Tensor Train and Tensor Ring do not constitute new state-of-the-art density estimators for high-dimensional data, but shows desirable properties such as generalizability.

Index Terms— Density estimation, Gaussian Mixture Model, Tensor factorization

1. INTRODUCTION

Density estimation is a key challenge within unsupervised machine learning where its usage spans multiple tasks such as outlier detection, data imputation, and classification through use of Bayes' theorem to mention a few. A widely used model for density estimation is the Gaussian Micture Model (GMM).

$$p(x_1, x_2, \dots, x_M) = p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (1)$$

GMMs allow complex probability distributions to be approximated by use of simple linear superposition of Gaussian components, see (1). By varying the form of the covariance matrix, the models can vary in complexity. Hence, letting the variables covary will lead to a full covariance matrix, whereas a more restrictive model with independent variables leads to a diagonal covariance matrix. However, a concern clearly arises with GMMs as the full model scales poorly to higher dimensions, while the more restrictive model lacks expressivity. In order to address these problems, attention is drawn to the development within the field of tensor decompositions.

Today, a wide range of studies exist on decomposing tensors (multidimensional generalizations of matrices). Such decompositions serve to efficiently factorize tensors by a reduced number of parameters and thereby hinder the curse of dimensionality. For this reason, tensor decompositions have applications in various fields of unsupervised machine learning [1] and multiple types of decompositions have been proposed over time. Among these are the Canonical Polyadic (CP) decomposition and recent decompositions such as Tensor Train (TT) [2] and Tensor Ring (TR) [3].

An imminent thought is whether one can be inspired by such tensor decomposition procedures for density estimation to achieve highly expressive models that scales better to higher dimensions. For this reason, this paper presents an investigation of the merits and limitations using these new tensor factorized density estimators. In order to assess the performance, the developed methods are contrasted to the neural density estimator FJJORD [4] and conventional density estimation based on the GMM.

2. BACKGROUND

2.1. Canonical Polyadic Decomposition

The aim of the CP-decomposition is to decompose an *M*-dimensional tensor into a finite sum of *rank-one* tensors [5]:

$$A(i_1, i_2, ..., i_M) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \cdots U_M(i_M, \alpha),$$

where *A* is an *M*-dimensional tensor, $i_1, i_2, ..., i_M$ are the indices of a particular element of *A*, and where U_j is a collection of r vectors corresponding to the jth dimension of *A*, indexed by α. The CP-decomposition is transformed into a density by letting each collection of *U*-vectors be collections of univariate distributions. These in turn, are parameterized by univariate Gaussian distributions:

$$\begin{aligned} p(x_1, x_2, ..., x_M) &= \sum_{k=1}^{K} p(k) \prod_{m=1}^{M} p(x_m|k) \\ &= \sum_{k=1}^{K} w_k \prod_{m=1}^{M} \mathcal{N}(x_m|\mu_{k,m}, \sigma_{k,m}^2), \end{aligned}$$

where a normalizing factor has been placed over the product of univariate Gaussians to ensure that it is a proper probability

distribution, thus resulting in a traditional GMM with diagonal covariances matrices. For the CP-decomposition the *number of free parameters* is given by $(K-1)+2MK$. The CP-factorized model scales with $\mathcal{O}(KM)$. The number of free parameters will serve as a point of reference for comparison between the different models. If a full GMM is used the number of free parameters is $(K-1)+(2M+M(M-1)/2)K$, which scales with $\mathcal{O}(KM^2)$.

2.2. Tensor-Train Decomposition

The TT-decomposition seeks to improve the expressivity when compared to the CP-decomposition, but without suffering from the curse of dimensionality to the same extent as the full GMM [2]. The tensor-train decomposition decomposes an *M*-dimensional tensor as follows:

$$A(i_1,...,i_M) = \sum_{\alpha_0,...,\alpha_{M-1}} G_1(\alpha_0,i_1,\alpha_1)G_2(\alpha_1,i_2,\alpha_2) \cdots G_M(\alpha_{M-1},i_M,\alpha_M),$$

where each *G* corresponds to a three-dimensional array, with the exceptions of the boundary conditions imposed on the sizes of G_1 and G_M to ensure scalar output of the above indexed form. We factorize the density as

$$p(x_1,x_2,...,x_M) = \sum_{k_0,...,k_M}^{K_0,...,K_M} p(k_0)\prod_{m=1}^{M} p(x_m,k_m|k_{m-1})$$
$$= \sum_{k_0,...,k_M}^{K_0,...,K_M} w_{k_0}\prod_{m=1}^{M} w_{k_m,k_{m-1}}\mathcal{N}(x_m|\mu_{k_m,k_{m-1}},\sigma^2_{k_m,k_{m-1}}),$$

where we again have chosen to model each univariate density with a Gaussian distribution. This results in a chain (or train) of weighted Gaussian distributions. These are constructed as nested sums over each dimension. The number of free parameters for the TT-density is $(K-1)+MK(K-1)+2MK^2$. It thereby scales with $\mathcal{O}(K^2M)$.

Previously, the expressivity of the TT decomposition for modelling discrete multivariate probability distributions has been explored in [6].

2.3. Tensor-Ring Decomposition

The TR-decomposition moves from the sequential products of the TT-decomposition to a circular construction, in order avoid the limitations that the TT-decomposition might exhibit [3]. For our TT-density, these limits are the lack of dependencies for the first and final dimensions of the data. We model our density as:

$$p(x_1,x_2,...,x_M)$$
$$= \sum_{k_0,...,k_{M-1}}^{K_0,...,K_{M-1}} p(k_0)\left[\prod_{m=1}^{M-1} p(x_m,k_m|k_{m-1})\right] p(x_M|k_{M-1},k_0)$$
$$= \sum_{k_0,...,k_{M-1}}^{K_0,...,K_{M-1}} w_{k_0}\left[\prod_{m=1}^{M-1} w_{k_m,k_{m-1}}\mathcal{N}(x_m|\mu_{k_m,k_{m-1}},\sigma^2_{k_m,k_{m-1}})\right] \cdot \mathcal{N}(x_M|\mu_{k_{M-1},k_0},\sigma^2_{k_{M-1},k_0}),$$

which essentially connects the first and last carriages of the tensor train via the dependency on k_0. The number of free parameters for the TR density is $(K-1)+(M-1)K(K-1)+2MK^2$. It thereby also scales with $\mathcal{O}(K^2M)$.

3. METHODS

3.1. Inference

For inference, we rely on stochastic variational inference (SVI) using Pyro. In variational inference we generally seek to approximate a posterior distribution with a variational distribution. In practice this is achieved by maximizing the evidence lower bound (ELBO), which is equivalent to minimizing the KL-divergence [7]. Here we however restrict ourselves to maximum likelihood estimation, which we can achieve by specifying an empty variational distribution in Pyro [8]. In this case, the ELBO reduces to the log likelihood, $\log p_\theta(\mathcal{D})$, because our models only contain simple discrete latent variables $(K_0, K_1, \dots)$ sampled from categorical distributions that Pyro can marginalize out during inference.

We optimize the ELBO using the Adam optimizer [9] with a learning rate of $3\mathrm{e}{-4}$ and mini-batching.

3.2. Data

The performance of each of our models are assessed on the four tabular datasets *power*, *gas*, *hepmass* and *miniboone* presented by Grathwohl et. al in [4]. These datasets have respectively 6, 8, 21 and 43 variables and 2 049 280, 1 052 065, 525 123, and 36 488 observations. We used the same data preprocessing and the same training, validation, and test splits as Grathwohl et. al (2019). For visualisation purposes of the model's densities, three synthetic two-dimensional datasets *8gaussians*, *checkerboard*, and *2spirals* are considered with 1750 observations in all data splits.

3.3. Initialisation

The mean vectors for the CP and GMM models are initialised to the cluster centers resulting from a K-means clustering of the training data. Moreover, the variances in the CP models are initialised to the sample variance of the variables. The covariance matrices in the GMM models are initialised to the sample covariance matrices of the K-means clusters.

Initialisation of the TT and TR models is not straightforward as we cannot do a simple K-means clustering. Instead we randomly initialise the parameters of the model to sensible values and determine the log likelihood of the data with

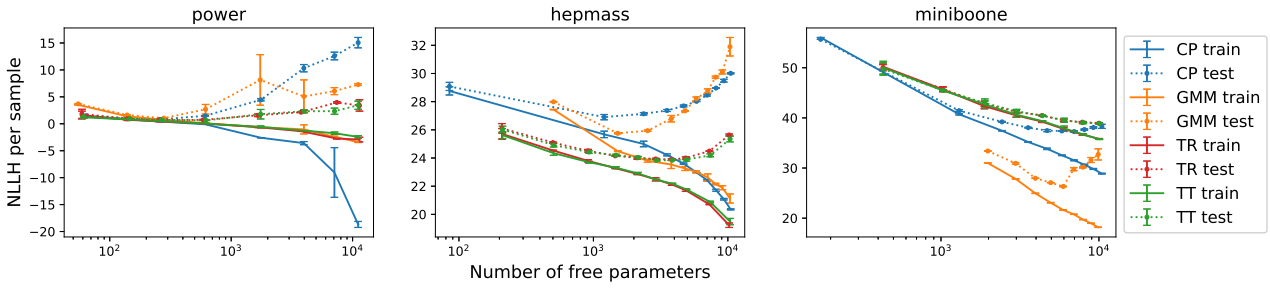

Fig. 1. **Training and test performance.** The models are evaluated at various numbers of free parameters at a sample size of 1750 for three datasets with increasing number of dimensions.

	power	gas	hepmass	miniboone	8gaussians	checkerboard	2spirals
FFJORD	-0.46	-8.59	**14.92**	**10.43**			
CP	-0.42	-10.30	22.76	29.23	1.47	2.02	1.85
GMM	-0.43	**-13.23**	20.78	17.84	1.47	2.04	**1.79**
TT	-0.47	-5.23	23.07	35.29	1.47	**1.93**	1.94
TR	**-0.53**	-5.33	23.06	35.27	1.48	2.01	1.85

Table 1. Test negative log likelihood per sample for the selected models. Results are averages over 3 runs.

this set of parameters. We then repeat this random initialisation 500 times and re-initialise to the set of parameters that resulted in the largest log likelihood.

For all models we let $K_0 = K_1 = K_2 = ...$ for more simple model selection. Furthermore, each model setting is trained and evaluated three times to get more representative results and combat the issue of local minima.

3.4. Ordering of variables

Due to the sequential structure of the TT and TR a natural question arises regarding whether there exists an optimal ordering of variables. By considering partial correlation, and reordering the variables with respect to a heuristic that seeks to maximize the total partial correlation, consistently better results were obtained. Thus, we chose to consider this ordering of variables in all but the experiment in Section 4.1. We use Algorithm 1 that runs in $\mathcal{O}(M^2)$ in order to optimize the structure of the TT and TR.

4. RESULTS

4.1. Comparison on full datasets

As described in Section 3.2 density estimation on the full datasets is performed with the TT and TR models and compared to FFJORD, the standard GMM, and CP models. All of the models (TT, TR, CP, and GMM) have been trained with varying complexity and their optimal complexity determined by their performance on the validation datasets. The validation performance for the different models and varying complexity can be found in Figure A.1 in the appendix.

The test performance of the selected models are shown in Table 1. For the synthetic dataset *8gaussians* all models perform nearly identically, while the TT model does considerably better for the *checkerboard* data. The GMM model is more able to characterise the somewhat complicated *2spirals* dataset, that is especially poorly characterised by the TT model. The densities obtained by the different models on the synthetic data are visualized in Figure A.2.

Both the TT and TR outperform FFJORD, the GMM, and CP models on the *power* dataset, albeit for the TT model only by a small margin. On the remaining tabular datasets the TT and TR models perform similarly without outperforming the other models.

4.2. Overfitting

In order to explore the merits and limitations of the various models, the performance with respect to overfitting is investigated. In an effort to purposefully overfit, subsamples of size 1750 have been collected from the *power*, *hepmass*, and *miniboone* training datasets. The four models are trained until convergence is achieved on the training set or until 10 000 epochs have elapsed. To allow for a fairer comparison, the K parameters for each model have been chosen to correspond to a similar number of free parameters. The mean train and test errors and their standard deviations can be seen in Figure 1. On the six-dimensional *power* dataset, the simpler CP and GMM models overfit to a greater extend than the TT and TR models. All four models appear to reach similar minima at around 120-150 free parameters. For the *hepmass* dataset, the GMM quickly overfits while reaching a minimum that is significantly larger than the minima of the TT- and TR-densities. Additionally, the CP-density fails to reach a comparable minimum and quickly overfits. For the *miniboone* dataset we quite strikingly see the GMM performing significantly better than the remaining models at a fraction of the number of free parameters. The performances of the TT- and TR-densities are pairwise similar for all runs and moreover reach lower standard deviations when comparing to both the CP-density and the GMM.

4.3. Model comparison for smaller sample sizes

To investigate the performance of the different models at varying complexities when less data is available, we subsample the training data of the *power*, *hepmass* and *miniboone* datasets. We choose 1750, 7000 and 28 000 samples for training, effectively yielding nine datasets. We use early stopping in order to obtain the best performing models. The results from the datasets with just 1750 and 7000 samples are shown in Figure 2. The 28 000 subsample results did not differ significantly from the 7000 subsample results and are for this reason omitted.

We observe that the TT and TR models perform very similarly and that their behaviour is consistent when increasing model complexity and subsample size. In contrast, the GMM is very inconsistent and is volatile to varying complexity and subsample size. The TT and TR perform best on the *power*, and *hepmass* dataset while the GMM performs the best on the *miniboone* dataset.

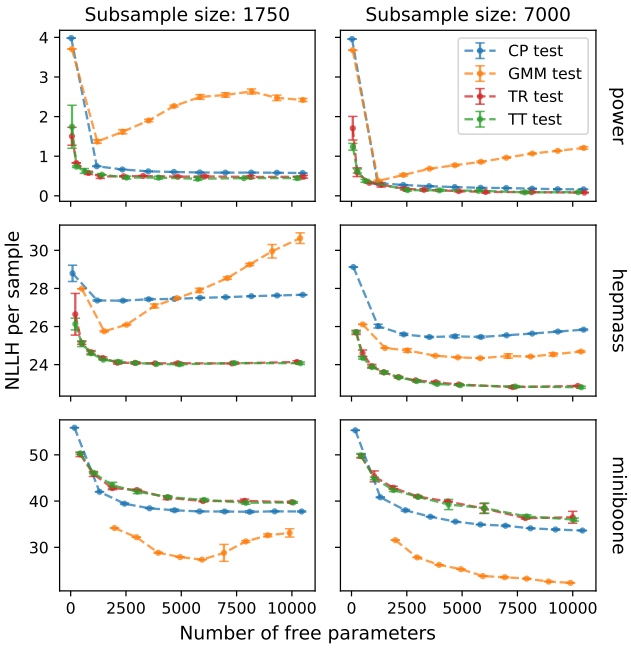

Fig. 2. **Test performance.** Comparison at various levels of model complexities.

4.4. Ordering of variables revisited

Due to the improved results when reordering variables with respect to partial correlation we subsequently investigate whether a more optimal ordering of variables in the TT and TR can be achieved. This is done by maximizing the correlation instead of the partial correlation between neighbouring variables. In Figure 3 we see that maximizing correlation slightly outperforms maximizing partial correlation. Similar results were obtained by smaller experiments on other datasets with various model complexities.

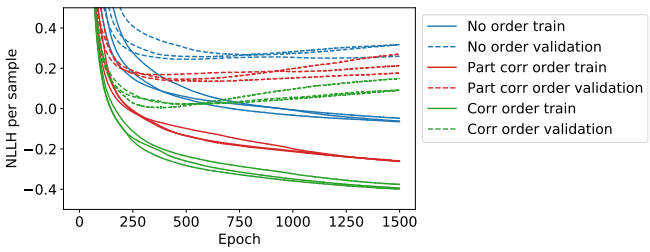

Fig. 3. **Performance of TT and TR by ordering variables.** The experiments were trained on the *power* dataset with 1750 subsamples and k=12.

5. DISCUSSION

From the results in Table 1 it is apparent that the TT and TR models do not form a new state-of-the-art for density estimators, though they do offer improvements on the *power* dataset. It is noteworthy that the performance of the TT and TR models on the full data were obtained without ordering the variables, and could likely be improved by considering an ordering, as Figure 3 indicates.

In general the TT model and TR model perform very similarly, but seem to differ more as the dimensionality of the data decreases.

We have consistently used the number of free parameters as our point of reference for comparison, but we must emphasise that the value of a free parameter differs between the various models. For the CP density two free parameters can correspond to the location and variance of a univariate Gaussian. Yet for the GMM, two free parameters could correspond to a location and a single covariance between two variables. For a high dimensional dataset, this discrepancy increases, as two covariances might offer less expressivity than an entire univariate Gaussian. For example in Figure 1 for the *power* dataset where the CP overfits, the number of parameters is so large that it approaches the possibility of placing univariate Gaussians at all dimensions of all data points.

Considering the results in Figure 1 and 2 we observe that the TT and TR are resilient towards local minima. Additionally they provide a high level of regularization, which minimizes overfitting. Despite increasing the number of free parameters available, the structure of the TT and TR seems to inherit the desirable property of generalizability. Thus, the TT and TR models seem to be especially useful when less data is available as seen in Figure 2.

The code can be found in the following GitHub repository: `https://github.com/jonasvj/TFDE`.

6. REFERENCES

[1] Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos, "Tensor decomposition for signal processing and machine learning," 2016.

[2] Ivan V. Oseledets, "Tensor-train decomposition," SIAM Journal on Scientific Computing, 2011, vol. 33.5, pp. 2295–2317.

[3] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki, "Tensor ring decomposition," 2016.

[4] Will Grathwohl et al., "Ffjord: Free-form continuous dynamics for scalable reversible generative models," 2018.

[5] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann, "Introduction to tensor decompositions and their applications in machine learning," 2017.

[6] Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and J. Ignacio Cirac, "Expressive power of tensor-network factorizations for probabilistic modeling, with applications from hidden markov models to quantum machine learning," 2019.

[7] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006, 8th.

[8] Pyro documentation, "Mle and map estimation," `https://pyro.ai/examples/mle_map.html`, Last visited: 06-05-2021.

[9] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

7. APPENDIX

Algorithm 1 Maximizing partial correlation

Input: data, M
PartMat = `abs`(PartCorrMatrix(data)) - `Identity`(M)
BestChain, BestSumPartCorr = [], 0
for d_o in PartMat **do**
  ThisChain, ThisSumPartCorr = [d_0], 0
  for i in range(M-1)) **do**
    LastVar = ThisChain[-1]
    NextVar = `argmax`(PartMat[LastVar, :])
    ThisChain.append(NextVar)
    ThisSumPartCorre + = PartMat[LastVar, NextVar]
    PartMat[LastVar, :] = 0
  end for
  if ThisSumPartCorr $\geq$ BestSumPartCorr **then**
    BestSumPartCorr = ThisSumPartCorr
    BestChain = ThisChain
  end if
end for
Output: BestChain

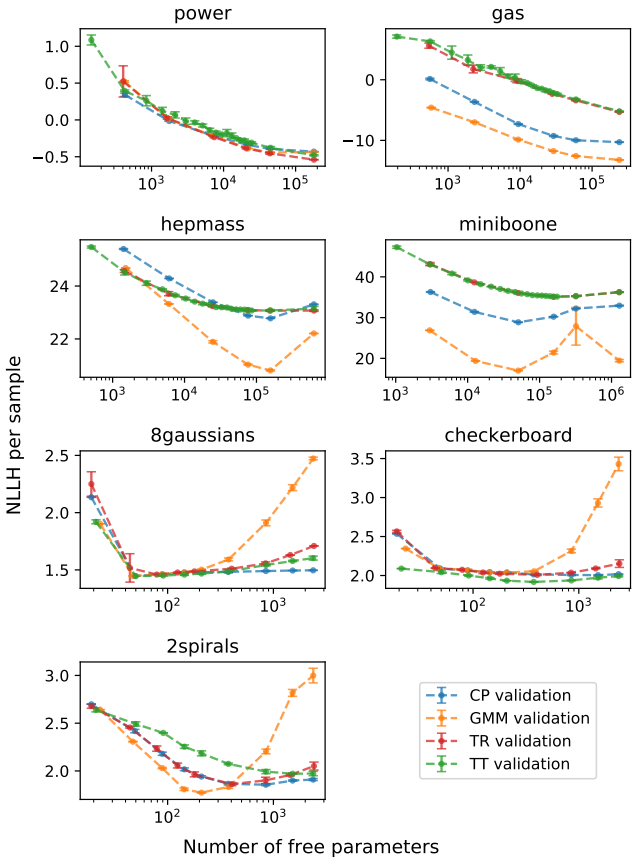

Fig. A.1. **Defining optimal model.** The validation performance for each model for varying model complexity on each of the datasets. The optimal model complexity for each model is determined by the validation performance. The optimal models are used for generating the results of Table 1.

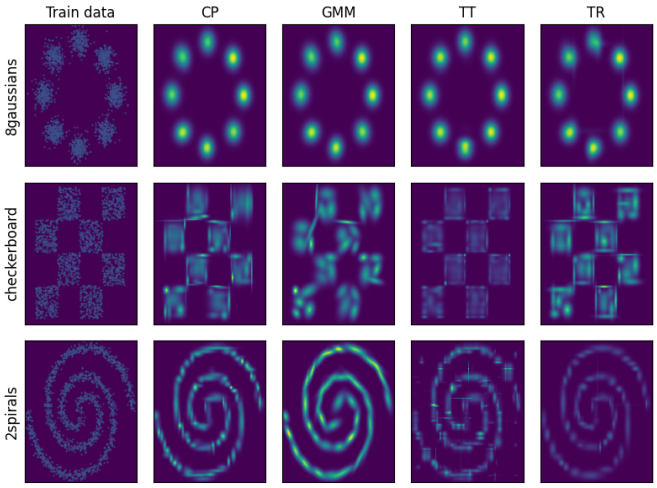

Fig. A.2. **Modelling of synthetic data.** The selected models qualitative performance on the 2D synthetic point data.