Tensor-Train Density Estimation

Georgii S. Novikov[1] **Maxim Panov**[1] **Ivan V. Oseledets**[1]

[1]CDISE, Skolkovo Institute of Science and Technology , Moscow, Russia

A SUPPLEMENTARY MATERIAL

A.1 DETAILS OF THE EXPERIMENTS

In all the experiments set of B-splines of degree 2 with knots uniformly distributed over the distribution support was used as basis functions for rank-1 feature maps. In all the experiments, Riemannian optimization with the optimal learning rate was used if not stated otherwise. In Section 5.3 Adam optimizer from PyTorch was used with default parameters. Sampling from TTDE was performed with 30 binary search iterations. In all the experiments, the batch size was 2^{10} elements per iteration. For all toy and model examples, we used infinite data generators. In real-world data experiments in Section 5.4, the rank r of the TTDE was 64, and the number of basis functions m was 128. Implementation of FFJORD was taken from `https://github.com/rtqichen/ffjord`, implementations of GLOW, Real NVP and MAF were taken from `https://github.com/ikostrikov/pytorch-flows` and used with parameters recommended by authors.

The first three components of the distribution used in Section 5.3 are depicted on Figure 1. Other $d-3$ components are standard Gaussian noise.

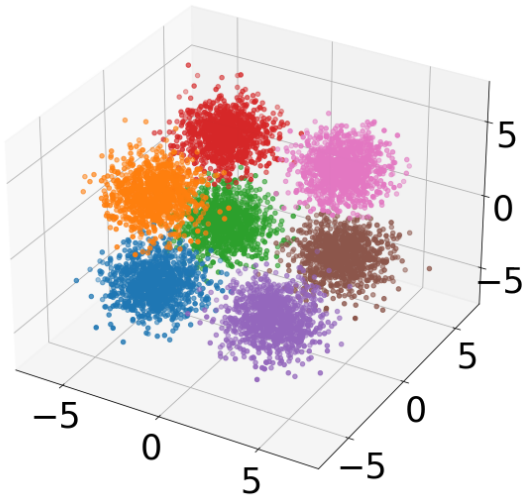

Figure 1: Visualization of the first three dimensions of the model distribution used in this work. It consists of 7 identical Gaussian distributions shown with different colours. Other $d-3$ dimensions are standard Gaussian noise.

A.2 EXISTING MEASURES OF DISCREPANCY

$\mathcal{KL}$-divergence is a popular measure of discrepancy between two distributions, and, during training, is presented in the form of the maximum likelihood problem:

$$\mathcal{KL}(p \parallel q_{\boldsymbol{\theta}}) = -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \log q_{\boldsymbol{\theta}}(\boldsymbol{x}) + const.$$

Different models were built to optimize this kind of discrepancy (including autoregressive models [Ryder et al., 2018], normalizing flows [Kobyzev et al., 2020], energy-based models [LeCun et al., 2006]). The main downside of the maximization of the likelihood is that it explicitly depends on the partition function of the approximation. Thus either the models should be constructed in such a way that the partition function could be efficiently calculated, or expensive Monte-Carlo methods should be used to approximate it during the optimization. We can not use $\mathcal{KL}$-divergence to train TTDE because, although it has a tractable partition function, function in tensor-train format is not guaranteed to be positive.

Fisher discrepancy loss does not depend on the partition function of the approximation:

$$\begin{aligned}
&\mathcal{L}(p, q_{\boldsymbol{\theta}}) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left\| \nabla \log p(\boldsymbol{x}) - \nabla \log q_{\boldsymbol{\theta}}(\boldsymbol{x}) \right\|^2 \\
&= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left\| \nabla \log q_{\boldsymbol{\theta}}(\boldsymbol{x}) \right\|^2 - \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \Delta \log q_{\boldsymbol{\theta}}(\boldsymbol{x}) + const.
\end{aligned}$$

Here $const$ depends only on p and does not depends on $q_{\boldsymbol{\theta}}$.

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

Because of the gradient of the logarithm, the normalization constant cancels out. The downside of this loss is that for complex models like neural networks, the Laplace operator is hard to calculate from both computational and numerical stability points. This loss can be used to train TTDE if we parameterize $\log p(x)$ instead of parameterizing $p(x)$ with the tensor-train model. However, in that case, we would lose the ability to calculate partition function and cumulative density function and thus would not be able to exact sample.

Different versions of adversarial loss were created and successfully used to learn complex distributions like images or speech. They use a separate model as a critic during the training process. Consider the following two variants for vanilla GAN and WGAN, respectively:

$$\begin{aligned}&\mathcal{L}(p, q_\theta) = \\ &= \max_D \left\{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[\log(1 - D(G(\boldsymbol{z})))\right]\right\},\end{aligned}$$

$$\begin{aligned}&\mathcal{L}(p, q_\theta) = \\ &= \max_D \left\{\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[D(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[D(G(\boldsymbol{z}))\right]\right\}.\end{aligned}$$

Generator G maps latent variable $\boldsymbol{z}$ with known distribution $p_{\boldsymbol{z}}$ to the sample space $\boldsymbol{x}$, and discriminator D tries to distinguish between real samples and generated samples. In these cases, $q_{\boldsymbol{\theta}}$ is defined implicitly. Separate choice of the critic architecture, instability of the optimization of the min-max problem, and the intractability of the implicit density function $q_{\boldsymbol{\theta}}$ are the problems that come with the power of adversarial models.

A.3 RIEMANNIAN OPTIMIZATION

Orthogonalization. Left- and right-orthogonalization of the tensor-train decomposition is three sets of matrices: set of left-orthogonal cores $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d$, set of right-orthogonal cores $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_d$ and set of unrestricted cores $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_d$, such that

$$\begin{aligned}&G_1 \times_2^1 \cdots \times_d^1 G_d = \\ &\boldsymbol{U}_1 \times_2^1 \cdots \boldsymbol{U}_{i-1} \times_i^1 \boldsymbol{S}_i \times_{i+1}^1 \boldsymbol{V}_{i+2} \cdots \times_d^1 \boldsymbol{V}_d\end{aligned} \tag{1}$$

for each i, where left-orthogonality means

$$\langle U_k[:, :, i], U_k[:, :, j]\rangle = \delta_j^i,$$

and right-orthogonality means

$$\langle V_k[i, :, :], V_k[j, :, :]\rangle = \delta_j^i.$$

Tangent space. Given the left- and right-orthogonalization of the given tensor-train decomposition of tensor X, tangent space $\mathcal{T}_{\boldsymbol{X}}(\mathcal{M})$ in that point could be constructed as follows:

$$\mathcal{T}_{\boldsymbol{X}}(\mathcal{M}) = \left\{ \begin{array}{c} T = U_1 \times_2^1 \cdots U_{i-1} \times_i^1 S_i^\delta \times_{i+1}^1 V_{i+1} \cdots \times_d^1 V_d, \\ \text{where } 1 \le i \le d,\ S_i^\delta \in \mathbb{R}^{r_{i-1} \times m \times r_i} \end{array} \right\}.$$

Although tensor T is presented as a sum of d tensors of rank r, they share common cores. Because of that, T can be represented with rank $2r$:

$$T = \begin{bmatrix} U_1 & S_1^\delta \end{bmatrix} \begin{bmatrix} V_2 & \\ S_2^\delta & U_2 \end{bmatrix} \cdots \begin{bmatrix} V_{d-1} & \\ S_{d-1}^\delta & U_{d-1} \end{bmatrix} \begin{bmatrix} S_d^\delta \\ V_d \end{bmatrix}$$

Automatic differentiation. If we define operator

$$T_{\boldsymbol{X}}(S_1^\delta, \cdots, S_d^\delta) = \sum_{i=1}^d U_1 \times_2^1 \cdots U_{i-1} \times_i^1 S_i^\delta \times_{i+1}^1 V_{i+1}$$

that maps delta-cores $\left\{S_i^\delta\right\}_{i=1}^d$ into the point in tangent plane, then for any function $g(\boldsymbol{X}) : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}$ projection of the true gradient $\nabla(\boldsymbol{X})$ onto the tangent plane $\mathcal{T}_{\boldsymbol{X}}(\mathcal{M})$ could be efficiently calculated as follows:

$$\mathbb{P}_{\mathcal{T}_{\boldsymbol{X}}(\mathcal{M})} \nabla g(\boldsymbol{X}) = T_{\boldsymbol{X}}(\tilde{S}_1^\delta, \cdots, \tilde{S}_d^\delta),$$

where

$$\tilde{S}_i^\delta = \frac{\partial}{\partial S_i^\delta} g(T_{\boldsymbol{X}}(S_1^\delta, \cdots, S_d^\delta)) \Big|_{S_1^\delta = S_1, S_2^\delta = \boldsymbol{O}_2, \cdots, S_d^\delta = \boldsymbol{O}_d}.$$

Here S_1 is defined in (1), and $\boldsymbol{O}_i$ is core with all elements equal to zero. All $\tilde{S}_i^\delta$ could be calculated using automatic differentiation of function $g \circ T_{\boldsymbol{X}}$.

References

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, May 2020. ISSN 0162-8828. doi: 10.1109/tpami.2020.2992934.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

Tom Ryder, Andrew Golightly, A. Stephen McGough, and Dennis Prangle. Black-box autoregressive density estimation for state-space models. *CoRR*, abs/1811.08337, 2018. URL `http://arxiv.org/abs/1811.08337`.