High-dimensional Density Estimation with Tensorizing flow

Yinuo Ren[*], Hongli Zhao[†], Yuehaw Khoo[†], Lexing Ying[*]

[*]Institute for Computational and Mathematical Engineering, Stanford University
[†] Department of Statistics, University of Chicago

Tensor Network Reading Group
Université de Montréal and Mila
November 21, 2023

Problem setting

Given N i.i.d. d-dimensional samples $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \cdots, x_d^{(i)})_{1 \le i \le N} \sim p^*(\boldsymbol{x})$, construct another distribution $p_\theta(\boldsymbol{x})$ that approximates $p^*(\boldsymbol{x})$.

- $p_\theta(\boldsymbol{x})$ is required to be *normalized*
- $p_\theta(\boldsymbol{x})$ should be easy to sample from

Maximum likelihood estimation (MLE)

- Empirical distribution:

$$p^{\mathsf{E}}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(\boldsymbol{x} - \boldsymbol{x}^{(i)}\right),$$

- MLE formulation:

$$\begin{aligned} \theta = \arg\min_\theta \mathrm{D}_{\mathrm{KL}}\left(p^*(\cdot) \| p_\theta(\cdot)\right) &= \arg\min_\theta \mathbb{E}_{\boldsymbol{x}\sim p^*}\left[-\log p_\theta(\boldsymbol{x})\right] \\ &\approx \arg\min_\theta \mathbb{E}_{\boldsymbol{x}\sim p^{\mathsf{E}}}\left[-\log p_\theta(\boldsymbol{x})\right] \end{aligned}$$

Flow-based Generative models

A simple base distribution $q_0(\boldsymbol{x}) \longrightarrow$ A challenging target distribution $q_1(\boldsymbol{x})$

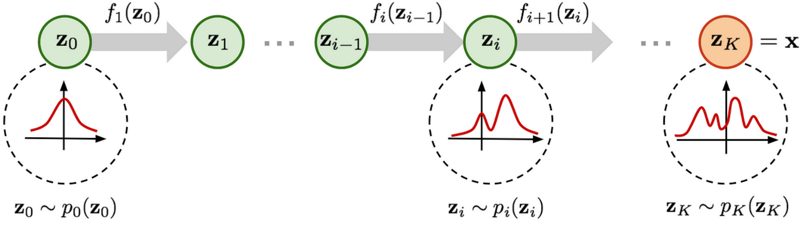
- **Goal**: To design a pushforward $f : \mathbb{R}^d \to \mathbb{R}^d$ mapping $q_0(\boldsymbol{x})$ to $q_1(\boldsymbol{x})$ that satisfies

$$q_1(\boldsymbol{x}) = q_0\left(f^{-1}(\boldsymbol{x})\right)\left|\det\left(\frac{\partial f^{-1}}{\partial \boldsymbol{x}}\right)\right|$$

- **Methodology**: Parametrize f_θ with a neural network θ and train with MLE

$$\min_\theta \mathbb{E}_{\boldsymbol{x}\sim q_1}\left[-\log q_0\left(f_\theta^{-1}(\boldsymbol{x})\right) - \log\left|\det\left(\frac{\partial f_\theta^{-1}}{\partial \boldsymbol{x}}\right)\right|\right]$$

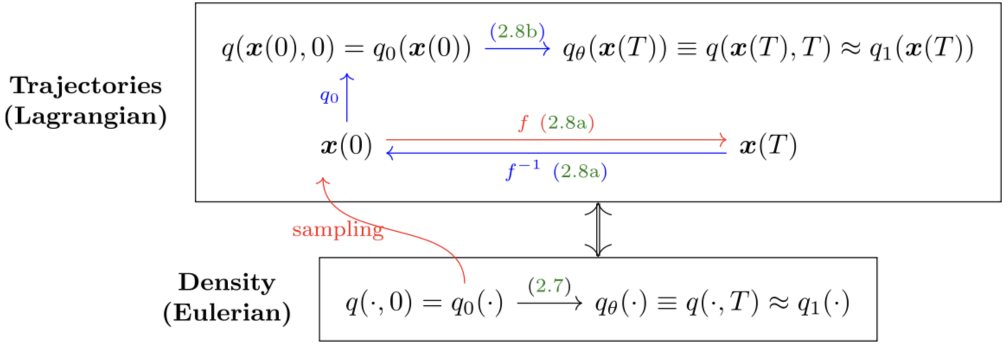
- **Examples**: Normalizing flow (NICE, RealNVP, MAF, Glow, etc.)



Continuous-time Flow Models

Regard f as the result of a flow that pushes the density $q(\boldsymbol{x}, t)$, with $q(\boldsymbol{x}, 0) = q_0(\boldsymbol{x})$, over time t while conserving total probability mass.

- **Related concepts**:
  - *Continuity equation*: $\frac{\partial q(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot [q(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x})] = 0$
  - *Brenier theorem*: $\boldsymbol{v}(\boldsymbol{x}) = \nabla\phi(\boldsymbol{x})$
  - *Lagragian formulation*: $\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \nabla\phi(\boldsymbol{x}(t))$, $\frac{\mathrm{d}q(\boldsymbol{x}(t), t)}{\mathrm{d}t} = -q(\boldsymbol{x}(t), t)\nabla^2\phi(\boldsymbol{x}(t))$
- **Methodology**: Parametrize $\phi_\theta(\boldsymbol{x})$ with a neural network and train with MLE

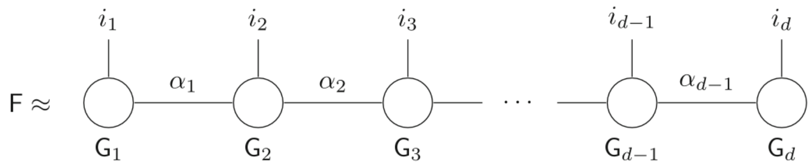


Discrete TT representation of a d-tensor $\mathsf{F}(i_1, \cdots, i_d)$

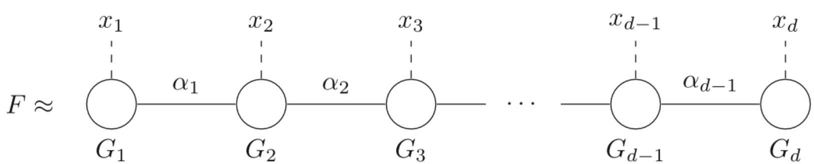
$$\mathsf{F}(i_1, \ldots, i_d) \approx \mathsf{G}_1(i_1, :)\mathsf{G}_2(:, i_2, :) \cdots \mathsf{G}_d(:, i_d),$$

Continuous TT representation of a d-dimensional function $F(x_{1:d})$

$$F(x_{1:d}) \approx \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1(x_1, \alpha_1)G_2(\alpha_1, x_2, \alpha_2) \cdots G_d(\alpha_{d-1}, x_d),$$



(a) Discrete tensor-train representation



(b) Continuous tensor-train representation

Challenges of Flow-based Models

- **Limited Expressivity:** Requires **highly expressive** functions to capture complex distributions $q_1(\boldsymbol{x})$
- **Computational Cost:** Intensive to evaluate function f and its Jacobian $\det\left(\frac{\partial f^{-1}}{\partial \boldsymbol{x}}\right)$
- **Mode Collapse:** Struggles with multi-modal distributions

Challenges of TT Representations

- **Inflexibility:** Limited in representing complex distributions
- **Strong Ansatz:** Leads to reduced spatial correlation
- **Truncation Error:** Arises from assumptions on bond dimensions (or ranks) r_i
- **Training Difficulty:** Presents a highly non-convex optimization challenge

How can we synergize the strengths of both models?

Tensorizing Flow

$$p^{\mathsf{E}}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(\cdot - \boldsymbol{x}^{(i)}\right) \xrightarrow{1} p^{\mathsf{TT}}(\cdot) \xrightarrow{2} p_\theta^{\mathsf{TF}}(\cdot) := q_\theta(\cdot) \approx p^*(\cdot)$$

1. Construct the approximate TT representation $p^{\mathsf{TT}}(\boldsymbol{x})$ from the set $\{\boldsymbol{x}^{(i)}\}_{1\le i\le N}$.
2. Define the potential function $\phi_\theta(\boldsymbol{x})$, parameterized by a neural network θ. Initialize $q(\boldsymbol{x}, 0) = p^{\mathsf{TT}}(\boldsymbol{x})$ and develop $q_\theta(\boldsymbol{x}) = q(\boldsymbol{x}, T)$.
3. Train the neural network using the set $\{\boldsymbol{x}^{(i)}\}_{1\le i\le N}$ to minimize the loss function:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim p^{\mathsf{E}}}\log p_\theta^{\mathsf{TF}}(\boldsymbol{x})$$

Main Advantages

- **Enhanced Expressivity**: $p^{\mathsf{TT}}(\boldsymbol{x})$ effectively captures multi-modality
- **Flexibility**: The subsequent NN-based flow refines density estimation
- **Reduced Computational Cost**: The near-identity nature of $\nabla\phi_\theta(\boldsymbol{x})$ allows for a simpler neural network to parameterize the flow

Ideal Case: Recover finite-rank and Markovian density p

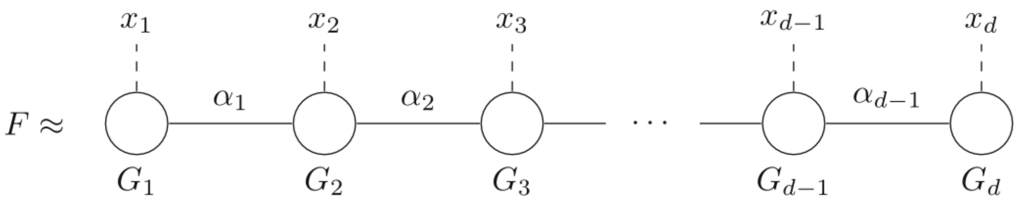
Assumptions

- **Finite-rank**: For $1 \leq k \leq d-1$, the *rank* of the reshaped version $p(x_{1:k}; x_{k+1:d})$ is r_k, *i.e.* $p(x_{1:k}; x_{k+1:d})$ as a *Hilbert-Schmidt kernel* admits the following *Schmidt decomposition*:

$$p(x_{1:k}; x_{k+1:d}) = \sum_{\alpha_k=1}^{r_k} \Phi_k(x_{1:k}; \alpha_k) \Psi_k(\alpha_k; x_{k+1:d})$$

- **Markovian**: The density function $p(\boldsymbol{x})$ is *Markovian*, *i.e.*

$$p(x_{1:d}) = p(x_1)p(x_2|x_1) \cdots p(x_d|x_{d-1})$$



Theorem 1 (Core determining equations)

Under the assumptions above, there exists a unique solution $G_1 : I \times [r_1] \to \mathbb{R}$, $G_2 : [r_1] \times I \times [r_2] \to \mathbb{R}, \ldots,$ $G_d : [r_{d-1}] \times I \to \mathbb{R}$ *to the following system of* core determining equations (CDEs)*:*

$$G_1(x_1; \alpha_1) = \Phi_1(x_1; \alpha_1),$$

$$\sum_{\alpha_{k-1}=1}^{r_{k-1}} \Phi_{k-1}(x_{1:k-1}; \alpha_{k-1}) G_k(\alpha_{k-1}; x_k, \alpha_k) = \Phi_k(x_{1:k-1}; x_k, \alpha_k), \; 2 \leq k \leq d-1,$$

$$\sum_{\alpha_{d-1}=1}^{r_{d-1}} \Phi_{d-1}(x_{1:d-1}; \alpha_{d-1}) G_d(\alpha_{d-1}; x_d) = p(x_{1:d-1}; x_d),$$

with

$$p(\boldsymbol{x}) = G_1(x_1, :)G_2(:, x_2, :) \cdots G_d(:, x_d).$$

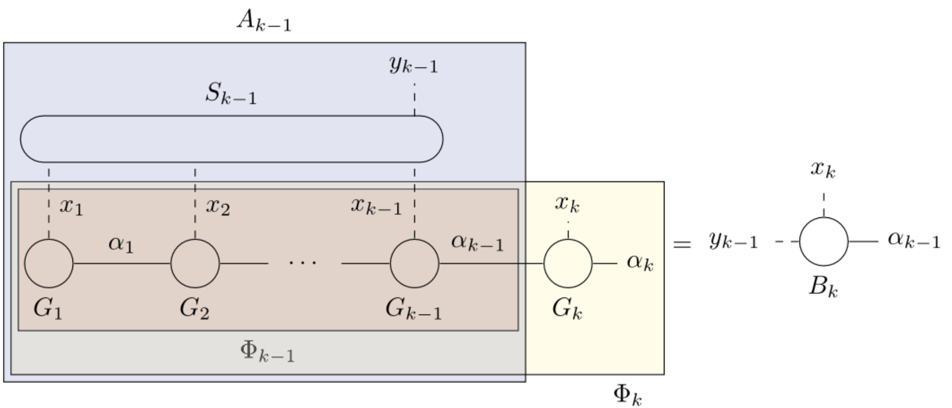
Finite-rank and Markovian $\Leftrightarrow$ Exact TT representation

Left-sketching Technique

Over-determined CDEs $$\sum_{\alpha_{k-1}=1}^{r_{k-1}} \Phi_{k-1}(x_{1:k-1};\alpha_{k-1})G_k(\alpha_{k-1};x_k,\alpha_k) = \Phi_k(x_{1:k-1};x_k,\alpha_k)$$

$$\Downarrow$$

Reduced CDEs $$\sum_{\alpha_{k-1}=1}^{r_{k-1}} A_{k-1}(y_{k-1};\alpha_{k-1})G_k(\alpha_{k-1};x_k,\alpha_k) = B_k(y_{k-1};x_k,\alpha_k)$$



How to select the left-sketching functions $S_{k-1}(y_{k-1}; x_{1:k-1})$?

Observations

Suppose $p(\boldsymbol{x})$ is Markovian, then

- $p(x_{i:k}; x_{k+1:j})$ and $p(x_{i:k}; x_{k+1})$ have the same column space
- $p(x_{i:k}; x_{k+1:j})$ and $p(x_k; x_{k+1:j})$ have the same row space

Algorithm

1. Select $S_{k-1}(y_{k-1}; x_{1:k-1}) = \delta(y_{k-1} - x_{k-1})$, *i.e.* the operation of marginalizing out the first $k-2$ dimensions
2. Form $B_k(x_{k-1}, x_k; \alpha_k)$ with the first r_k left singular vectors of $p_k(x_{k-1}, x_k; x_{k+1})$
3. Obtain A_k by marginalizing out the first dimension of B_k

Remarks

- **The exact TT representation of any ideal (*finite-rank* and *Markovian*) density $p(\boldsymbol{x})$ can be obtained with the algorithm above**
- The algorithm only requires 2- or 3-marginals $p_k(x_{k-1}, x_k; x_{k+1})$ of $p(\boldsymbol{x})$

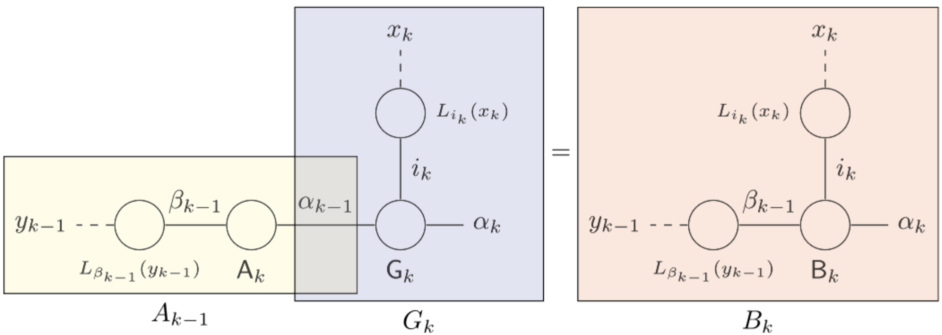
General Case: Approximate the target density $p^*(x)$

- Construct **kernel density estimators** $p_k^{\mathsf{S}}(x_{k-1:k+1})$ of the mariginals p_k^* from samples $\{\boldsymbol{x}^{(i)}\}_{1\le i\le N}$:

$$p_k^{\mathsf{S}}(x_{k-1:k+1}) := \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x_{k-1:k+1} - x_{k-1:k+1}^{(i)}}{h}\right)$$

- Discretize continuous dimensions by series expansion with **normalized Legendre polynomials**

$$\mathsf{G}_k(\alpha_{k-1}; i_k, \alpha_k) = \int_I G_k(\alpha_{k-1}; x_k, \alpha_k) L_{i_k}(x_k)\mathrm{d}x_k$$



Continuous-time Flow

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \nabla\phi_\theta(\boldsymbol{x}(t)), \quad \frac{\mathrm{d}q(\boldsymbol{x}(t),t)}{\mathrm{d}t} = -q(\boldsymbol{x}(t),t)\nabla^2\phi_\theta(\boldsymbol{x}(t))$$

- **Potential function**: $\phi_\theta(\boldsymbol{x})$ parameterized by a neural network
- **Architecture**: Four-layer MLP initialized with the identity map
- **Initial density**: Approximate TT representation, *i.e.* $q(\boldsymbol{x},0) = p^{\mathsf{TT}}(\boldsymbol{x})$
- **Final density**: $q_\theta(\boldsymbol{x}) := q(\boldsymbol{x},T) \approx p^*(\boldsymbol{x})$
- **Loss**: MLE

$$\mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim p^{\mathsf{E}}} \log q_\theta(\boldsymbol{x})$$

- **Implementation**:

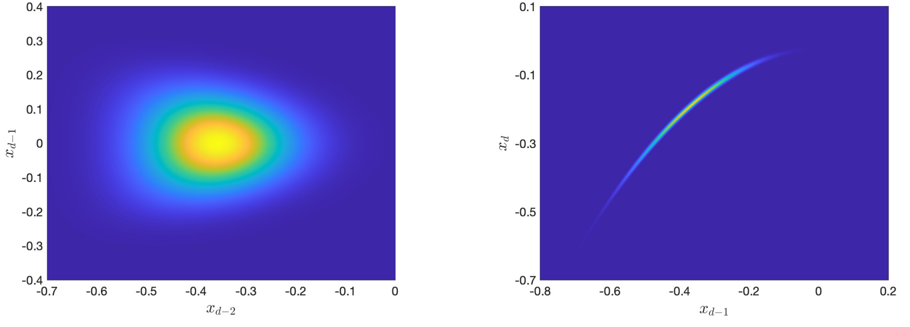
$$\boldsymbol{x}(T) \sim p^{\mathsf{E}} \xrightarrow[\phi_\theta]{\text{Runge-Kutta}} \boldsymbol{x}(0) \xrightarrow{\text{evaluate}} q(\boldsymbol{x}(0),0) = p^{\mathsf{TT}}(\boldsymbol{x}(0)) \xrightarrow[\phi_\theta]{\text{Runge-Kutta}} q(\boldsymbol{x}(T),T)$$

Rosenbrock distribution

Consider the distribution $p^*(\boldsymbol{x}) \propto \exp\left(-v(\boldsymbol{x})/2\right)$, where

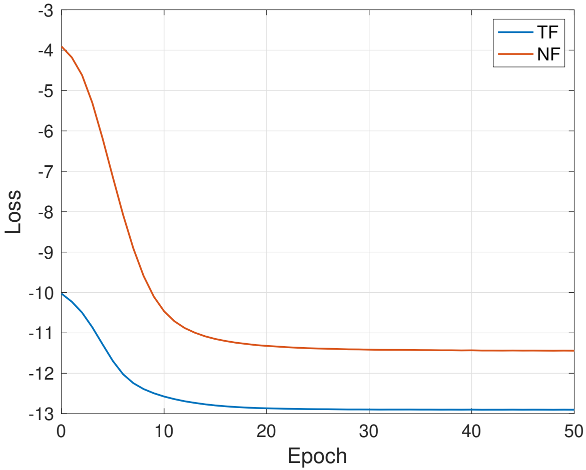
$$v(\boldsymbol{x}) = \sum_{i=1}^{d-1} \left[c_i^2 x_i^2 + \left(c_{i+1} x_{i+1} + 5(c_i^2 x_i^2 + 1) \right)^2 \right]$$

- **Parameters**: $d = 10$, $c_i = 2$, $1 \leq i \leq d-2$, $c_{d-1} = 7$, and $c_d = 200$.
- *Isotropic* in the first $d-2$ variables while *concentrated* along a curve on the last two dimensions

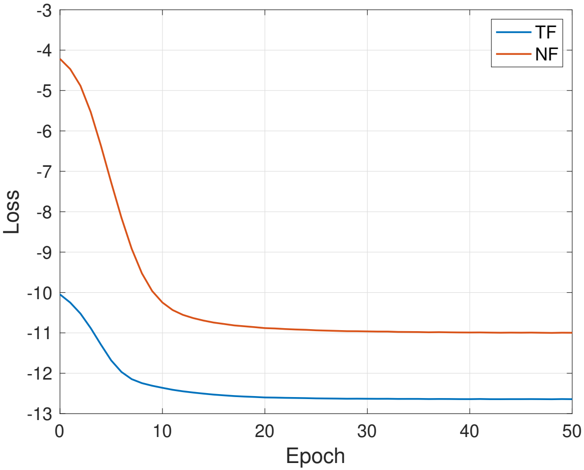


EXPERIMENTS
ROSENBROCK DISTRIBUTION

Learning curves



(a) Training loss



(b) Test loss

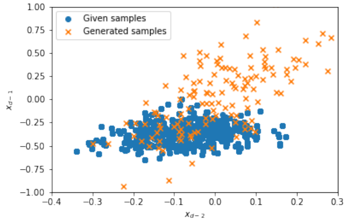
- **Tensorizing flow** outperforms **normalizing flow** in terms of both initial (approx. TT representation) and final loss (approx. TT representation + continuous-time flow).

EXPERIMENTS

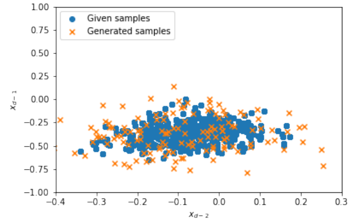
ROSENBROCK DISTRIBUTION

Sampling results

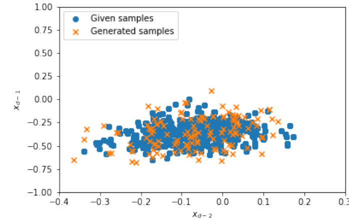
- $(d-2)$- and $(d-1)$-th dimension



(a) NF

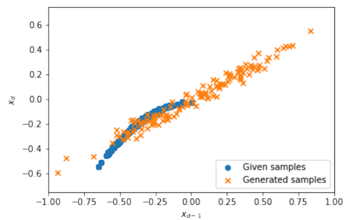


(b) TT representation

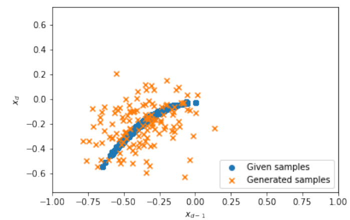


(c) TF

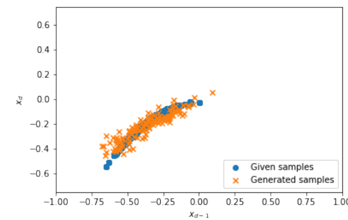
- $(d-1)$- and d-th dimension



(a) NF



(b) TT representation



(c) TF

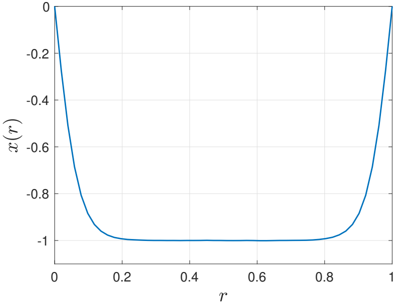
- No need for **extra-fine** grids for last two dimensions as in [1]

1D Ginzburg-Landau Distribution

Ginzburg-Landau distribution

$$\mathcal{E}[x(\cdot)] = \int_\Omega \left[\frac{\delta}{2}|\nabla_{\boldsymbol{r}} x(\boldsymbol{r})|^2 + \frac{1}{\delta}V(x(\boldsymbol{r}))\right] \mathrm{d}\boldsymbol{r},$$

where the potential $V(x) = \left(1 - x^2\right)^2/4$.

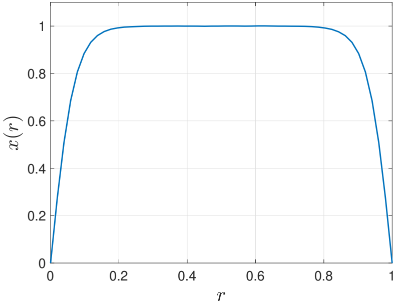


1D Ginzburg-Landau distribution

Consider the distribution $p^*(\boldsymbol{x}) \propto \exp\left(-\beta E(\boldsymbol{x})\right)$, where

$$E(\boldsymbol{x}) = \sum_{i=1}^{d+1}\left[\frac{\delta}{2}\left(\frac{x_i - x_{i-1}}{h}\right)^2 + \frac{1}{4\delta}\left(1 - x_i^2\right)^2\right]$$

- **Settings**: $\Omega = [0, L]$, $h = L/(d+1)$, $x_i \approx x(ih)$
- **Dirichlet boundary conditions**: $x_0 = x_{d+1} = 0$

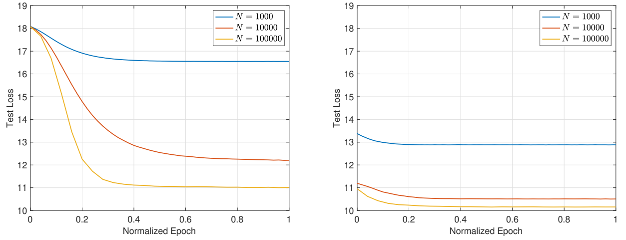


EXPERIMENTS

1D GINZBURG-LANDAU DISTRIBUTION

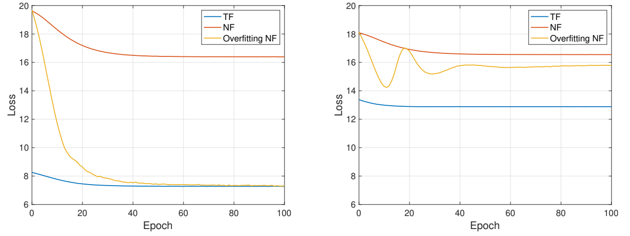
Ablation Study with $d = 16$, $\delta = 1$, $\beta = 3$

Test loss comparison w/different sample sizes N



(a) Normalizing flow **(b)** Tensorizing flow

Comparison w/different NN architectures



(a) Training loss **(b)** Test loss

- The initial approx. TT representation improves as N increases
- TF with 10^4 samples outperforms NF of the same NN architecture with 10^5 samples

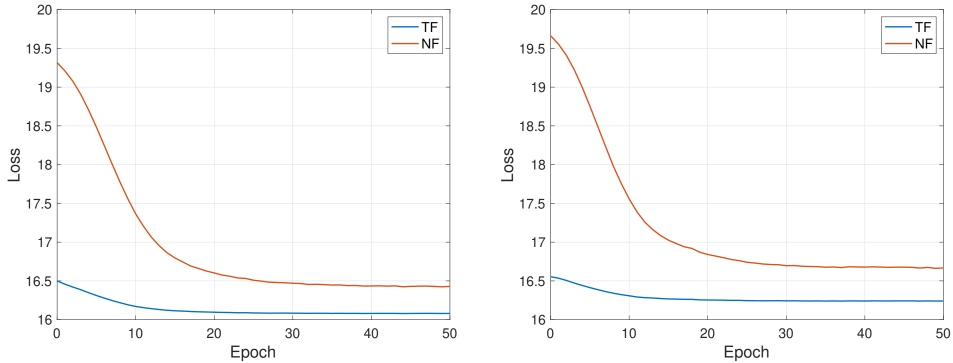
- NF with 10^6 parameters overfits significantly
- TF with 10^4 parameters outperforms NF with 10^6 parameters

2D Ginzburg-Landau distribution

Consider the distribution $p^*(\boldsymbol{x}) \propto \exp\left(-\beta E(\boldsymbol{x})\right)$, where

$$E(\boldsymbol{x}) = \sum_{i=1}^{\sqrt{d}} \sum_{j=1}^{\sqrt{d}} \left[\frac{\delta}{2} \left(\left(\frac{x_{i,j} - x_{i-1,j}}{h} \right)^2 + \left(\frac{x_{i,j} - x_{i,j-1}}{h} \right)^2 \right) + \frac{1}{4\delta} \left(1 - x_{i,j}^2 \right)^2 \right].$$

- TF learns a complicated **non-**Markovian distribution



(a) Training loss **(b)** Test loss

Discussions

Related Work: Tensorizing Flow for Variational Inference [3]

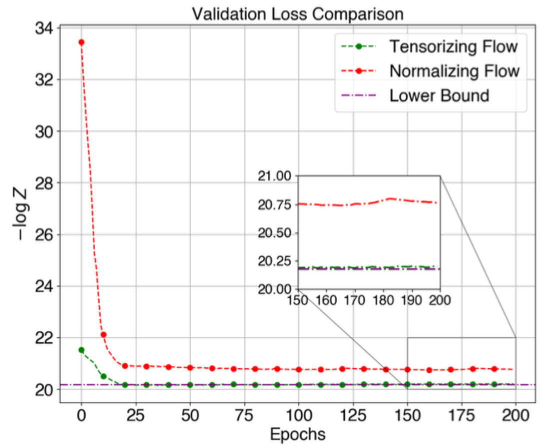
- **Goal**: Given an energy function $U : \Omega \to \mathbb{R}$, learn a distribution $p^*(\boldsymbol{x}) \propto \exp\left(-U(\boldsymbol{x})\right)$
- **Methodology**: Construct a tensorizing flow $p_\theta^{\mathsf{TF}}(\boldsymbol{x})$ and train by minimizing the KL divergence

$$\begin{aligned}\theta &= \arg\min_\theta \mathrm{D}_{\mathrm{KL}}\left(p_\theta^{\mathsf{TF}}(\cdot)\|p^*(\cdot)\right)\\ &= \arg\min_\theta \mathbb{E}_{\boldsymbol{x}\sim p_\theta^{\mathsf{TF}}}\left[\log p_\theta^{\mathsf{TF}}(\boldsymbol{x}) - \log p^*(\boldsymbol{x})\right]\\ &= \arg\min_\theta \mathbb{E}_{\boldsymbol{x}\sim p_\theta^{\mathsf{TF}}}\left[\log p_\theta^{\mathsf{TF}}(\boldsymbol{x}) + U(\boldsymbol{x})\right]\end{aligned}$$

- **Differences**:
  - Construct an approximate TT representation for $\exp(-U(\boldsymbol{x}))$
  - Draw samples from $p_\theta^{\mathsf{TF}}(\boldsymbol{x})$ instead of $p^*(\boldsymbol{x})$

Experimental Results

- Gaussian mixture distribution (multi-modal)



DISCUSSIONS

Takeaways

- **Tensorizing flow**: First to combine the flexibility of **neural networks** and the efficiency and robustness of **tensor-train representations**
- Step 1. Apply **left-sketching** and **kernel density estimation** techniques to construct an approximate TT representation
- Step 2. Adapt **continuous-time flow model** and parameterize the flow with a simple (but sufficient) neural network architecture
- Tensorizing flow
  - achieves better sample and computational efficiency than normalizing flow
  - is less prone to overfitting
  - is particularly effective for high-dimensional multi-modal distributions possibly with singularities

Future Work

- Explore other expansion bases, *e.g.* Fourier basis and Chebyshev polynomials
- Replace the continuous-time flow model with more powerful ones
- Design more adaptive schemes for non-Markovian models with more sophisticated graph structures (preliminary work by our group [5])

References I

[1] Dolgov, S., Anaya-Izquierdo, K., Fox, C., and Scheichl, R. Approximation and sampling of multivariate probability distributions in the tensor train decomposition. *Statistics and Computing 30*, 3 (2020), 603–625.

[2] Hur, Y., Hoskins, J. G., Lindsey, M., Stoudenmire, E. M., and Khoo, Y. Generative modeling via tensor train sketching. *Applied and Computational Harmonic Analysis 67* (Nov. 2023), 101575.

[3] Khoo, Y., Lindsey, M., and Zhao, H. Tensorizing flows: A tool for variational inference. *arXiv preprint arXiv:2305.02460* (2023).

[4] Ren, Y., Zhao, H., Khoo, Y., and Ying, L. High-dimensional density estimation with tensorizing flow. *Research in the Mathematical Sciences 10*, 3 (Sept. 2023), 30.

[5] Tang, X., Hur, Y., Khoo, Y., and Ying, L. Generative modeling via tree tensor network states. *arXiv preprint arXiv:2209.01341* (2022).

Thank you! Merci beaucoup!