Conditional Sampling and Density Estimation with Triangular Convex Flows

By

Zheyu Wang

Deepanshu Verma Ph.D.
Co-Advisor

Lars Ruthotto Ph.D.
Co-Advisor

Mathematics

2023

Conditional Sampling and Density Estimation with Triangular Convex Flows

By

Zheyu Wang

Deepanshu Verma Ph.D.
Co-Advisor

Lars Ruthotto Ph.D.
Co-Advisor

Abstract

Conditional Sampling and Density Estimation with Triangular Convex Flows
By Zheyu Wang

We introduce Triangular Convex Flows (TC-Flow), a method for learning conditional probability distributions given samples from the joint probability distributions. Unlike previous methods that rely on constructing monotone triangular transport maps through soft penalties and partial integration, TC-Flow uses a novel map parameterization based on the input gradient of scalar-valued fully and partially input convex neural networks (FICNN and PICNN). The approach guarantees monotone maps without requiring specific network weights and ensures optimal maps under optimal transport theory by approximating Kantorovich potentials. During training, we parallelize the process over map components and minimize the expected negative log-likelihood of the samples. We demonstrate the effectiveness of TC-Flow on synthetic two-dimensional datasets and compare it to existing triangular maps model on high-dimensional benchmark problems. Our numerical experiments show that TC-Flow is competitive with the state-of-the-art model in both expressiveness and scalability.

Conditional Sampling and Density Estimation with Triangular Convex Flows

By

Zheyu Wang

Deepanshu Verma Ph.D.
Co-Advisor

Lars Ruthotto Ph.D.
Co-Advisor

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Mathematics

2023

Acknowledgments

I firmly believe that research is the ultimate path along which humanity advances courageously and collaboratively. To work in this fantastically complicated field, one needs to possess passion, determination and knowledge. I could not have gained any of these without the help of Emory University, my committee members, and my family and friends.

I want to express my sincere gratitude towards Emory University's Applied Mathematics and Statistics program for helping me establish a solid foundation in fundamental subjects for research. The Honors program, in particular, provided me with the opportunity to explore the unknown by applying my knowledge, which kindled my passion.

The question one wishes to answer is one of the most important aspects of research. Dr. Ruthotto was the person who guided me to that question. Without his profound insights in the subject of applied mathematics and unconditional support, I could not have started and finished the project. As my honors thesis advisor, Dr. Verma helped me established confidence in studying relevant concepts behind the project and implementing the model from scratch. His unwavering support throughout this one-year journey is at the heart of the results presented in this thesis. Together, my co-advisors helped me gain immense passion towards research and fascinating knowledge in computational mathematics. I also want to express my gratitude towards Dr. McAlister who introduced me to the field of statistical learning and provided me with foundational knowledge in machine learning.

Finally, I want to express my gratitude towards my family and friends who supported me along this research journey. Without them, I could not have the finished this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conditional probability distributions play a fundamental role in describing dependence between random variables distributed according to a joint probability distribution. Direct sampling from such conditional distributions is a crucial task for numerous problems in machine learning, statistics, and engineering.

One example is the widely studied inverse problems. Generally, inverse problems involve a system $\mathbf{y} = \Psi(\mathbf{x}) + \boldsymbol{\epsilon}$, where $\mathbf{y}$ denotes noisy observations, Ψ denotes the forward operator, $\mathbf{x}$ denotes parameters, and $\boldsymbol{\epsilon}$ denotes noise. The problem's objective is to estimate or sample from the posterior distribution $\pi_{\text{pos}}(\mathbf{x}|\mathbf{y})$ in the Bayesian framework. However, since the parameters $\mathbf{x}$ are typically high-dimensional and the operator Ψ is usually non-linear, the resulting posterior can be extremely complex and impossible to sample directly from. This particular challenge extends to most other problems that involve conditional probability distributions.

To overcome such difficulty, [8] proposed a measure transport approach for understanding complex probability distributions in general. The core method is to construct invertible transport maps, or generators in machine learning literature, $g : \mathbb{R}^n \to \mathbb{R}^n$ between a reference distribution η and the complex target distribution of interest π, whose densities we denote as ρ_1 and ρ_0 respectively. Invertibility is necessary since learning direct maps g and performing density estimation require the inverse maps g^{-1}.

To introduce notations and provide an intuitive example, let us assume that samples $(x, y) \sim \pi$ are from the two-dimensional joint distribution we wish to characterize and samples $(z_x, z_y) \sim \eta$ are from the two-dimensional reference distribution. In this setting, if g is the true transport map, then it must satisfy $\rho_0(x, y) = \rho_1\left(g^{-1}(x, y)\right)$. In another word, g is said to *push forward* η to π if the previous condition is satisfied. Map g is also known as a *deterministic coupling* between η and π as $g(z_x, z_y) \sim \pi$.

With this formulation, there exist infinitely many possible maps. To construct unique transport maps that are guaranteed to be invertible, we will focus on a special type of transport maps proposed in [8]: the monotone triangular transport maps. These maps are unique under theoretical guarantee from optimal transport and are automatically invertible due to monotonicity constraints on the map components over the reference input arguments. Moreover, each component of the map exposes a coupling between the reference marginal $\eta(z_k)$ and the target conditional $\pi(x_k|x_1, \ldots, x_{k-1})$, which are pivotal for conditional sampling. For efficiency and solving problems that involve conditional sampling and density estimation, we also consider the block triangular transport maps proposed in [7].

The proposed TC-Flow method represents the strict triangular and block triangular maps using the gradient fields of input convex neural networks (ICNN)[1]. This way, monotonicity is naturally imposed due to the convex structure of the networks. TC-Flow therefore saves the computational costs induced by numerical integration to ensure monotonicity.

The TC-Flow maps are optimal as well since they are known to approximate the Kantorovich potential. TC-Flow also shares an analogous modeling scheme to normalizing flows [10] from the machine learning community. To learn the maps, we use the same maximum likelihood estimation (MLE) scheme as normalizing flows by expressing the target density using the inverse maps and the reference density through change of variable.

One crucial feature that arises from the strict triangular and block triangular map structure is that the training objective function can be separated by map components. This way, TC-Flow can construct the transport map in parallel, which significantly accelerates the training process for high-dimensional distributions.

1.1 Contributions and Outline

In this thesis, we present TC-Flow: a novel monotone parameterization of the triangular and block triangular transport maps using ICNNs. Our parameterization guarantees uniqueness and invertibility without the needs of partial integration or soft penalties. The proposed model improves upon the state-of-the-art Adaptive Transport Maps model [2] in terms of accuracy and scalability.

This thesis will include the theoretical and modeling backgrounds in Chapter 2, an elaborate explanation of TC-Flow and relevant optimization problems in Chapter 3, results from numerical experiments in chapter 4, discussion on future directions in chapter 5, and finally the conclusion in Chapter 6.

Chapter 2

Theoretical and Modeling Background

In this chapter, we will discuss the relevant theoretical and modeling background of TC-Flow. We will first present an intuitive understanding of the optimal transport problem and extend to Brenier's theorem. Then, we analyze how the theorem motivates the TC-Flow modeling framework. We then introduce the details of maximum likelihood estimation for training. Finally, we will discuss the details of the Adaptive Transport Maps algorithm, which would be the model we compare TC-Flow against.

2.1 Optimal Transport Maps

The general Monge-Kantorovich optimal transport (OT) problem can be formulated as follows [8]:

$$\min_{g} \quad \int_{\mathbb{R}^n} c\left(\mathbf{z}, g(\mathbf{z})\right)\, d\eta(\mathbf{z}). \tag{2.1}$$

Function $c : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ encodes the cost of transporting particle $\mathbf{z} \sim \eta$ to $g(\mathbf{z}) \sim \pi$, and map g is known as the transport plan bewteen distributions η and π in the OT context.

While there are infinitely many choices of g, we wish to identify the OT plan that is the unique minimizer of the objective in (2.1). Under this framework, Yann Brenier has proven a tremendously useful theorem for our discussion. Brenier's theorem [4] states that if the map g is the gradient of a convex potential C, then it is the unique solution of (2.1) when $c(\mathbf{z}, g(\mathbf{z})) = \|\mathbf{z} - g(\mathbf{z})\|^2$. In another word, $g = \nabla C$ is the OT plan under the L^2 cost. One important thing is that this theorem assumes some weak restrictions on η and π such as they are atomless. Therefore, we will assume from now that η and π are absolutely continuous.

2.2 Monotone Triangular Transport Maps

The generic monotone triangular transport map $g : \mathbb{R}^n \to \mathbb{R}^n$, proposed in [8] between the reference distribution η and the target distribution π can be formulated as follows for $\mathbf{z} \sim \eta$ and $\mathbf{x} \sim \pi$:

$$g(\mathbf{z}) = \begin{bmatrix} g_1(z_1) \\ g_2(z_2;\, x_1) \\ g_3(z_3;\, x_1, x_2) \\ \vdots \\ g_n(z_n;\, x_1, \ldots, x_{n-1}) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.2}$$

By convention, the reference η is always chosen to be the standard Gaussian distribution. Therefore, we will assume that $\eta = N(0, \mathbf{I})$ for the rest of the thesis. Each map component in (2.2) is monotone in its last input argument, i.e.,

$$\langle g_i(z_i) - g_i(z_i'),\, z_i - z_i' \rangle > 0 \;\; \forall\, z_i, z_i' \in \mathbb{R}.$$

In this setting, each component of g takes the outputs of the previous components as inputs. To be specific, $g_i(z_i|x_{<i})$ pushes forward the i^{th} marginal of the standard Gaussian η_i to the target conditional distribution $\pi(x_i|x_{<i})$. Here, the notation $x_{<i}$ represents input vector $[x_1, \ldots, x_{i-1}]^\top$. Each map component therefore characterizes a marginal conditional of the target joint distribution decomposed by chain rule:

$$\pi(\mathbf{x}) \ = \ \pi(x_1)\pi(x_2|x_1)\pi(x_3|x_1, x_2)\cdots\pi(x_n|x_{<n}).$$

Then, to sample from the k^{th} conditional assuming $k > 1$, we can provide arbitrary inputs $x^*_{<k}$ to the corresponding map component. Triangular maps of this form are also known as the Knothe-Rosenblatt rearrangement [8].

For TC-Flow, we combine the Knothe-Rosenblatt rearrangement and Brenier's Theorem to design the map components as the gradients of strictly convex functions $\Phi_i : \mathbb{R}^i \to \mathbb{R}$ such that $\nabla_i^2\Phi_i \succ 0$. Then, the map described in (2.2) becomes

$$g(\mathbf{z}) \ = \begin{bmatrix} \nabla_{z_1}\Phi_1(z_1) \\ \nabla_{z_2}\Phi_2(z_2;\, x_1) \\ \nabla_{z_3}\Phi_3(z_3;\, x_1, x_2) \\ \vdots \\ \nabla_{z_n}\Phi_n(z_n;\, x_1, \ldots, x_{n-1}) \end{bmatrix}. \qquad (2.3)$$

This map formulation is then guaranteed to be component-wise optimal under Brenier's theorem as well as invertible.

2.3 Maximum Likelihood Estimation

The notion of constructing invertible maps between a reference and a target distribution resembles the normalizing flows approach [10, 5] in the machine learning community. Generator g_Θ in this context is represented by neural networks with weights Θ. To learn the optimal Θ, normalizing flows utilize maximum likelihood estimation (MLE). The essence of MLE training is to express the approximated target log-likelihood $\log \rho_\Theta(\mathbf{x})$ with the inverse generator g_Θ^{-1} and the reference log-likelihood ρ_1 using change of variable:

$$\log \rho_0(\mathbf{x}) \approx \log \rho_\Theta(\mathbf{x}) = \log \rho_1\left(g_\Theta^{-1}(\mathbf{x})\right) + \log\left|\det \nabla g_\Theta^{-1}(\mathbf{x})\right|.$$

The training objective is then to maximize $\mathbb{E}_{\mathbf{x}\sim\pi}\left[\log \rho_\Theta(\mathbf{x})\right]$, which is equivalent to minimizing $\mathbb{E}_{\mathbf{x}\sim\pi}\left[-\log \rho_\Theta(\mathbf{x})\right]$ given training samples $\mathbf{x}^{[i]}$ from the target. This minimization problem can also be interpreted as minimizing the Kullback-Leibler divergence (KL divergence) between the approximated target density ρ_Θ and the true target density ρ_0. We can write the KL divergence as

$$D_{\mathrm{KL}}\left[\rho_0 \parallel \rho_\Theta\right] := \int_{\mathbb{R}^n} \rho_0(\mathbf{x}) \log\left(\frac{\rho_0(\mathbf{x})}{\rho_\Theta(\mathbf{x})}\right) d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim\pi}\left[\log\left(\frac{\rho_0(\mathbf{x})}{\rho_\Theta(\mathbf{x})}\right)\right].$$

Since $\rho_0(\mathbf{x})$ is independent of Θ and is unknown to our model, we will drop this term and only minimize $\mathbb{E}_{x\sim\pi}\left[-\log \rho_\Theta(\mathbf{x})\right]$ instead, which is exactly the objective of MLE.

Due to the similarity in the modeling scheme, we will adopt the same maximum likelihood training scheme to learn our triangular transport maps parameterized by ICNN weights Θ. The optimization problem associated with training is then

$$\arg\min_{\Theta} \quad \mathbb{E}_{\mathbf{x}\sim\pi}\left[-\log \rho_1\left(g_\Theta^{-1}(\mathbf{x})\right) - \log\left|\det \nabla g_\Theta^{-1}(\mathbf{x})\right|\right]. \tag{2.4}$$

We can then use algorithms like stochastic approximation (SA) or sample average approximation (SAA) algorithms to solve the problem.

2.4 Adaptive Transport Map Model

Extensive work has been done on representing and learning the triangular transport maps [2, 8, 9]. The current state-of-the-art approach is the Adaptive Transport Map (ATM) method proposed in [2]. The ATM model first learns the inverse transport map $g^{-1}(\mathbf{x})$, then inverts it to obtain the direct map. Instead of enforcing monotinicity through utilizing gradient of convex functions, the ATM method incorporates the rectifier operator R_k on a smooth non-monotone function $f_k : \mathbb{R}^k \to \mathbb{R}$:

$$R_k(f_k)(x_{\leq k}) \;=\; f_k(x_{<k}, 0) + \int_0^{x_k} g\left(\partial_k\, f_k(x_{<k}, t)\right)\, \mathrm{d}t.$$

The notation $x_{\leq k}$ represents input vector $[x_1, \ldots, x_k]^\top$ and function $g : \mathbb{R} \to \mathbb{R}_+$ is a positive function. This way, the gradient with respect to x_k is

$$\partial_k R_k(f_k)(x_{\leq k}) \;=\; g\left(\partial_k f_k(x_{\leq k})\right) \;>\; 0.$$

Given i.i.d. training samples from the target, the ATM algorithm searches for f_k in the function space V_k, which can be expressed as the tensor product of the Lebesgue spaces and the Hilbert space:

$$V_k = L^2_{\eta_1} \otimes L^2_{\eta_2} \otimes \cdots \otimes H^1_{\eta_k}.$$

The Hilbert space H^1 introduces enough regularity for the triangular maps' last map components. Using multi-index notation, the ATM algorithm learns the optimal multi-index set $\Lambda = \{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_m\}$ where $\boldsymbol{\alpha}_i = \{\alpha_1, ..., \alpha_k\}$. This way, the function f_k can be constructed as

$$f_k(x_{\leq k}) \;=\; \sum_{i=1}^{m} \left(c^i_{\boldsymbol{\alpha}_i} \prod_{j=1}^{k} \psi^j_{\alpha_j}(x_j) \right).$$

Here $c^i_{\boldsymbol{\alpha}_i} \in \mathbb{R}$, and $\{\psi^j_{\alpha_j}\}_{\alpha \in \mathbb{N}_0}$ is chosen to be basis of $L^2_{\eta_j}$ if $j < k$ and of $H^1_{\eta_k}$ if $j = k$. Certainly, identifying the appropriate basis is pivotal for the ATM algorithm. The current canonical choice of basis is the Probabilists' Hermite polynomials with linearization outside of an empirical interval [0.01, 0.99].

The ATM algorithm initializes $\Lambda_0 = \emptyset$ and adaptively add new optimal $\boldsymbol{\alpha}^*$ into the set:

$$\Lambda_{t+1} = \Lambda_t \cup \{\boldsymbol{\alpha}^*_t\} \quad \text{such that} \quad \boldsymbol{\alpha}^*_t \notin \Lambda_t.$$

One interesting fact is that the algorithm searches for both the optimal cardinality of the multi-index set Λ as well as the optimal multi-indices values in the reduced margin set:

$$\Lambda^{RM}_t = \{\boldsymbol{\alpha} \notin \Lambda_t \mid \boldsymbol{\alpha} - \mathbf{e}_i \in \Lambda_t \; \forall i \in [1, k] \;\text{ with }\; \alpha_i \neq 0\}.$$

This way, the size of the search space for new multi-indices does not grow too fast as dimension increases.

Chapter 3

Formulation and Analysis of Triangular Convex Flows

In this chapter, we will explain the details of the model architecture and the associated optimization problems of our TC-Flow model. We start off by introducing the ICNNs parameterized triangular maps and the block triangular maps. Then we will discuss the non-linear and convex optimization problems we solve respectively for training and inverting TC-Flow.

3.1 Input Convex Neural Networks

To represent the convex potentials Φ_i in (2.3), we will use the FICNN F_μ and PICNN P_θ proposed in [1]. A m-layer FICNN can be expressed mathematically as the sequence

$$\begin{aligned}&\mathbf{z}_{k+1} = \sigma_k\left(\mathbf{L}_k^{(z)+}\mathbf{z}_k + \mathbf{L}_k^{(y)}\mathbf{y} + \mathbf{b}_k\right),\ \text{for } k = 0, \dots, m-1\\ &F_\mu(\mathbf{y}) = \mathbf{z}_m, \quad \mathbf{z}_0 = \mathbf{y}.\end{aligned}$$

Here $\mathbf{z}_k$ denotes the intermediate layer outputs, σ_k denotes the activation function, and $\mu = \{\mathbf{L}^{(z)}_{0:m-1}, \mathbf{L}^{(y)}_{0:m-1}, \mathbf{b}_{0:m-1}\}$ are the parameters. For the fist network layer, $\mathbf{L}^{(y)}_0 = 0$. Superscripts (z) and (y) denote the "path" with which the weights are associated with. To ensure convexity in input $\mathbf{y}$, the weights $\mathbf{L}^{(z)}_k$ are constrained to be non-negative, denoted by the "+" sign, and the activation functions σ_k are chosen to be convex and non-decreasing. The resulting convexity can be easily proved based on the operations preserving convexity described in [3].

Now, a m-layer the PICNN can be expressed in a similar fashion:

$$\begin{aligned}
&\mathbf{u}_{k+1} = \sigma^{(u)}_k \left(\mathbf{L}^{(u)}_k \mathbf{u}_k + \mathbf{b}^{(u)}_k\right), \\
&\mathbf{z}_{k+1} = \sigma^{(z)}_k \Bigg(\mathbf{L}^{(z)+}_k \left(\mathbf{z}_k \circ [\mathbf{L}^{(zu)}_k \mathbf{u}_k + \mathbf{b}^{(zu)}_k]_+\right) + \\
&\qquad \mathbf{L}^{(y)}_k \left(\mathbf{y} \circ [\mathbf{L}^{(yu)}_k \mathbf{u}_k + \mathbf{b}^{(yu)}_k]\right) + \mathbf{L}^{(uz)}_k \mathbf{u}_k + \mathbf{b}_k\Bigg), \\
&\text{for } k = 0, \dots, m-1 \\
&\mathbf{u}_0 = \mathbf{x}, \quad \mathbf{z}_0 = \mathbf{y}, \quad P_\theta(\mathbf{x}, \mathbf{y}) = \mathbf{z}_m.
\end{aligned}$$

Here θ are the parameters, and $\mathbf{L}^{(y)}_0, \mathbf{L}^{(yu)}_0, \mathbf{b}^{(yu)}_0 = 0$. The PICNN is convex in $\mathbf{y}$ but not necessarily in $\mathbf{x}$, and $\mathbf{u}_k$ are the layer activations of input $\mathbf{x}$. The The symbol $\circ$ denotes the Hadamard product, or element-wise product, between matrices. Similar to FICNN, to guarantee convexity, the weights $\mathbf{L}^{(z)}_k$ and the term $\mathbf{L}^{(zu)}_k \mathbf{u}_k + \mathbf{b}^{(zu)}_k$ are constrained to be non-negative, and activation functions $\sigma^{(z)}_k$ are convex non-decreasing.

3.2 ICNN Triangular Maps

With the FICNN and PICNN, we can parameterize the convex potentials Φ_i. However, since the training problem (2.4) only involves the inverse map and its gradient, we will use the ICNNs to represent the inverse convex potentials Φ^{-1}_i whose gradients are the inverse map components. This way

$$g_{\mu,\theta}^{-1}(\mathbf{x}) = \begin{bmatrix} \nabla_{x_1} F_\mu(x_1) \\ \nabla_{x_2} P_{\theta_2}(x_1, x_2) \\ \vdots \\ \nabla_{x_n} P_{\theta_n}(x_1, \dots, x_n) \end{bmatrix}, \tag{3.1}$$

and each ICNN is only convex in the last argument. With this formulation, the direct transport map in (2.3) becomes

$$g_{\mu,\theta}(\mathbf{z}) = \begin{bmatrix} \nabla_{z_1} F_\mu^{-1}(z_1) \\ \nabla_{z_2} P_{\theta_2}^{-1}(z_2; x_1) \\ \vdots \\ \nabla_{z_n} P_{\theta_n}^{-1}(z_n; x_1, \dots, x_{n-1}) \end{bmatrix}. \tag{3.2}$$

The i^{th} map component in (3.2) is automatically monotone in z_i due to the ICNN convexity. Worth noting is that conditional sampling using (3.2) only requires the PICNN parameterized components. In this sense, we can replace the F_μ in (3.1) with the identity mapping and learn only the other components. Then, for example, to sample from the conditional $\pi(x_4|x_3, x_2, x_1)$, we can provide arbitrary inputs x_1^*, x_2^* and compute $\nabla_{z_4} P_{\theta_4}^{-1}(z_4; x_1^*, x_2^*, x_3)$. Note that we obtain input x_3 from $\nabla_{z_3} P_{\theta_3}^{-1}(z_3; x_1^*, x_2^*)$.

3.3 ICNN Block Triangular Maps

To highlight the concept of conditional distribution, we now denote the joint target distribution as $\pi(\mathbf{x}, \mathbf{y})$ and the reference distribution as $\eta(\mathbf{z}_\mathbf{x}, \mathbf{z}_\mathbf{y})$ where $\mathbf{y}$ is the conditional input. As discussed in [7], the lower-triangular maps has one disadvantage: sensitive to variable ordering. One trivial example in $\mathbb{R}^2$ is that learning the following maps,

$$g = \begin{bmatrix} g_1(z_x) \\ g_2(z_y; x) \end{bmatrix} \quad \text{and} \quad g = \begin{bmatrix} g_1(z_y) \\ g_2(z_x; y) \end{bmatrix},$$

poses drastically different challenges for some problem. To address this, [7] proposed the block triangular maps. For TC-Flow, the direct block triangular maps can be written as follows:

$$g_{\mu,\theta}(\mathbf{z_x}, \mathbf{z_y}) = \begin{bmatrix} g_\mu(\mathbf{z_y}) \\ g_\theta\left(\mathbf{z_x}; \mathbf{y}\right) \end{bmatrix} = \begin{bmatrix} \nabla_{\mathbf{z_y}} F_\mu^{-1}(\mathbf{z_y}) \\ \nabla_{\mathbf{z_x}} P_\theta^{-1}(\mathbf{z_x}; \mathbf{y}) \end{bmatrix} \tag{3.3}$$

In this formulation, F_μ^{-1} is convex in all inputs and P_θ^{-1} is only convex in $\mathbf{z_x}$. Components g_θ is the couplings between the reference marginals and the target marginal conditionals. Such block triangular maps are more scalable compared to strict triangular maps in that it only involves two ICNNs and are less sensitive to variable ordering.

The block triangular maps in (3.3) are also well-suited for conditional sampling and density estimation, which can be achieved by setting F_μ to be the identity map and only learn the PICNN map. Considering these benefits, we will only adopt such block-triangular maps when learning high-dimensional target distributions. Finally, to ensure the convex potentials are strongly convex, we will add a trainable quadratic term to F_μ and P_θ like did in [6]. To illustrate, we will use the block triangular maps and the associated inverse potentials are:

$$\begin{aligned} \Phi_\mu^{-1} &= g(w_1) \cdot g(F_\mu\left(\mathbf{y}\right)) + (\mathrm{ReLU}(w_2) + g(w_3)) \cdot \frac{1}{2}\|\mathbf{y}\|_2^2 \\ \Phi_\theta^{-1} &= g(w_1) \cdot g(P_\theta\left(\mathbf{y}, \mathbf{x}\right)) + (\mathrm{ReLU}(w_2) + g(w_3)) \cdot \frac{1}{2}\|\mathbf{x}\|_2^2 \end{aligned}$$

The scalars w_1, w_2, w_3 are the trainable weights for the network outputs and the quadratic terms, respectively. Function g is chosen to be the softplus function $g(x) = \log(1 + \exp(x))$.

3.4 Training Problem

There are two crucial optimization problems associated with our model: the non-linear training problem and the convex inversion problem. First, I want to discuss the training problem for the strict triangular case of (2.2) then extend to the block triangular maps. With the ICNN parameterization, the minimization problem in (2.4) for strict triangular maps becomes:

$$\underset{\mu,\boldsymbol{\theta}}{\arg\min} \quad \mathbb{E}_{\mathbf{x}\sim\pi}\left[-\log\ \rho_1\left(g^{-1}_{\mu,\boldsymbol{\theta}}(\mathbf{x})\right) - \log\det\nabla g^{-1}_{\mu,\boldsymbol{\theta}}(\mathbf{x})\right]. \tag{3.4}$$

The notation $\boldsymbol{\theta}$ represents all the n-1 sets of PICNN weights. Here, the absolute value around the determinant is omitted since by construction the determinant will be positive.

During training, we will use SA to approximate the expectation in (3.4) of mini batch samples $x^{[i]}, i = 1, \ldots, M$:

$$J_{\mu,\boldsymbol{\theta}} = -\frac{1}{M}\sum_{i=1}^{M}\left[\ \log\rho_1\left(g^{-1}_{\mu,\boldsymbol{\theta}}(\mathbf{x}^{[i]})\right)\ +\ \log\det\nabla g^{-1}_{\mu,\boldsymbol{\theta}}(\mathbf{x}^{[i]})\right]. \tag{3.5}$$

Following [8], since the multivariate Gaussian can be expressed as the product of the marginals, we can separate $J_{\mu,\boldsymbol{\theta}}$ into independent optimization problems for each of map component. For the sake of illustration, I will only write out the training objective for the k^{th} map component such that $k > 1$:

$$J_{\theta_k} \ = \ -\frac{1}{M}\sum_{i=1}^{M}\left[\log\ \rho_1^{[k]}\left(g_{\theta_k}^{-1[k]}\left(x_{\leq k}^{[i]}\right)\right) + \log\ \nabla_{x_k}\left(g_{\theta_k}^{-1[k]}\left(x_{\leq k}^{[i]}\right)\right)\right].$$

The superscript $[k]$ represent the k^{th} component. This way, we can train the ICNN maps in parallel to learn the optimal ICNN parameters.

Now for the block triangular case, the objective function is also separable. For the block triangular maps, the gradient of the inverse map has the following lower block triangular structure:

$$\nabla g_{\mu,\theta}^{-1}(\mathbf{x},\mathbf{y}) = \begin{bmatrix} \nabla_{\mathbf{y}}^2 F_\mu(\mathbf{y}) & 0 \\ \nabla_{\mathbf{y}}\nabla_{\mathbf{x}} P_\theta(\mathbf{y},\mathbf{x}) & \nabla_{\mathbf{x}}^2 P_\theta(\mathbf{y},\mathbf{x}) \end{bmatrix}$$

Assuming that $(\mathbf{x},\mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^q$, we can write out the training objective function as

$$J_{\mu,\theta} = -\frac{1}{M}\sum_{i=1}^{M} \left[\log\, \rho_1\left(g_{\mu,\theta}^{-1}(\mathbf{x},\mathbf{y})\right) + \log\left(\det \mathbf{H}_F \det \mathbf{H}_P\right)\right]$$

Here $\mathbf{H}_F = \nabla_{\mathbf{y}}^2 F_\mu(\mathbf{y}) \in \mathbb{R}^{q\times q}$ and $\mathbf{H}_P = \nabla_{\mathbf{x}}^2 P_\theta(\mathbf{y},\mathbf{x}) \in \mathbb{R}^{p\times p}$ are the block diagonal components of the gradient. This way, we have two independent optimization objectives for the FICNN and PICNN respectively:

$$J_\mu = -\frac{1}{M}\sum_{i=1}^{M} \left[\log \rho_1\left(\nabla_{\mathbf{y}} F_\mu(\mathbf{y}^{[i]})\right) + \log\det \nabla_{\mathbf{y}}^2 F_\mu(\mathbf{y}^{[i]})\right]$$
$$J_\theta = -\frac{1}{M}\sum_{i=1}^{M} \left[\log \rho_1\left(\nabla_{\mathbf{x}} P_\theta(\mathbf{y}^{[i]},\mathbf{x}^{[i]})\right) + \log\det \nabla_{\mathbf{x}}^2 P_\theta(\mathbf{y}^{[i]},\mathbf{x}^{[i]})\right]$$

Finally, to solve the training problems, we will use the Adam stochastic optimization algorithm. During training, after each weight update using Adam, we project the ICNN weights under non-negative constraint into the non-negative orthant using Rectified Linear Units (ReLU): $\mathrm{ReLU}(x) = \max(x, 0)$.

3.5 Inversion Problem

The ultimate goal of our model is to learn the direct transport maps. Therefore, we need to invert the learnt inverse maps obtained from training via MLE. This can be readily done by solving convex optimization problems as proposed in [6]. To invert the FICNN parameterized block triangular map, we solve

$$\arg\min_{\mathbf{v}} \quad F_\mu(\mathbf{v}) \; - \; \mathbf{z}_\mathbf{y}^\top \mathbf{v}.$$

The objective function in this problem is clearly convex in $\mathbf{v}$. Here $\mathbf{z}_\mathbf{y} \sim N(0, \mathbf{I})$ where $\mathbf{I} \in \mathbb{R}^{q \times q}$, and the minimizer $\mathbf{v}^* = \nabla_{\mathbf{z}_\mathbf{y}} F_\mu^{-1}(\mathbf{z}_\mathbf{y})$ are the direct map outputs, or generated samples, $\mathbf{y}$.

Now for the PICNN parameterized block triangular map component, we solve

$$\arg\min_{\mathbf{v}} \quad P_\theta\left(\mathbf{y},\ \mathbf{v}\right) \; - \; \mathbf{z}_\mathbf{x}^\top \mathbf{v}.$$

Here $\mathbf{z}_\mathbf{x} \sim N(0, \mathbf{I})$ where $\mathbf{I} \in \mathbb{R}^{p \times p}$ and the minimizer $\mathbf{v}^* = \nabla_{\mathbf{z}_\mathbf{x}} P_\theta^{-1}\left(\mathbf{z}_\mathbf{x};\ \mathbf{y}\right)$ are the generated samples $\mathbf{x}$. To sample conditionally, we simply need to provide arbitrary input $\mathbf{y}^*$ to P_θ.

To solve the inversion problems, we will use the Limited-memory BFGS algorithm with strong Wolfe line search. Other alternatives such as Newton's method with backtracking line search are also feasible.

Chapter 4

Numerical Experiments

We will demonstrate TC-Flow's performance on generative sampling and density estimation through numerical experiments. We first illustrate the efficacy of TC-Flow using two-dimensional synthetic datasets. Then, we compare TC-Flow's performance against ATM on high-dimensional benchmark problems. All experiments are conducted on one Quadro RTX 8000 GPU with 48 GB of RAM. Code to reproduce all experiments is available in `https://github.com/OliverWang7/TC-Flow`.

4.1 Two Dimensional Synthetic Data

To demonstrate TC-Flow's effectiveness in characterizing the complex target distributions, we will train the model on 2-D synthetic datasets distributed according to non-Gaussian and disjoint distributions. To set up the experiment, we prepare 30000 samples from the target as training data.

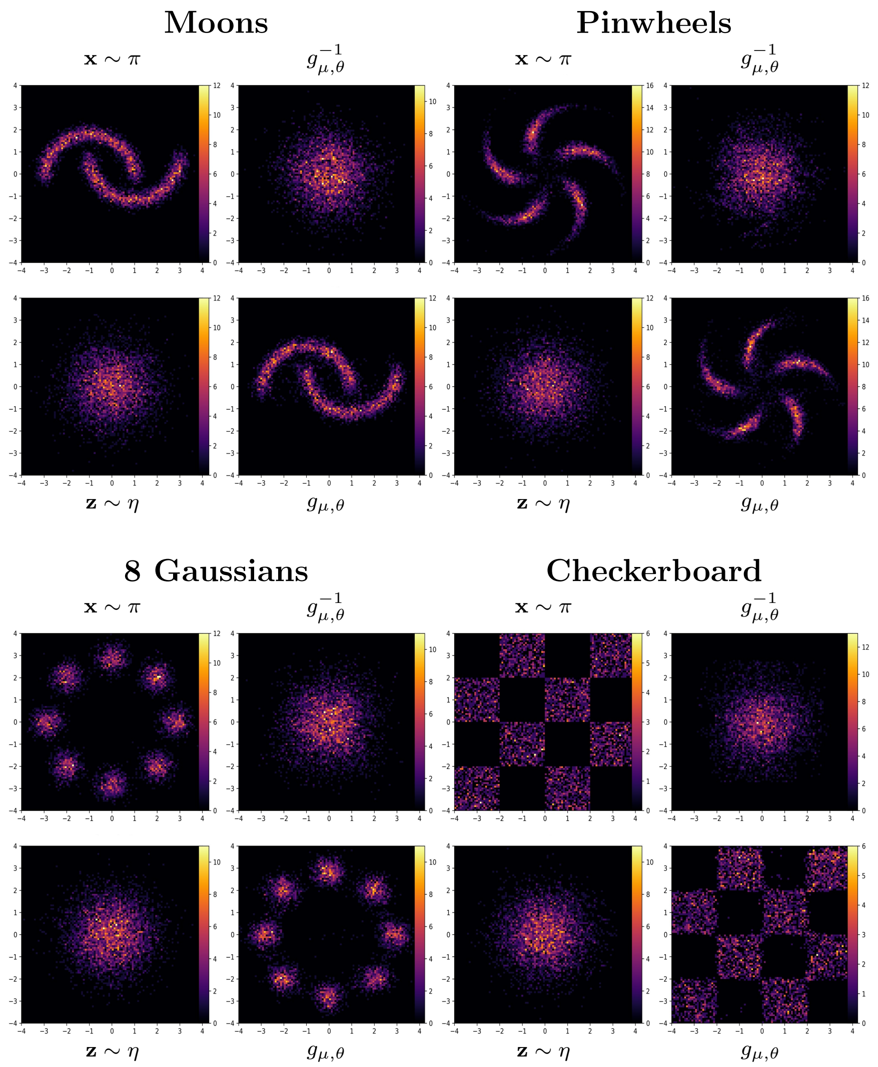

Figure 4.1: 2-D toy dataset generative sampling. **Top Left**: samples from the target distribution. **Top Right**: distribution of the inverse map outputs. **Bottom Left**: samples from the reference distribution. **Bottom Right**: generated samples.

We then use the PyTorch Adam optimizer with batch size of 64 and learning rate of 0.005 to optimize the ICNNs' weights. Based on the 2-D histogram plots in Figure 4.1, we see a good match between the actual target samples and the generated samples.

4.2 Tabular Dataset

We then compare TC-Flow's performance on the Wine Quality and Parkinsons datasets from the UCI machine learning repository to ATM's. To pre-process the datasets, we remove discrete valued data features and normalize each feature by subtracting the empirical mean then dividing by the empirical standard deviation. Then, we remove one variable of every pair with a Pearson correlation coefficient greater than 0.98.

For evaluation metric, we adopt the mean negative log-likelihood, which is exactly the training loss. To construct the maps, we partition the sample features into two halves then learn the joint distribution of the first half using a FICNN and the conditional distribution of the second half using a PICNN, which results in a block triangular map.

The scalar-valued FICNN and PICNN both have layer width of 256 and depth of 3 hidden layers. We use the PyTorch Adam optimizer for the training problem with learning rate=0.005, $\beta = (0.9, 0.999)$, $\epsilon = 1e - 08$, and weight decay of 0. We initialize both weights for the quadratic term to 0.1, and for the ICNN output to 0. We use a 8/1/1 split over the datasets and use 1 fold for validation and 1 fold for testing. For the Wine Quality dataset (Red Wine and White Wine), we use a batch size of 32 and 64 for Parkinsons. We evaluate our model using the validation set every 20 training steps and save the best model based on validation loss. Finally, we test the TC-Flow maps on the hold-out test dataset. The following TC-Flow results are obtained from 10 runs on GPU and the ATM results are reported in the paper [2].

Datasets	(d, nex)	ATM	TC-Flow
Red	(11, 1599)	9.8 ± 0.4	8.98 ± 0.16
White	(11, 4898)	11.0 ± 0.2	11.01 ± 0.12
Parkinson's	(15, 5875)	2.8 ± 0.4	2.71 ± 0.08

Table 4.1: Test mean negative log likelihood (lower the better)

To visualize TC-Flow's performance on the high-dimensional datasets, we project the results onto two dimensions and obtain 2-d histograms:

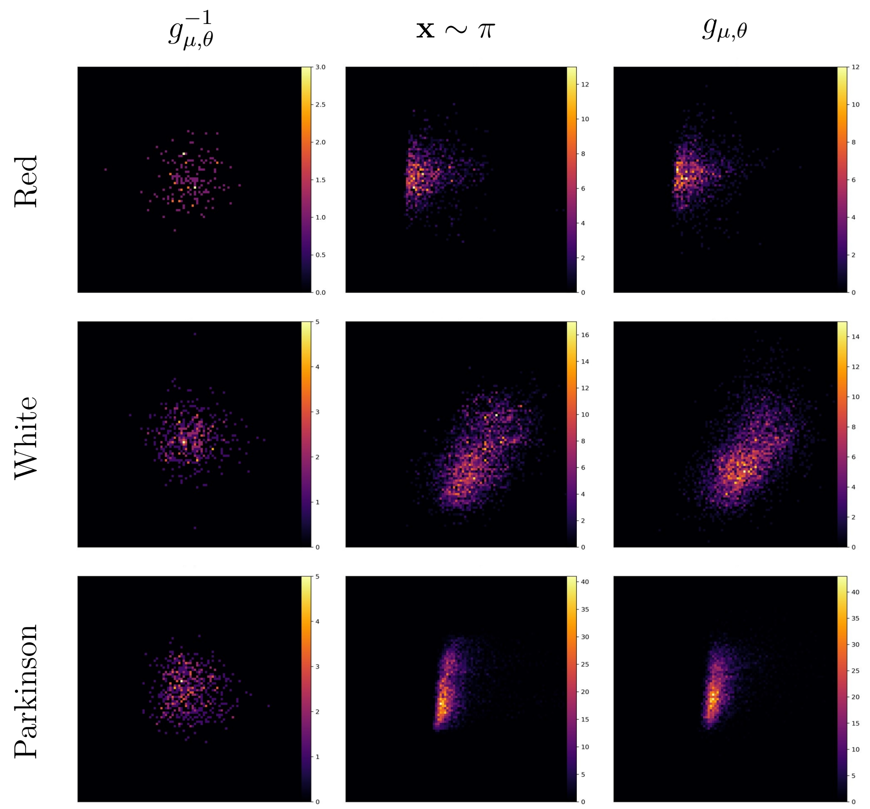

Figure 4.2: Tabular dataset generative sampling. **Left**: distribution of the inverse maps outputs. **Middle**: samples from the target distributions. **Right**: generated samples.

Chapter 5

Discussions and Future Directions

We presented a scalable and expressive monotone triangular and block triangular map parameterization using TC-Flow. The numerical results suggest that TC-Flows performs better in terms of mean negative log-likelihood than the ATM model on the Red Wine dataset and comparable for the other two datasets. The primary reason behind such improvements is that the ATM model does not capture sharp boundaries in the target distributions well. For example, our model learns the two-dimensional projected distribution for Red Wine dataset in Figure 4.2 that exhibits a vertical boundary, while the ATM model do not capture such boundary.

One property of strict triangular maps (2.2) is that they preserve sparsity [2]. According to [11], the sparsity pattern J_T of triangular maps for the k^{th} map component is defined as

$$J_T = \left\{(j,k) : j < k,\ \partial_j \nabla_k \Phi_k = 0\right\}, \tag{5.1}$$

if the k^{th} variable is independent of the j^{th} variable. As shown in [2], the ATM model learns triangular maps that inherit the intrinsic sparsity pattern to the dataset. The metric to measure sparsity pattern is through $\int |\partial_j \partial_k \Phi(x)|^2 \pi(x)\,\mathrm{d}x$ from test samples. Based on our evaluations using this metric, the TC-Flow block triangular maps do not preserve sparsity. However, the strict triangular maps learned by TC-Flow do exhibit sparsity. Therefore, one interesting future experiment could be to study the sparsity-efficiency trade-off between the triangular maps versus block triangular maps. Furthermore, if we have knowledge of the sparsity pattern a priori, we can construct the strict or block triangular maps such that (5.1) automatically holds for variables with conditional independence.

Chapter 6

Concluding Remarks

We presented TC-Flow, an ICNN based approach for conditional sampling using monotone triangular transport maps. The method characterizes complex conditional distributions with transport maps that are guaranteed to be optimal and invertible by theory. Numerical experiments demonstrate the effectiveness of TC-Flow in density estimation and generative sampling on benchmark problems. Future efforts could exploit conditional independence for map representation and learning, analyze other optimization algorithms, and explore better ICNN architectures.

Bibliography

[1] Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 06–11 Aug 2017.

[2] Ricardo Baptista, Youssef Marzouk, and Olivier Zahm. On the representation and learning of monotone triangular transport maps, 2022.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[4] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

[6] Chin-Wei Huang, Ricky T. Q. Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2021.

[7] Nikola Kovachki, Ricardo Baptista, Bamdad Hosseini, and Youssef Marzouk. Conditional Sampling With Monotone GANs. *arXiv*, stat.ML, 06 2020.

[8] Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. Sampling via measure transport: An introduction. *Handbook of uncertainty quantification*, 1:2, 2016.

[9] Matthew D. Parno and Youssef M. Marzouk. Transport map accelerated markov chain monte carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018.

[10] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[11] Alessio Spantini, Daniele Bigoni, and Youssef Marzouk. Inference via low-dimensional couplings. *Journal of Machine Learning Research 19 (2018) 1-71*, 2018.