Sum-of-Squares Polynomial Flow

Priyank Jaini [1 2 3] **Kira A. Selby** [1 2] **Yaoliang Yu** [1 2 3]

Abstract

Triangular map is a recent construct in probability theory that allows one to transform any source probability density function to any target density function. Based on triangular maps, we propose a general framework for high-dimensional density estimation, by specifying one-dimensional transformations (equivalently conditional densities) and appropriate conditioner networks. This framework (a) reveals the commonalities and differences of existing autoregressive and flow based methods, (b) allows a unified understanding of the limitations and representation power of these recent approaches and, (c) motivates us to uncover a new Sum-of-Squares (SOS) flow that is interpretable, universal, and easy to train. We perform several synthetic experiments on various density geometries to demonstrate the benefits (and shortcomings) of such transformations. SOS flows achieve competitive results in simulations and several real-world datasets.

1. Introduction

Neural density estimation methods are gaining popularity for the task of multivariate density estimation in machine learning (Kingma et al., 2016; Dinh et al., 2015; 2017; Papamakarios et al., 2017; Uria et al., 2016; Huang et al., 2018). These generative models provide a tractable way to evaluate the exact density, unlike generative adversarial nets (Goodfellow et al., 2014) or variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014). Popular methods for neural density estimation are *autoregressive models* (Neal, 1992; Bengio & Bengio, 1999; Larochelle & Murray, 2011; Uria et al., 2016) and *normalizing flows* (Rezende & Mohamed, 2015; Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013). These models aim to learn an invertible, bijective and increasing transformation $\mathbf{T}$ that pushes forward a (simple) source probability density (or measure, in general) to a target density such that computing the inverse $\mathbf{T}^{-1}$ and the Jacobian $|\mathbf{T}'|$ is *easy*.

In probability theory, it has been rigorously proven that increasing *triangular* maps (Bogachev et al., 2005) are universal, i.e. any source density can be transformed into a target density using an increasing triangular map. Indeed, the Knothe-Rosenblatt transformation (Ch.1, Villani, 2008) gives a (heuristic) construction of such a map, which is unique up to null sets (Bogachev et al., 2005). Furthermore, by definition the inverse and the Jacobian of a triangular map can be very efficiently computed through univariate operations. However, for multivariate densities computing the exact Knothe-Rosenblatt transform itself is not possible in practice. Thus, a natural question is: Given a pair of densities, how can we efficiently estimate this unique increasing triangular map?

This work is devoted to studying these increasing, bijective, and monotonic triangular maps, in particular how to estimate them in practice. In §2, we precisely formulate the density estimation problem and propose a general maximum likelihood framework for estimating densities using triangular maps. We also explore the properties of the triangular map required to push a source density to a target density.

Subsequently, in §A, we trace back the origins of the triangular map and connect it to many recent works on generative modelling. We relate our study of increasing, bijective, triangular maps to works on iterative Gaussianization (Chen & Gopinath, 2001; Laparra et al., 2011) and normalizing flows Tabak & Vanden-Eijnden (2010); Tabak & Turner (2013); Rezende & Mohamed (2015). We show that a triangular map can be decomposed into compositions of one dimensional transformations or equivalently univariate conditional densities, allowing us to demonstrate that all *autoregressive models* and *normalizing flows* are subsumed in our general density estimation framework. As a by-product, this framework also reveals that *autoregressive models* and *normalizing flows* are in fact equivalent. Using this unified framework, we study the commonalities and differences of the various aforementioned models. Most importantly, this framework allows us to study the universality in a much cleaner and more streamlined way. We present a unified understanding of the limitations and representation power of these approaches, summarized concisely in Table 1 below.

In §3, by understanding the pivotal properties of triangular maps and using our proposed framework, we uncover a new neural density estimation procedure called the Sum-of-Squares polynomial flows (SOS flows). We show that SOS flows are akin to higher order approximation of $\mathbf{T}$ depending

on the degree of the polynomials used. Subsequently, we show that SOS flows are universal, i.e. given enough model complexity, they can approximate any target density. We further show that (a) SOS flows are a strict generalization of the inverse autoregressive flow (IAF) of Kingma et al. (2016), (b) they are interpretable; its coefficients directly control the higher order moments of the target density and, (c) SOS flows are easy to train; unlike NAFs (Huang et al., 2018) which require non-negative weights, there are no constraints on the parameters of SOS.

In §C, we report our empirical analysis. We performed holistic synthetic experiments to gain intuitive understanding of triangular maps and SOS flows in particular. Additionally, we compare SOS flows to previous neural density estimation methods on real-world datasets where it achieved competitive performance.

We summarize our main contributions as follows:

- We study and propose a rigorous framework for using triangular maps for density estimation
- Using this framework, we study the similarities and differences of existing flow based and autoregressive models
- We provide a unified understanding of the limitations and representational power of these methods
- We propose SOS flows that are universal, interpretable, and easy to train.
- We perform several synthetic and real-world experiments to demonstrate the efficacy of SOS flows.

2. Density estimation through triangular map

In this section we set up our main problem, introduce key definitions and notations, and formulate the general approach to estimate density functions using triangular maps.

Let p, q be two probability density[1] functions (w.r.t. the Lebesgue measure) over the source domain $\mathsf{Z} \subseteq \mathbb{R}^d$ and the target domain $\mathsf{X} \subseteq \mathbb{R}^d$, respectively. Our main goal is to find a *deterministic* transformation $\mathbf{T} : \mathsf{Z} \to \mathsf{X}$ such that for all (measurable) set $B \subseteq \mathsf{X}$,

$$\int_B q(\mathbf{x})\mathrm{d}\mathbf{x} \approx \int_{\mathbf{T}^{-1}(B)} p(\mathbf{z})\mathrm{d}\mathbf{z}.$$

In particular, when $\mathbf{T}$ is bijective and differentiable (e.g. Rudin, 1987), we have the change-of-variable formula $\mathbf{x} = \mathbf{T}(\mathbf{z})$ such that

$$\begin{aligned} q(\mathbf{x}) &= p(\mathbf{z})/|\mathbf{T}'(\mathbf{z})| \\ &= p(\mathbf{T}^{-1}\mathbf{x})/|\mathbf{T}'(\mathbf{T}^{-1}\mathbf{x})| =: \mathbf{T}_{\#}p, \end{aligned} \tag{1}$$

where $|\mathbf{T}'(\mathbf{z})|$ is the (absolute value) of the Jacobian (determinant of the derivative) of $\mathbf{T}$. In other words, by pushing the source random variable $\mathbf{z} \sim p$ through the map $\mathbf{T}$ we can obtain a new random variable $\mathbf{x} \sim q$. This "push-forward" idea has played an important role in optimal transport theory (Villani, 2008) and in Monte carlo simulations.

Here, our interest is to learn the target density q through the map $\mathbf{T}$. Let $\mathcal{F}$ be a class of mappings and use the KL divergence[2] to measure closeness between densities. We can formulate the density estimation problem as:

$$\min_{\mathbf{T}\in\mathcal{F}} \mathsf{KL}(q\|\mathbf{T}_{\#}p) \equiv -\int q(\mathbf{x}) \log \frac{p(\mathbf{T}^{-1}\mathbf{x})}{|\mathbf{T}'(\mathbf{T}^{-1}\mathbf{x})|}\mathrm{d}\mathbf{x}. \tag{2}$$

When we only have access to an i.i.d. sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \sim q$, we can replace the integral above with empirical averages, which amounts to maximum likelihood estimation:

$$\max_{\mathbf{T}\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \Big[-\log|\mathbf{T}'(\mathbf{T}^{-1}\mathbf{x}_i)| + \log p(\mathbf{T}^{-1}\mathbf{x}_i)\Big]. \tag{3}$$

Conveniently, we can choose *any* source density p to facilitate estimation. Typical choices include the standard normal density on $\mathsf{Z} = \mathbb{R}^d$ (with zero mean and identity covariance) and uniform density over the cube $\mathsf{Z} = [0, 1]^d$.

Computationally, being able to solve (3) efficiently relies on choosing a map $\mathbf{T}$ whose

- inverse $\mathbf{T}^{-1}$ is "cheap" to compute;
- Jacobian $|\mathbf{T}'|$ is "cheap" to compute.

Fortunately, this is always possible. Following Bogachev et al. (2005) we call a (vector-valued) mapping $\mathbf{T} : \mathbb{R}^d \to \mathbb{R}^d$ triangular if its j-th component T_j only depends on the first j variables $x_1, \ldots, x_j$. The name "triangular" is derived from the fact that the derivative of $\mathbf{T}$ is a triangular matrix function[3]. We call $\mathbf{T}$ increasing if for all $j \in [d]$, T_j is increasing w.r.t. the j-th variable, or equivalently, if $\mathbf{T}$ is increasing w.r.t. the element-wise partial order.

Theorem 1 (Bogachev et al. 2005)**.** *For any two densities p and q over $\mathsf{Z} = \mathsf{X} = \mathbb{R}^d$, there exists a unique (up to null sets of p)* increasing *triangular map $\mathbf{T} : \mathsf{Z} \to \mathsf{X}$ so that $q = \mathbf{T}_{\#}p$. The same[4] holds over $\mathsf{Z} = \mathsf{X} = [0, 1]^d$.*

Conveniently, to compute the Jacobian of an increasing triangular map we need only multiply d *univariate* partial derivatives $|\mathbf{T}'(\mathbf{x})| = \prod_{j=1}^{d} \frac{\partial T_j}{\partial x_j}$. Similarly, inverting an

[1] All of our results can be extended to two probability measures satisfying mild regularity conditions. For simplicity and concreteness we restrict to probability densities here.

[2] Other statistical divergences can be used as well.

[3] The converse is clearly also true if our domain is connected.

[4] More generally on any open or closed subset of $\mathbb{R}^d$ if we interpret the monotonicity of $\mathbf{T}$ appropriately (Alexandrova, 2006).

increasing triangular map requires inverting d *univariate* functions sequentially, each of which can be efficiently done through say bisection. Bogachev et al. (2005) further proved that the change-of-variable formula (1) holds for any increasing triangular map $\mathbf{T}$ (without any additional assumption but using the right-side derivative).

Thus, triangular mappings form a very appealing function class for us to learn a target density as formulated in (2) and (3). Indeed, many recent generative models in machine learning are precisely special cases of this approach as we discuss in detail in Appendix A. Before proceeding further, let us give some examples to help understand Theorem 1.

Example 1. *Consider two probability densities p and q on the real line $\mathbb{R}$, with distribution function F and G, respectively. Then, we can define the increasing map $T = G^{-1} \circ F$ such that $q = \mathbf{T}_{\#}p$, where $G^{-1} : [0,1] \to \mathbb{R}$ is the quantile function of q:*

$$G^{-1}(u) := \inf\{t : G(t) \geq u\}.$$

Indeed, it is well-known that $F(Z) \sim$ uniform if $Z \sim p$ and $G^{-1}(U) \sim q$ if $U \sim$ uniform. Theorem 1 is a rigorous iteration of this univariate argument by repeatedly conditioning. Note that the increasing property is essential for claiming the uniqueness of $\mathbf{T}$. Indeed, for instance, let p be standard normal, then both $\mathbf{T} = id$ and $\mathbf{T} = -id$ push p to the same target normal density.

Example 2 (Pushing uniform to normal)**.** *Let p be uniform over $[0,1]$ and $q \sim \mathcal{N}(\mu, \sigma^2)$ be normal distributed. The unique increasing transformation*

$$\begin{aligned} T(z) = G^{-1} \circ F &= \mu + \sqrt{2}\sigma \cdot \operatorname{erf}^{-1}(2z-1) \\ &= \mu + \sqrt{2}\sigma \cdot \sum_{k=0}^{\infty} \frac{\pi^{k+1/2} c_k}{2k+1} (z - \tfrac{1}{2})^{2k+1}, \end{aligned}$$

where $\operatorname{erf}(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-s^2} \mathrm{d}s$ is the error function, which was Taylor expanded in the last equality. The coefficients $c_0 = 1$ and $c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}$. We observe that the derivative of T is an infinite sum of squares of polynomials. In particular, if we truncate at $k = 0$, we obtain

$$T(z) = \mu + \sqrt{2\pi}\sigma(z - \tfrac{1}{2}) + O(z^3). \tag{4}$$

Example 3 (Pushing normal to uniform)**.** *Similar as above but we now find a map S that pushes q to p:*

$$\begin{aligned} S(x) = F^{-1} \circ G &= \Phi(\tfrac{x-\mu}{\sigma}) \\ &= \frac{1}{2} + \frac{1}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k}{k!(2k+1)} \left(\frac{x-\mu}{\sqrt{2}\sigma}\right)^{2k+1}, \end{aligned}$$

where Φ is the cdf of standard normal. As shown by Medvedev (2008), S must be the inverse of the map T in Example 2. We observe that the derivative of S is no longer a sum of squares of polynomials, but we prove later that it is approximately so. If we truncate at $k = 0$, we obtain

$$S(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}\sigma}(x - \mu) + O(x^3),$$

where the leading term is also the inverse of the leading term of T in (4).

We end this section with two important remarks.

Remark 1. *If the target density has disjoint support e.g. mixture of Gaussians (MoGs) with well-separated components, then the resulting transformation will need to admit sharp jumps for areas of near zero mass. This follows by analyzing the transformaiton $T(z) = G^{-1} \circ F$. The slope $T'(z)$ of $T(z)$ is the ratio of the quantile pdfs of the source density and the target density. Therefore, in regions of near zero mass for target density, the transformation will have near infinite slope. In Appendix B, we demonstrate this phenomena specifically for well-separated MoGs and show that a piece-wise linear function transforms a standard Gaussian to MoGs. This also opens the possibility to use the number of jumps of an estimated transformation as the indication of the number of components in the data density.*

Remark 2. *So far we have employed the (increasing) triangular map* $\mathbf{T}$ explicitly *to represent our estimate of the target density. This is advantageous since it allows us to easily draw samples from the estimated density, and, if needed, it results in the estimated density formula* (1) *immediately. An alternative would be to parameterize the estimated density directly and explicitly, such as in mixture models, probabilistic graphic models and sigmoid belief networks. The two approaches are conceptually* equivalent*: Thanks to Theorem 1, we know choosing a family of triangular maps fixes a family of target densities that we can represent, and conversely, choosing a family of target densities fixes a family of triangular maps that we can* implicitly *learn. The advantage of the former approach is that given a sample from the target density, we can infer the "pre-image" in the source domain while this information is lost in the second approach.*

3. Sum-of-Squares Polynomial Flow

In Section 2 we developed a general framework for density estimation using triangular maps, and in Appendix A we showed the many recent generative models are all trying to estimate a triangular map in one way or another. In this section we give a surprisingly simple way to parameterize triangular maps, which, when plugged into (3), leads to a new density estimation algorithm that we call sum-of-squares (SOS) polynomial flow.

Our approach is motivated by some classical result on simulating univariate non-normal distributions. Let z be univariate standard normal. Fleishman (1978) proposed to simulate

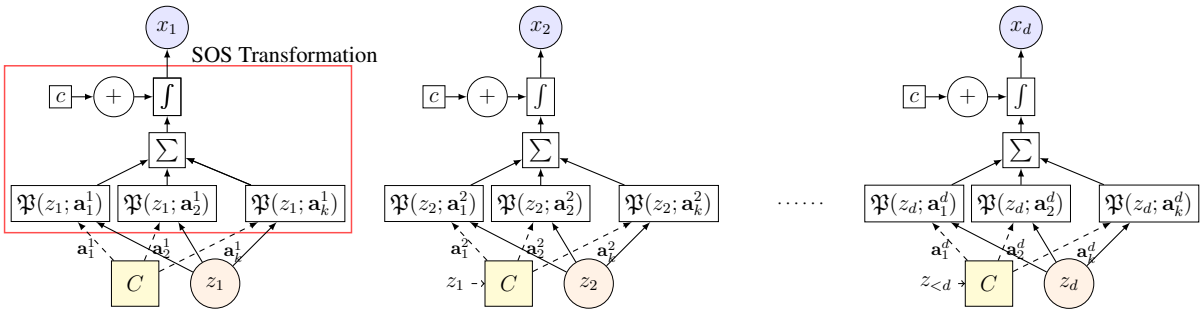

Figure 1. Schematic of SOS flows depicting the conditioner network and relevant transformations.

a non-normal distribution by fitting a degree-3 polynomial:

$$x = \mathfrak{P}_3(z;\mathbf{a}) = a_0 + a_1 z + a_2 z^2 + a_3 z^3,$$

where the coefficients $\{a_l\}$ are estimated by matching the first 4 moments of x with those of empirical data. This approach was quite popular in practice because it allows researchers to precisely control the moments (such as skewness and kurtosis). However, three difficulties remain: (1) with degree-3 polynomial one can only (approximately) simulate a (very) strict subset of non-normal distributions. This can be addressed by using polynomials of higher degrees and better quantile matching techniques (Headrick, 2009). (2) The estimated coefficients $\{a_l\}$ may not guarantee the monotonicity of the polynomial, making inversion and density evaluation difficult, if not impossible. (3) Extension to the multivariate case was done through composing a linear map (Vale & Maurelli, 1983), which can be quite inefficient.

We show that all three difficulties can be overcome using SOS flows. First, let us recall a classic result in algebra:

Theorem 2. *A univariate real polynomial is increasing iff it can be written as:*

$$\mathfrak{P}_{2r+1}(z;\mathbf{a}) = c + \int_0^z \sum_{\kappa=1}^{k} \left(\sum_{l=0}^{r} a_{l,\kappa} u^l \right)^2 \mathrm{d}u, \qquad (5)$$

where $c \in \mathbb{R}$, $r \in \mathbb{N}$, and k can be chosen as small as 2.

Note that a univariate increasing polynomial is strictly increasing iff it is not a constant. Theorem 2 is obtained by integrating a nonnegative polynomial, which is necessarily a sum-of-squares, see e.g. (Marshall, 2008). Now, by applying (5) to model each conditional density in (8) we effectively addressed the last two issues above. Pleasantly, this approach strictly generalizes the affine triangular map (9) of (Kingma et al., 2016), which amounts to truncating $r = 0$ in (5). However, by using a larger r, we can learn certain densities more faithfully (especially for capturing higher order statistics), without significantly increasing the computational complexity. Additionally, implementing (5) in practice is simple: It can be computed *exactly* since it is an integral of univariate polynomials.

Lastly, we prove that as the degree r increases, we can approximate any triangular map. We prove our result for the domain $\mathsf{Z} = \mathsf{X} = \mathbb{R}^d$, but the same result holds for other domains if we slightly modify the proof in the appendix.

Theorem 3. *Let C be the space of real univariate continuous functions, equipped with the topology of compact convergence. Then, the set of increasing polynomials is dense in the cone of increasing continuous functions.*[5]

Since the topology of pointwise convergence is weaker than that of compact convergence (e.g. uniform convergence on every compact set), we immediately know that there exists a sequence of increasing polynomials of the form (5) that converges pointwise to any given continuous function. This universal property of increasing polynomials allows us to prove the universality of SOS flows, e.g. the capability of approximating any (continuous) triangular map.

SOS flow consists of two parts: an increasing (univariate) polynomial $\mathfrak{P}_{2r+1}(z_j;\mathbf{a}_j)$ of the form (5) for modelling conditional densities and a conditioner network $C_j(z_1,\ldots,z_{j-1})$ for generating the coefficients $\mathbf{a}_j$ of the polynomial $\mathfrak{P}_{2r+1}(z_j;\mathbf{a}_j)$. In other words, the triangular map learned using SOS flows has the following form:

$$\forall j,\ \ T_j(z_1,\ldots,z_j) = \mathfrak{P}_{2r+1}\big(z_j; C_j(z_1,\ldots,z_{j-1})\big). \quad (6)$$

If we choose a universal conditioner (that can approximate any continuous function), such as a neural net, then combining with Theorem 2 and Theorem 3 we verify that the triangular maps in the form of (6) can approximate any increasing continuous triangular map in the pointwise manner. It then follows that the transformed densities will converge weakly to any desired target density (i.e. in distribution). This solves the first issue mentioned before Theorem 2. We remark that our universality proof for SOS flows is significantly shorter and more streamlined than the previous

[5]This result does not seem to be surprising, given that it is well-known polynomials are dense in C. Note however that it does *not* follow immediately. For instance, it took significant effort for Huang et al. (2018) to prove a similar result, which is arguably one of their main contributions.

attempt of Huang et al. (2018), and it can be seemingly extended to analyze other models summarized in Table 1.

As pointed out by Papamakarios et al. (2017) we can also construct conditioner networks C_j that take inputs $x_1, \ldots, x_{j-1}$, instead of $z_1, \ldots, z_{j-1}$. They are equivalent in theory but one can be more convenient than the other, depending on the downstream application. Figure 1 illustrates the main components of a single-block SOS flow, where we implement the conditioner network in the same way as in (Papamakarios et al., 2017). To get a higher degree approximation, we can either increase r or stack a few single-block SOS flows, as shown in Figure 7 (in appendix). The former approach appears to be more general but also more difficult to train due to the larger number of parameters. Indeed, the effective number of parameters for SOS flows obtained by stacking L blocks with k polynomials of degree $2r+1$ is $L \cdot k \cdot (r+1)$ whereas achieving the same representation with a single block wide SOS flow would require $1/2 \cdot k \cdot ((2r+1)^L - 1)$ parameters. In Appendix C.1 we perform simulated experiments to compare deep vs. wide SOS flows.

SOS flow is similar to the neural autoregressive flow (NAF) of Huang et al. (2018) in the sense that both are capable of approximating any (continuous) triangular map hence learning any desired target density. However, SOS flow has the following advantages:

- As mentioned before, SOS flow is a strict generalization of the inverse autoregressive flow (IAF) of Kingma et al. (2016), which corresponds to setting $r = 0$.
- SOS flow is more interpretable, in the sense that its parameters (i.e. coefficients of the polynomials) directly control the first few moments of the target density.
- SOS flow may be easier to train, as there is no constraint on its parameters $\mathbf{a}$. In contrast, NAF needs to make sure the parameters are nonnegative[6].

We evaluated the performance of SOS flows on both synthetic and real-world datasets for density estimation, and compare it to several alternative autoregressive models and flow based methods. Due to space constraints we defer our empirical analysis to Appendix C.

4. Conclusion

We have presented a unified framework for estimating complex densities using monotone and bijective triangular maps. The main idea is to specify one-dimensional transformations and then iteratively extend to higher-dimensions using conditioner networks. Under this framework, we analyzed popular autoregressive and flow based methods and revealed their similarities and differences, and we provided a unified and streamlined approach for understanding the representation power of these methods. Along the way we uncovered a new sum-of-squares polynomial flow that we show is universal, interpretable and easy to train. We discussed the various advantages of SOS flows for stochastic simulation and density estimation, and we performed various experiments on simulated data to explore the properties of SOS flows. Lastly, SOS flows achieved competitive results on real-world datasets. In the future we plan to carry out the analysis indicated in Table 1, and to formally establish the respective advantages between deep and wide SOS flows.

[6] A typical remedy is to re-parameterize through an exponential transform. However, numerically this may easily lead to overflows or underflows.

Acknowledgement

We thank the reviewers for their insightful comments, and Csaba Szepesvári for discussions. We gratefully acknowledge support from NSERC. PJ was also supported by scholarships from Borealis AI and Huawei.

References

Alexandrova, D. Convergence of Triangular Transformations of Measures. *Theory of Probability & Its Applications*, 50(1):113–118, 2006.

Ballé, J., Laparra, V., and Simoncelli, E. P. Density modeling of images using a generalized normalization transformation. In *ICLR*, 2016.

Bengio, Y. and Bengio, S. Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks. In *NeurIPS*, 1999.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Bogachev, V. I., Kolesnikov, A. V., and Medvedev, K. V. Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309–335, 2005.

Carlier, G., Galichon, A., and Santambrogio, F. From Knothe's Transport to Brenier's Map and a Continuation Method for Optimal Transport. *SIAM Journal on Mathematical Analysis*, 41(6):2554–2576, 2010.

Chen, S. S. and Gopinath, R. A. Gaussianization. In *NeurIPS*, pp. 423–429, 2001.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6572–6583, 2018.

Deco, G. and Brauer, W. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. In *ICLR workshop*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *ICLR*, 2017.

Fleishman, A. I. A method for simulating non-normal distributions. *Psychometrika*, 43(4):521–532, 1978.

Friedman, J. H., Stuetzle, W., and Schroeder, A. Projection Pursuit Density Estimation. *Journal of the American Statistical Association*, 79(387):599–608, 1984.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked autoencoder for distribution estimation. In *ICML*, pp. 881–889, 2015.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, pp. 2672–2680, 2014.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Headrick, T. C. *Statistical Simulation Power Method Polynomials and Other Transformations*. CRC Press, 2009.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural Autoregressive Flows. In *ICML*, 2018.

Huber, P. J. Projection Pursuit. *The Annals of Statistics*, 13 (2):435–475, 1985.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *NeurIPS*, pp. 4743–4751, 2016.

Knothe, H. Contributions to the theory of convex bodies. *The Michigan Mathematical Journal*, 4(1):39–52, 1957.

Laparra, V., Camps-Valls, G., and Malo, J. Iterative Gaussianization: From ICA to Random Rotations. *IEEE Transactions on Neural Networks*, 22(4):537–549, 2011.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *AIStats*, pp. 29–37, 2011.

MacKay, M., Vicol, P., Ba, J., and Grosse, R. B. Reversible Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 9043–9054, 2018.

Marshall, M. *Positive Polynomials and Sums of Squares*. AMS, 2008.

McLachlan, G. and Peel, D. *Finite mixture models*. John Wiley & Sons, 2004.

Medvedev, K. V. Certain properties of triangular transformations of measures. *Theory of Stochastic Processes*, 14 (1):95–99, 2008.

Mulansky, B. and Neamtu, M. Interpolation and Approximation from Convex Sets. *Journal of Approximation Theory*, 92(1):82–100, 1998.

Neal, R. M. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, 1992.

Oliva, J., Dubey, A., Zaheer, M., Poczos, B., Salakhutdinov, R., Xing, E., and Schneider, J. Transformation Autoregressive Networks. In *ICML*, pp. 3898–3907, 2018.

Oord, A. V., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. In *ICML*, pp. 1747–1756, 2016.

Ostrovski, G., Dabney, W., and Munos, R. Autoregressive Quantile Networks for Generative Modeling. In *ICML*, pp. 3936–3945, 2018.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *NeurIPS*, pp. 2338–2347, 2017.

Redlich, A. N. Supervised Factorial Learning. *Neural Computation*, 5(5):750–766, 1993.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *ICML*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Rosenblatt, M. Remarks on a Multivariate Transformation. *The Annals of Mathematical Statistics*, 23(3):470–472, 1952.

Rudin, W. *Real and Complex Analysis*. McGraw-Hill, 3rd edition, 1987.

Tabak, E. G. and Turner, C. V. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Tabak, E. G. and Vanden-Eijnden, E. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Talagrand, M. Transportation cost for Gaussian and other product measures. *Geometric & Functional Analysis GAFA*, 6(3):587–600, 1996.

Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17 (1):7184–7220, 2016.

Vale, C. D. and Maurelli, V. A. Simulating multivariate nonnormal distributions. *Psychometrika*, 48(3):465–471, 1983.

Villani, C. *Optimal Transport: Old and New*, volume 338. Springer, 2008.

Wenliang, L., Sutherland, D., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. *arXiv preprint arXiv:1811.08357*, 2018.

A. Connection to existing works

The results in Section 2 suggest using (3) with $\mathcal{F}$ being a class of triangular maps for estimating a probability density q. In this section we put this general approach into historical perspective, and connect it to the many recent works on generative modelling. Due to space constraint, we limit our discussion to work that are directly relevant to ours.

Origins of triangular map: Rosenblatt (1952), among his contemporary peers, used the triangular map to transform a continuous multivariate distribution into the uniform distribution over the cube. Independently, Knothe (1957) devised the triangular map to transform uniform distributions over convex bodies and to prove generalizations of the Brunn-Minkowski inequality. Talagrand (1996), unaware of the previous two results and in the process of proving some sharp Gaussian concentration inequality, effectively discovered the triangular map that transforms the Gaussian distribution into any continuous distribution. The work of Bogachev et al. (2005) rigorously established the existence and uniqueness of the triangular map and systematically studied some of its key properties. Carlier et al. (2010) showed surprisingly that the triangular map is the limit of solutions to a class of Monge-Kantorovich mass transportation problems under quadratic costs with diminishing weights. None of these pioneering works considered using triangular maps for density estimation.

Iterative Gaussianization and Normalizing Flow: In his seminal work, Huber (1985) developed the important notion of non-Gaussianality to explain the projection pursuit algorithm of Friedman et al. (1984). Later, Chen & Gopinath (2001), based on a heuristic argument, discovered the triangular map approach for density estimation but deemed it impractical because of the seemingly impossible task of estimating too many conditional densities. Instead, Chen & Gopinath (2001) proposed the iterative Gaussianization technique, which essentially decomposes[7] the triangular map into the composition of a sequence of alternating diagonal maps $\mathbf{D}_t$ and linear maps $\mathbf{L}_t$. The diagonal maps are estimated using the univariate transform in Example 1 where G is standard normal and F is a mixture of standard normals. Later, Laparra et al. (2011) simplified the linear map into random rotations. Both approaches, however, suffer cubic complexity w.r.t. dimension due to generating or evaluating the linear map. The recent work of (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013) coined the name *normalizing flow* and further exploited the straightforward but crucial observation that we can approximate the triangular map through a sequence of "simple" maps such as radial basis functions or rotations composed with diagonal maps. Similar simple maps have also been explored in Ballé et al. (2016). Rezende & Mohamed (2015) designed a "rank-1" (or radial) normalizing flow and applied it to variational inference, largely popularizing the idea in generative modelling. All these approaches are not estimating a triangular map *per se*, but the main ideas are nevertheless similar.

(*Bona fide*) **Triangular Approach**: Deco & Brauer (1995) (see also Redlich (1993)), to our best knowledge, is among the first to mention the name "triangular" *explicitly* in tasks (nonlinear independent component analysis) related to density estimation. More recently, Dinh et al. (2015) recognized the promise of even simple triangular maps in density estimation. The (increasing) triangular map in (Dinh et al., 2015) consists of two simple (block) components: $T_1(\mathbf{x}_1) = \mathbf{x}_1$ and $T_2(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_2 + m(\mathbf{x}_1)$, where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is a two-block partition and m is a map parameterized by a neural net. The advantage of this triangular map is its computational convenience: its Jacobian is trivially 1 and its inversion only requires evaluating m. Dinh et al. (2015) applied different partitions of variables, iteratively composed several such simple triangular maps and combined with a diagonal linear map[8]. However, these triangular maps appear to be too simple and it is not clear if through composition they can approximate any increasing triangular map. In subsequent work, Dinh et al. (2017) proposed the extension where $T_1(\mathbf{x}_1) = \mathbf{x}_1$ but $T_2(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_2 \odot \exp(s(\mathbf{x}_1)) + m(\mathbf{x}_1)$, where $\odot$ denotes the element-wise product. This map is again increasing triangular.

Autoregressive Neural Models: A joint probability density function can be factorized into the product of marginal and conditionals:

$$q(x_1, \ldots, x_d) = q(x_1) \prod_{j=2}^{d} q(x_j | x_{j-1}, \ldots, x_1).$$

In his seminal work, Neal (1992) proposed to model each (discrete) conditional density by a simple linear logistic function (with the conditioned variables as inputs). This was later extended by Bengio & Bengio (1999) using a two-layer nonlinear neural net. The recent work of Uria et al. (2016) proposed to decouple the hidden layers in Bengio & Bengio (1999) and to introduce heavy weight sharing to reduce overfitting and computational complexity. Already in (Bengio & Bengio, 1999), univariate mixture models were mentioned as a possibility to model each conditional density, which was further substantiated in (Uria et al., 2016). More

[7] This can be made precise, much in the same way as decomposing a triangular matrix into the product of two rotation matrices and a diagonal matrix, i.e. the so-called Schur decomposition.

[8] They also considered a more general coupling that may no longer be triangular.

Table 1. Various auto-regressive and flow-based methods expressed under a unified framework. All the conditioners can take inputs $\mathbf{x}$ instead of $\mathbf{z}$. The symbol 🔗 is used for weight sharing, 👻 for use of masks for efficient implementation, 🏛 for universality of the method and, Δ if the method learns a triangular transformation explicitly (E) or implicitly (I). ? implies that universality of these methods has neither been proved or disproved although it can now be analyzed with ease using our framework. $S_j(z_j; \boldsymbol{\theta}_j)$ is defined in eq. (8) and $\mathfrak{P}_{2r+1}(z_j; \mathbf{a}_j)$ is defined in eq. (5).

Model	conditioner C_j output	$T_j\big(z_j\,; C_j(z_1, \ldots, z_{j-1})\big)$	🔗	👻	🏛	Δ
Mixture (e.g. McLachlan & Peel, 2004)	$\boldsymbol{\theta}_j$	$S_j(z_j; \boldsymbol{\theta}_j)$	✗	✗	✓	I
(Bengio & Bengio, 1999)	$\boldsymbol{\theta}_j(z_{<j})$	$S_j(z_j; \boldsymbol{\theta}_j)$	✗	✗	?	I
MADE (Germain et al., 2015)	$\boldsymbol{\theta}_j(z_{<j})$	$S_j(z_j; \boldsymbol{\theta}_j)$	✓	✓	?	I
NICE (Dinh et al., 2015)	$\mu_j(z_{<l})$	$z_j + \mu_j \cdot \mathbb{1}_{j \notin [l]}$	✗	✗	?	E
NADE (Uria et al., 2016)	$\boldsymbol{\theta}_j(z_{<j})$	$S_j(z_j; \boldsymbol{\theta}_j)$	✓	✗	?	I
IAF (Kingma et al., 2016)	$\sigma_j(z_{<j}),\ \mu_j(z_{<j})$	$\sigma_j z_j + (1-\sigma_j)\mu_j$	✓	✓	?	E
MAF (Papamakarios et al., 2017)	$\alpha_j(z_{<j}),\ \mu_j(z_{<j})$	$z_j \exp(\alpha_j) + \mu_j$	✓	✓	?	E
Real-NVP (Dinh et al., 2017)	$\alpha_j(z_{<l}), \mu_j(z_{<l})$	$\exp(\alpha_j \cdot \mathbb{1}_{j \notin [l]}) \cdot z_j + \mu_j \cdot \mathbb{1}_{j \notin [l]}$	✗	✗	?	E
NAF (Huang et al., 2018)	$\mathbf{w}_j(z_{<j})$	$\mathrm{DNN}(z_j\,; \mathbf{w}_j)$	✓	✓	✓	E
SOS	$\mathbf{a}_j(z_{<j})$	$\mathfrak{P}_{2r+1}(z_j; \mathbf{a}_j)$	✓	✓	✓	E

precisely, they model the j-th conditional density as:

$$q(x_j|x_{j-1}, \ldots, x_1) = \sum_{\kappa=1}^{k} w_{j,\kappa}\, \mathcal{N}(x_j; \mu_{j,\kappa}, \sigma_{j,\kappa}) \qquad (7)$$
$$\boldsymbol{\theta}_j := (w_{j,\kappa}, \mu_{j,\kappa}, \sigma_{j,\kappa})_{\kappa=1}^{k} = C_j(x_{j-1}, \ldots, x_1),$$

where C_j is the so-called conditioner network that outputs the parameters for the (univariate) mixture distribution in (7). According to Example 1 there exists a unique increasing map $S_j(\cdot\,; \boldsymbol{\theta}_j)$ that maps a univariate standard normal random variable z_j into x_j that follows (7). In other words,

$$x_j = S_j(z_j; \boldsymbol{\theta}_j) =: T_j(z_1, \ldots, z_{j-1}, z_j), \qquad (8)$$

where the last equality follows from induction, using the fact that $\boldsymbol{\theta}_j = C_j(x_{j-1}, \ldots, x_1)$. Thus, as already pointed out in Remark 2, specifying a family of conditional densities as in (7) is **equivalent** as (implicitly) specifying a family of triangular maps. In particular, if we use a nonparametric family such as mixture of normals, then the induced triangular maps can approximate any increasing triangular map. The special case, when $k = 1$ in (7), was essentially dealt with by Kingma et al. (2016): for $k = 1$ the map $S_j(\boldsymbol{\theta}_j) = \mu_j + \sigma_j z_j$ hence the triangular map

$$T_j(z_1, \ldots, z_{j-1}, z_j) = \mu_j(z_{<j}) + \sigma_j(z_{<j}) \cdot z_j. \qquad (9)$$

Obviously, not every triangular map can be written in the form (9), which is affine in z_j when $z_{<j}$ are fixed. To address this issue, Kingma et al. (2016) composed several triangular maps in the form of (9), hoping this suffices to approximate a generic triangular map. In contrast, Huang et al. (2018) proposed to replace the affine form in (9) with a univariate neural net (with z_j as input and μ_j and σ_j serve as weights). Lastly, based on binary masks, Germain et al. (2015) and Papamakarios et al. (2017) proposed efficient implementations of the above that compute all parameters in a single pass of the conditioner network. It should be clear now that (a) autoregressive models implement exactly a triangular map; (b) specifying the conditional densities directly is equivalent as specifying a triangular map explicitly.

Other Variants. Recurrent nets have also been used in autoregressive models (effectively triangular maps). For instance, Oord et al. (2016) used LSTMs to directly specify the conditional densities while MacKay et al. (2018) chose to explicitly specify the triangular maps. The two approaches, as alluded above, are equivalent, although one may be more efficient in certain applications than the other. Oliva et al. (2018) tried to combine both while Kingma & Dhariwal (2018) used an invertible 1×1 convolution. We note that the work of Ostrovski et al. (2018) models the conditional quantile function, which is equivalent to but can sometimes be more convenient than the conditional density.

Non-Triangular Flows. Sylvester Normalizing Flows (SNF) (Berg et al., 2018) and FFJORD (Grathwohl et al., 2018) are examples of normalizing flows that employ non-triangular maps. They both propose efficient methods to compute the Jacobian for change of variables. SNF utilizes Sylvester's Determinant Theorem to generalize planar flows and efficient computation of the Jacobian. FFJORD, on the other hand, defines a generative model based on continuous-time normalising flows proposed by Chen et al. (2018) that

map latent variables to target data distribution using ODEs. This formulation requires the Jacobian trace to evaluate the log-density which is done efficiently using Hutchinson's trace estimator.

B. Transformation for Mixture of Gaussians

The slope $T'(z)$ of T at any point z is given by

$$T'(z) = \frac{p(z)}{q\Big(T(z)\Big)}, \quad \text{where} \quad x = T(z)$$
$$= \frac{p\Big(F^{-1}(t)\Big)}{q\Big(G^{-1}(t)\Big)}, \quad \text{where} \quad t = F(z)$$

i.e. the slope $T'(z)$ is the ratio of probability density quantiles (pdQs) of the source random variable and the target random variable.

We now analyze the transformation required to transform a standard normal distribution to mixture of normal distributions. Figure 2 shows three columns of plots: In the leftmost column, the top plot is the source distribution (Z) which is standard normal. The bottom plot is the target distribution for the random variable X which is a Gaussian mixture model with two components. The means are -10 and 10, the variance is 1 and weights are 0.5 for each component respectively. The middle plot shows the transformation T required to push forward a standard normal distribution to the target. In the second column of plots, we now transform a standard normal distribution to a mixture distribution but with means as -20 and 20, i.e. the components are more separated. Finally, in the plots given in the rightmost column, we transform a standard normal distribution to a mixture of three Gaussians with means -20, -5, and 15. The variances are 1 and weights are $\frac{1}{3}$ respectively.

We make the following observations here: In all three plots for the transformation, we notice that the transformation admits jumps (close to being vertical) i.e. the slope at these points is large and close to infinity. This is expected since the regions where the target has almost zero mass but the source has finite mass would lead to a slope with such behavior. In the plots, this is the region in between the components where the mass of the target density approaches zero. Furthermore, the larger this area, the longer is the height of this jump (see plots on column one and column two). With densities that have two such areas, the transformation as expected has two jumps (plots on column three). The slope of T on the extremes is a constant and is equal to the standard deviation of the component on that extreme. This is because:

$$\lim_{z \to \pm\infty} T'(z) = \lim_{z \to \pm\infty} \frac{p(z)}{q\Big(T(z)\Big)}$$

As $z \to \infty$, q is approximately equal to the component on the positive extreme of the x-axis. This easily gives that $\lim_{z\to\infty} T'(z) = \sigma_+$ where σ_+ is the standard deviation of the component on the positive extreme of the x-axis; similarly, we get $\lim_{z\to-\infty} T'(z) = \sigma_-$ i.e. $T'(z)$ is a constant in almost all the region of zero mass on the left of the component on the negative extreme and on the right of the positive extreme (verified in Figure 2). Finally, the only regions where $T'(z)$ is finite is whenever $q(x) > q(\tilde{x})$ where $\tilde{x} \leq \mu_i \pm 2\sigma_i$ where the index i stands for the i^{th} component. Therefore, any T that transforms a standard normal distribution to a mixture of Gaussians will be approximately piece-wise linear with jumps. The number of linear pieces in this transformation will be equal to the number of components in the mixture. The slopes of these linear pieces will be a function of the standard deviations of the mixture components. Additionally, the height of the jump will be a function of the mixing weights and standard deviation of the mixture components.

C. Experiments

C.1. Simulated Experiments

We performed a host of experiments on simulated data to gain in-depth understanding of SOS flows.

In Figure 3 we demonstrate the ability of SOS flows to represent transformations that lead to multi-modal densities by generating data from a mixture of Gaussians for two cases - well-connected and disjoint support. The true transformation can be computed exactly following Example 1. We show three transformations learned by SOS flows for each case corresponding to a deep SOS flow, wide SOS flow and wide-deep SOS flow. As is evident, SOS flows were fairly successful in learning the transformations. We further estimated the parameters of these simulated densities using Gaussian mixtures trained using maximum likelihood under three cases - exact (same number of components as target density), under-specified (lesser number of components) and over-specified. Subsequently, we plot the resulting transformation in each case following Example 1. While, GMMs with exact components work well as expected, the transformations learned by under-specified and over-specified models are not as good. This experiment also goes on to show that using a parameterized density to model conditionals is equivalent to implicitly learning a transformation. We also performed experiments to study the effect of relative ordering of variables for the conditioner network and the represenational power of deep and wide SOS flows. Due to space constraints we defer the figures and explanation to Appendix C.2.

Finally, we tested SOS flows on a suite of 2D simulated datasets – Funnel, Banana, Square, Mixture of Gaussians

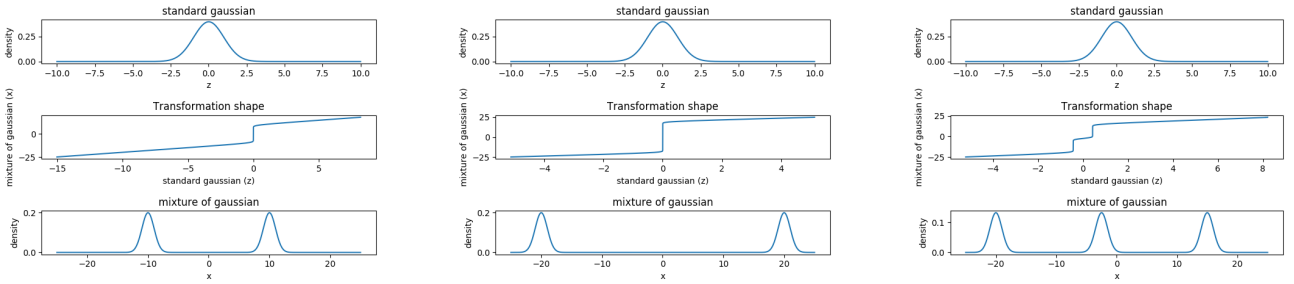

Figure 2. Transformation curves from standard Gaussian to mixture of Gaussians.

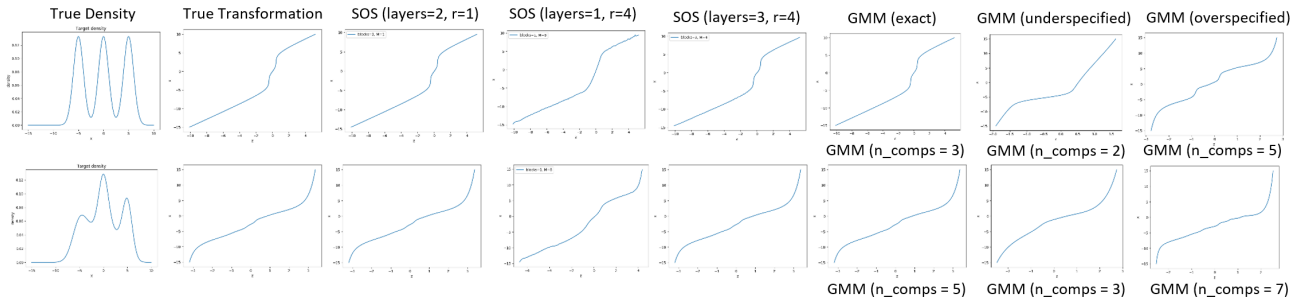

Figure 3. **Top Row:** First plot from the left shows the target density, a mixture of three component Gaussians with means = (-5, 0, 5), variances = (1, 1, 1) and, weights = (1/3, 1/3, 1/3). The second plot shows the exact transformation required to transform a standard Gaussian to this mixture. The next three plots shows the transformation learned by SOS flows with different configurations (deep, wide and wide-deep, respectively). The last three plots show the transformation learned by estimating the parameters of the Gaussian mixture using log-likelihood with exact (3), under-specified (2) and over-specified (5) number of components respectively. **Bottom Row:** Same as Top Row but with target density being a mixture of five Gaussians with means = (-5, -2, 0, 2, 5), variances = (1.5, 2, 1, 2, 1) and, weights = 0.2 each.

and Mixture of Rings. These datasets cover a broad range of geometries and have been considered before by Wenliang et al. (2018). For these experiments, we constructed our model by stacking 3 blocks with each block being a sum of *two* polynomials each of degree *four*. We plot the log density function learned by SOS flow and the true model in Figure 4. The model is able to capture the true log density of datasets like Funnel and Banana. The true densities of Funnel and Banana are a simple linear transformation of Gaussians. Hence, flow based models that learn a continuous and smooth transformation are expected to perform well on these datasets. However, SOS demonstrates certain artifacts at the sharp corners of the Square although it is able to capture the overall density nicely. These three datasets – Funnel, Banana, and Square – were part of the unimodal simulated datasets.

The multimodal datasets included Mixture of Gaussians (MoG) and Mixture of Rings (MoR). As discussed earlier in Remark 1, when the target distribution has regions of near zero mass, the learned transformation admits sharp jumps to capture such regions. Flow based models by virtue of being invertible and smooth are often unable to learn such sharp jumps. SOS flows performs reasonably well for mixture of Gaussians although there are certain artifacts in the model

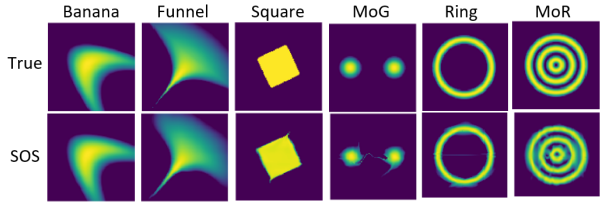

Figure 4. **Top Row:** True log-densities for various toy datasets. **Bottom Row:** Log-densities learned by SOS flows.

that try to connect the two components. Similarly, there are some artifacts connecting the rings for the Mixture of Rings datasets. However, this issue of separated components can be dealt with relative ease in practice using clustering. For datasets suspected to have disjoint components, one can run a simple clustering algorithm to identify the separated components. Each cluster can then be modeled individually with weights being equal to its sample count. However, this heuristic may not work well for high-dimensional datasets with sparse samples.

C.2. Effect of Degree

Next, we explore the effect of relative ordering for the conditioner network for SOS flows as well as mixture of

Table 2. Average test log-likelihoods and standard deviation for SOS flows over 10 trials (higher is better). The other methods report the average log-likelihood and standard deviation over five trials. The numbers in the parenthesis indicate the number of stacked blocks for the resultant transformation.

Method	Power	Gas	Hepmass	MiniBoone	BSDS300
MADE	0.40 $\pm$ 0.01	8.47 $\pm$ 0.02	-15.15 $\pm$ 0.02	-12.24 $\pm$ 0.47	153.71 $\pm$ 0.28
MAF affine (5)	0.14 $\pm$ 0.01	9.07 $\pm$ 0.02	-17.70 $\pm$ 0.02	-11.75 $\pm$ 0.44	155.69 $\pm$ 0.28
MAF affine (10)	0.24 $\pm$ 0.01	10.08 $\pm$ 0.02	-17.73 $\pm$ 0.02	-12.24 $\pm$ 0.45	154.93 $\pm$ 0.28
MAF MoG (5)	0.30 $\pm$ 0.01	9.59 $\pm$ 0.02	-17.39 $\pm$ 0.02	-11.68 $\pm$ 0.44	156.36 $\pm$ 0.28
TAN	0.48 $\pm$ 0.01	11.19 $\pm$ 0.02	-15.12 $\pm$ 0.02	-11.01 $\pm$ 0.48	157.03 $\pm$ 0.07
NAF DDSF (5)	0.62 $\pm$ 0.01	11.91 $\pm$ 0.13	-15.09 $\pm$ 0.40	-8.86 $\pm$ 0.15	157.73 $\pm$ 0.04
NAF DDSF (10)	0.60 $\pm$ 0.02	11.96 $\pm$ 0.33	-15.32 $\pm$ 0.23	-9.01 $\pm$ 0.01	157.43 $\pm$ 0.30
SOS (3)	0.68 $\pm$ 0.01	12.02 $\pm$ 0.41	-15.05 $\pm$ 0.10	-8.90 $\pm$ 0.11	156.43 $\pm$ 0.41

Table 3. Negative test log-likelihoods for various density estimation models on image datasets (lower is better). * results/models used multi-scale convolutional architectures.

Method	MNIST	CIFAR10
Real-NVP	1.06*	3.49*
Glow	1.05*	3.35*
FFJORD	0.99*	3.40*
MADE	2.04	5.67
MAF	1.89	4.31
SOS	1.81	4.18

Gaussians. We again generated two sets of 2D densities given by $p(x_1, x_2) = \mathcal{N}(x_2\,; 0, 4)\mathcal{N}(x_1\,; 0.25x_2^2, 1)$ and $p(x_1, x_2) = \mathcal{N}(x_2\,; 2, 2)\mathcal{N}(x_1\,; 1/3x_2^3, 1.5)$. However, we trained both SOS flows and GMMs with the reverse order i.e. x_1, x_2. For SOS flows we again tested using both deep and wide flows whereas for MoGs we tested with varying number of components for each conditional. We present the plots in Figure 5. The best performance here is by a deep SOS flow. Furthermore, while a flat SOS flow is able to achieve almost the same geometrical shape as the target density, its learned density still differs from the true density. For mixture of Gaussians, a large number of components for each conditional improved the performance of the resulting model.

We also test the representational power of deep and wide SOS flows and the results are given in Figure 6. In the first row, the true transformation was simulated by stacking multiple blocks of SOS transformation. Subsequently, we generated the target density using this transformation and estimated it using a deep flow, wide flow, wide-deep flow and mixture of Gaussians. In the second row, we simulated the true transformation using a single block SOS transformation and performed the same experiment as before. In both simulations, we tried to break our model by adding random noise to the coefficients of simulated transformation. As the figure shows, however, both deep and wide

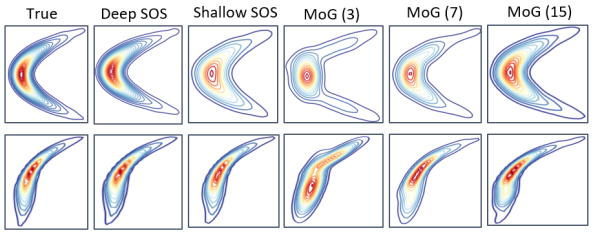

Figure 5. **Top:** Left plot shows the target density given by $p(x_1, x_2) = \mathcal{N}(x_2\,; 0, 4)\mathcal{N}(x_1\,; 0.25x_2^2, 1)$. The second plot shows the density learnt by SOS flows with 3 blocks and a sum of 2 polynomials with degree 3 with ordering (x_1, x_2). Third plot shows the density learnt by SOS flows with 1 block and a sum of 2 polynomials with degree 4 and ordering (x_1, x_2). The last three plots estimate this density using a Mixture of Gaussian conditionals with varying components given in parenthesis and ordering (x_1, x_2). **Bottom:** Same as Top but with target density given by $p(x_1, x_2) = \mathcal{N}(x_2\,; 2, 2)\mathcal{N}(x_1\,; 0.33x_1^3, 1.5)$.

variants performed equally well in terms of representation. As expected however, the training time for wider flows was significantly longer than that for deeper flows.

C.3. Real-World Datasets

We also performed density estimation experiments on 5 real world datasets that include four datasets from the UCI repository and BSDS300. These datasets have been previously considered for comparison of flows based methods (Huang et al., 2018).

The SOS transformation was trained using maximum likelihood method with source density as standard normal distribution. We used stochastic gradient descent to train our models with a batch size of 1000, learning rate = 0.001, number of stacked blocks = 8, number of polynomials (k) = 5 and, degree of polynomials (r) = 4 with number of epochs for training = 40. We compare our method to previous works on normalizing flows and autoregressive models which include

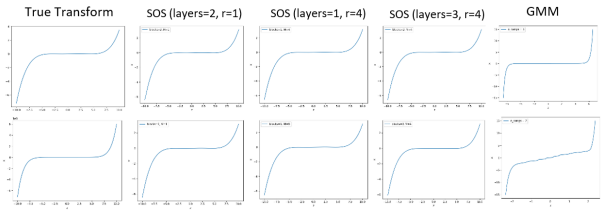

Figure 6. **Top Row:** Transformation defined by a deep SOS flow with $M = r = 1$ and blocks =4. The next three plots show SOS flows learning this transformation with different configurations (deep, wide and, wide-deep). The last plot shows the transformation learned when a Gaussian mixture model learns the density (or transformation). **Bottom Row:** Same as Top Row but the true transformation was derived by a wide and shallow SOS flow with $M = r = 4$ and blocks=1.

MADE-MoG (Germain et al., 2015), MAF (Papamakarios et al., 2017), MAF-MoG (Papamakarios et al., 2017) and NAFs (Huang et al., 2018). In Table 3, we report the average log-likelihood obtained using 10 fold cross-validation on held-out test sets for SOS flows. The performance reported for other methods are those reported in (Huang et al., 2018). The results show that SOS flows are able to achieve competitive performance as compared to other methods.

D. Proofs

Lemma 1 (Mulansky & Neamtu 1998)**.** *Let S be a dense subspace of X and let $C \subseteq X$ be a convex set such that $\mathrm{int}(C) \neq \emptyset$. Then $C \cap S$ is dense in C.*

Proof. Since the interior $\mathrm{int}(C)$ is open and nonempty, and S is dense, we know $\mathrm{int}(C) \cap S$ is dense in $\mathrm{int}(C)$. (Every open set of $\mathrm{int}(C)$ is also an open set of X, hence intersects the dense set S.) Moreover, since C is convex and $\mathrm{int}(C) \neq \emptyset$, we know $\mathrm{cl}(\mathrm{int}(C)) = \mathrm{cl}(C)$, hence $\mathrm{cl}(\mathrm{int}(C) \cap S) = \mathrm{cl}(C)$, i.e., $\mathrm{int}(C) \cap S$, whence also the "larger" set $C \cap S$, is dense in C. □

Theorem 3. *Let C be the space of real univariate continuous functions, equipped with the topology of compact convergence. Then, the set of increasing polynomials is dense in the cone of increasing continuous functions.*[9]

Proof. Let us define P to be the space of polynomials, and I the space of increasing functions. We need only prove on any compact set K, the set of polynomials of the form (5), i.e. $\mathsf{I} \cap \mathsf{P}$ thanks to Theorem 2, is dense in $\mathsf{C}(K) \cap \mathsf{I}$. By Weierstrass' theorem we know P is dense in $\mathsf{C}(K)$. Moreover, the convex subset $\mathsf{I} \cap \mathsf{C}(K)$ has nonempty interior (take say a linear function with positive slope). Applying Lemma 4 of (Mulansky & Neamtu, 1998) completes the proof[10]. □

[9] This result does not seem to be surprising, given that it is well-known polynomials are dense in C. Note however that it does *not* follow immediately. For instance, it took significant effort for Huang et al. (2018) to prove a similar result, which is arguably one of their main contributions.

E. Stacked SOS Flows

Figure 7 shows a schematic for SOS flows with multiple blocks stacked together. The SOS transformation is depicted in Figure 1.

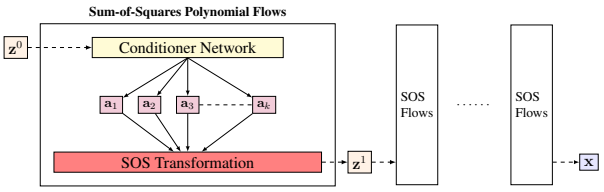

Figure 7. Schematic of SOS flows by stacking multiple blocks of SOS transformation.

[10] We thank Csaba Szepesvári to bring (Mulansky & Neamtu, 1998) to our attention, which allowed us to reduce a lengthy proof to the current slim one.