Riemannian SOS-Polynomial Normalizing Flows*

Jonathan Schwarz[1,2], Felix Draxler[1,2,3], Ulrich Köthe[3], and Christoph Schnörr[1,2]

[1] Heidelberg Collaboratory for Image Processing, Heidelberg University, Germany
[2] Image and Pattern Analysis Group, Heidelberg University, Germany
[3] Visual Learning Lab, Heidelberg University, Germany

Abstract. Sum-of-Squares polynomial normalizing flows have been proposed recently, without taking into account the convexity property and the geometry of the corresponding parameter space. We develop two gradient flows based on the geometry of the parameter space of the cone of SOS-polynomials. Few proof-of-concept experiments using non-Gaussian target distributions validate the computational approach and illustrate the expressiveness of SOS-polynomial normalizing flows.

Keywords: normalizing flows · SOS polynomials · Riemannian gradient flows.

1 Introduction

Optimal transport has become a central topic for mathematical modelling [24,19] and for computational approaches to data analysis and machine learning [17]. Wasserstein distances based on various transportation cost functions and their dual formulations, parametrized by deep networks, provide a framework for generative data-driven modeling [6].

A current prominent line of research initiated by [21,20] concerns the representation and estimation of so-called normalizing flows, in order to model a data distribution ν in terms of an elementary reference measure μ, typically the standard Gaussian $\mu = \mathcal{N}(0, I_n)$, as pushforward measure $\nu = T_\sharp \mu$ with respect to a transportation map (diffeomorphism) T. This framework supports a broad range of tasks like density estimation, exploring a posteriori distributions, latent variable models, variational inference, uncertainty quantification, etc. See [14,12,16] for recent surveys.

A key requirement is the ability to evaluate efficiently both T and T^{-1} along with the corresponding Jacobians. Based on classical work [11], triangular maps T and their relation to optimal transport, therefore, has become a focus of research [8,4]. While the deviation from *optimal* transport, as defined by [7], can be bounded by transportation inequalities [22], merely regarding triangular maps

* This work is supported by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy EXC-2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster).

T as diffeomorphisms (performing *non*-optimal transport) does not restrict expressiveness [3]. Accordingly, triangular maps parametrized by deep networks are nowadays widely applied.

Contribution, Organization. A basic property that ensures the invertibility of T is monotonicity, which in connection with triangular maps can be achieved by the coordinatewise integration of nonnegative functions. In a recent paper [10], Sum-of-Squares (SOS) polynomials that are nonnegative by construction, were used for this purpose, as part of the standard procedure for training deep networks. However, both the convexity properties and the geometry of the parameter space of the cone of SOS polynomials [13,2] were completely ignored. In this work, we take this geometry into account and devise computational approaches to the construction of transportation maps T. Specifically, we contribute:

- We introduce basic notions in Section 2 and specify the parametrization of triangular transportation maps using SOS polynomials in Section 3.
- Based on this parametrization, two algorithms for learning the parameters from given data are developed in Section 4. Algorithm 1 directly exploits the Riemannian geometry of the positive definite matrix cone. Algorithm 2 pulls back the objective function to the tangent bundle and performs ordinary gradient descent using a Krylov subspace method for approximating a related matrix-valued entire function.
- We evaluate both algorithms and the expressiveness of SOS-polynomial flows in Section 5 using few academical non-Gaussian distributions. To enable a clear assessment, we do not use a deep network for additional parametrization.

Our findings regarding the first algorithm are quite positive which stimulates further research on suitable extensions to large problem dimensions.

Notation. Let $n \in \mathbb{N}$, then $[n]$ denotes the set $\{1, 2, \ldots, n\}$. We denote the the vector space of real multivariate polynomials in n variables of degree at most $d \in \mathbb{N}$ by $\mathbb{R}[x]_d = \mathbb{R}[x_1, \ldots, x_n]_d$. $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in \mathbb{R}[x]_d$ is a monomial corresponding to $\alpha \in \mathbb{N}_d^n = \{\alpha \in \mathbb{N}^n \colon |\alpha| = \sum_{i\in[n]} \alpha_i \leq d\}$. The vectors

$$v_d(x) = (x^\alpha) \in \mathbb{R}^{s_n(d)}, \quad \alpha \in \mathbb{N}_d^n, \quad s_n(d) = \binom{n+d}{d}, \tag{1.1}$$

that comprise all monomials in n variables of degree not greater than d, form a basis of $\mathbb{R}[x]_d$. The number n of variables is implicitly determined by the number of arguments, and may vary. For example, if $d = 2$, then $v_2(x) = (1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^2)^\top$ with $s_n(2) = \frac{1}{2}(n+1)(n+2)$. We set $t_n(d) = s_{n-1}(d) = \dim v_d(x_1, \ldots, x_{n-1}, 0)$. S^n, S_+^n and $\mathcal{P}_n$ denote the spaces of symmetric matrices, of symmetric and positive semidefinite matrices, and of symmetric and positive definite matrices, respectively, of dimension $n \times n$. $\langle a, b\rangle = a^\top b$ denotes the Euclidean inner product of $a, b \in \mathbb{R}^n$.

2 Preliminaries

2.1 Normalizing Flows

Let μ and ν denote the reference measure and the target measure supported on $\mathbb{R}^n$, respectively. Throughout this paper, we assume that $\mu = \mathcal{N}(0, I_n)$ is the standard multivariate Gaussian distribution and that ν is absolutely continuous with respect to the Lebesgue measure such that

$$d\mu(x) = p(x)dx, \qquad d\nu(y) = q(y)dy \tag{2.1}$$

with density functions p, q. Our objective is to compute a smooth diffeomorphism $T\colon \mathbb{R}^n \to \mathbb{R}^n$ such that $\nu = T_\sharp\mu$ becomes the pushforward (or image) measure of μ with respect to T, defined by

$$\nu(V) = \mu\big(T^{-1}(V)\big), \qquad V \subset \mathbb{R}^n, \tag{2.2}$$

for all measurable subsets V. In terms of the densities (2.1), Eq. (2.1) reads

$$q(y) = p\big(T^{-1}(y)\big)\,|\det dT^{-1}(y)|, \tag{2.3a}$$

$$p(x) = q\big(T(x)\big)\,|\det dT(x)|, \qquad y = T(x) \tag{2.3b}$$

with the Jacobian matrices dT, dT^{-1}. As detailed in Sections 2.2 and 3, we consider a subclass of diffeomorphisms

$$\mathcal{T}_\mathcal{A} = \{T_\mathcal{A} \in \mathrm{Diff}(\mathbb{R}^n)\colon \mathcal{A} \in \mathcal{P}_{n,d}\}, \tag{2.4}$$

whose elements are defined by (3.5b) and (3.6). Assuming samples

$$\{y_i\}_{i\in[N]} \sim \nu \tag{2.5}$$

from the target distribution to be given, the goal is to determine some $T_\mathcal{A} \in \mathcal{T}_\mathcal{A}$ such that (2.2) approximately holds. To this end, following [14, Section 4], we set $S_\mathcal{A} = T_\mathcal{A}^{-1}$ and consider the KL divergence

$$\mathrm{KL}\left((S_\mathcal{A})_\sharp q \| p\right) = \mathrm{KL}\left(q \| (T_\mathcal{A})_\sharp p\right) = \mathbb{E}_q[-\log p \circ S_\mathcal{A} - \log\det dS_\mathcal{A}] + c, \tag{2.6}$$

where the constant c collects terms not depending on $S_\mathcal{A}$. Replacing the expectation by the empirical expectation defines the objective function

$$J\colon \mathcal{P}_{n,d} \to \mathbb{R}, \qquad J(\mathcal{A}) = \frac{1}{N}\sum_{i\in[N]}\Big(-\log p\big(S_\mathcal{A}(y_i)\big) - \log\det dS_\mathcal{A}(y_i)\Big). \tag{2.7}$$

After detailing the class of maps (2.4) in Sections 2.2 and 3, the Riemannian gradient flow with respect to (2.7) will induce a *normalizing flow* of q to p (Section 4).

2.2 Triangular Increasing Maps

A mapping $T\colon \mathbb{R}^n \to \mathbb{R}^n$ is called *triangular* and *increasing*, respectively, if each component function T_k only depends on variables x_i with $i \leq k$ (property (2.8a)) and if each function (2.8b) is increasing in x_k.

$$T_k(x) = T_k(x_1, \ldots, x_k), \quad \forall k \in [n] \tag{2.8a}$$

$$x_k \mapsto T_k(x_1, \ldots, x_k), \quad \forall k \in [n] \tag{2.8b}$$

The existence of a triangular map $T\colon C_1 \to C_2$ for any two open solid convex subsets $C_1, C_2 \subset \mathbb{R}^n$ was shown by Knothe [11]. More generally, the existence of a unique (up to μ-equivalence) triangular increasing map T that achieves (2.2), for any given absolutely continuous probability measures μ, ν, was established by [3, Lemma 2.1]. Property (2.8a) implies that the Jacobian matrices dT and dT^{-1} are triangular, which is computationally convenient in connection with (2.3) and (2.7).

3 SOS Polynomials and Triangular Increasing Maps

In this section, we adopt the approach from [10] using SOS polynomials for the construction of increasing triangular maps. The key difference is that we will exploit the geometry and convexity of the parameter space in Section 4 for deriving normalizing flows.

Definition 1 (SOS polynomial [13]). *A polynomial $p \in \mathbb{R}[x]_{2d}$ is a sum-of-squares (SOS) polynomial if there exist $q_1, ..., q_m \in \mathbb{R}[x]_d$ such that*

$$p(x) = \sum_{k \in [m]} q_k^2(x). \tag{3.1}$$

We denote the subset of SOS polynomials by $\Sigma[x]_{2d} \subset \mathbb{R}[x]_{2d}$.

The following basic proposition says that each SOS polynomial corresponds to a parameter matrix A on the positive definite manifold.

Theorem 1 ([2, Thm. 3.39]). *A polynomial $p(x) = \sum_{\alpha \in \mathbb{N}^n_{2d}} p_\alpha x^\alpha$ is SOS if and only if there exists a matrix A such that*

$$p(x) = \langle v_d(x), A v_d(x) \rangle, \qquad A \in \mathcal{P}_{s_n(d)}. \tag{3.2}$$

Note that $p(x) \geq 0$, $\forall x \in \mathbb{R}^n$, by construction. Next, we use (2.8) and the representation (3.2) in order to define a family (2.4) of increasing triangular maps. Based on (3.2), define the sequence of SOS polynomials

$$p_{[k]}(x) := p_{[k]}(x_1, \ldots, x_k) = \langle v_d(x_1, \ldots, x_k), A_{[k]} v_d(x_1, \ldots, x_k) \rangle \tag{3.3a}$$

$$\in \Sigma[x_1, \ldots, x_k]_{2d}, \quad A_{[k]} \in \mathcal{P}_{s_k(d)}, \quad k \in [n] \tag{3.3b}$$

and the sequence of linear forms

$$\langle c_{[k]}, v_d(x_1, \dots, x_{k-1}, 0)\rangle, \qquad c_{[k]} \in \mathbb{R}^{t_n(d)}, \quad k \in [n] \tag{3.4}$$

that are parametrized by symmetric positive definite matrices $A_{[k]}$ and vectors $c_{[k]}$, respectively. We collectively denote these parameters by

$$\mathcal{A} := \{c_{[1]}, \dots, c_{[n]}, A_{[1]}, \dots, A_{[n]}\} \in \mathcal{P}_{n,d} \tag{3.5a}$$

$$\mathcal{P}_{n,d} := \mathbb{R}^{t_1(d)} \times \cdots \times \mathbb{R}^{t_n(d)} \times \mathcal{P}_{s_1(d)} \times \cdots \times \mathcal{P}_{s_n(d)}. \tag{3.5b}$$

Then the map

$$T_{\mathcal{A}} \in \mathrm{Diff}(\mathbb{R}^n), \qquad x \mapsto T_{\mathcal{A}}(x) = \big(T_{[1]}(x_1), \dots, T_{[n]}(x_1, \dots, x_n)\big)^\top \tag{3.6a}$$

$$T_{[k]}(x_1, \dots, x_k) = \langle c_{[k]}, v_d(x_1, \dots, x_{k-1}, 0)\rangle \tag{3.6b}$$

$$+ \int_0^{x_k} p_{[k]}(x_1, \dots, x_{k-1}, \tau)\, \mathrm{d}\tau \tag{3.6c}$$

is *triangular and increasing* due to the nonnegativity of the SOS polynomials $p_{[k]}$. The inverse maps $S_{\mathcal{A}} = T_{\mathcal{A}}^{-1}$ have a similar structure and could be parametrized in the same way. The objective function (2.7) therefore is well defined.

4 Riemannian Normalizing Flows

In this section, we will develop two different gradient descent flows with respect to the objective function (2.7) that take into account the geometry of the parameter space $\mathcal{P}_{n,d}$ (3.5b). Either flow is supposed to transport the target measure ν that is only given through samples (2.5), to the reference measure μ. This will be numerically evaluated in Section 5.

Section 4.1 works out details of the Riemannian gradient flow leading to Algorithm 1. Section 4.2 develops a closely related flow using different numerical techniques, leading to Algorithm 2. In what follows, the tangent space to (3.5b) at $\mathcal{A}$ is given and denoted by

$$\mathcal{S}_{n,d} = T_{\mathcal{A}}\mathcal{P}_{n,d} = \mathbb{R}^{t_1(d)} \times \cdots \times \mathbb{R}^{t_n(d)} \times S^{s_1(d)} \times \cdots \times S^{s_n(d)}. \tag{4.1}$$

4.1 Riemannian Gradient

Consider the open cone of positive definite symmetric $n \times n$ matrices $\mathcal{P}_n$. This becomes a Riemannian manifold [1] with the metric

$$g_A(U, V) = \mathrm{tr}(A^{-1}UA^{-1}V), \qquad U, V \in T_A\mathcal{P}_n = S^n. \tag{4.2}$$

The Riemannian gradient of a smooth function $J\colon \mathcal{P}_n \to \mathbb{R}$ reads

$$\mathrm{grad}\, J(A) = A\,(\partial_A J(A))\,A, \tag{4.3}$$

where $\partial J(A)$ denotes the Euclidean gradient. The exponential map is globally defined and has the form

$$\exp_A(U) = A^{\frac{1}{2}} \operatorname{expm}(A^{-\frac{1}{2}} U A^{-\frac{1}{2}}) A^{\frac{1}{2}}, \qquad A \in \mathcal{P}_n, \quad U \in S^n, \tag{4.4}$$

with the matrix exponential function $\operatorname{expm}(B) = e^B$, $B \in \mathbb{R}^{n\times n}$. Discretizing the flow using the geometric explicit Euler scheme with step size h and iteration counter $t \in \mathbb{N}$ yields

$$A_{t+1} = \exp_{A_t}\big(-h \operatorname{grad} J(A_t)\big) \tag{4.5a}$$

$$= A_t^{\frac{1}{2}} \operatorname{expm}\big(-h A_t^{\frac{1}{2}} \partial_A J(A_t) A_t^{\frac{1}{2}}\big) A_t^{\frac{1}{2}}, \qquad t \in \mathbb{N}, \quad A_0 \in \mathcal{P}_n. \tag{4.5b}$$

Applying this discretization to the respective components of (3.5) yields the following natural gradient flow for the objective function (2.7):

Algorithm 1: Riemannian SOS Flow

Initialization
Choose $\mathcal{A}_0 \in \mathcal{P}_{n,d}$ such that $T_{[1]} \approx \mathrm{id}$.
while ***not converged*** **do**
$(A_{[k]})_{t+1} =$
$(A_{[k]})_t^{\frac{1}{2}} \operatorname{expm}\big(-h (A_{[k]})_t^{\frac{1}{2}} \partial_{A_{[k]}} J(\mathcal{A}_t)(A_{[k]})_t^{\frac{1}{2}}\big)(A_{[k]})_t^{\frac{1}{2}}, \quad \forall k \in [n],$
$(c_{[k]})_{t+1} = (c_{[k]})_t - h\, \partial_{c_{[k]}} J(\mathcal{A}_t), \quad \forall k \in [n].$

4.2 Exponential Parameterization

Consider again first the case of a smooth objective function $J\colon \mathcal{P}_n \to \mathbb{R}$. We exploit the fact that the exponential map (4.4) is *globally* defined on the entire tangent space S^n of $(\mathcal{P}_n, g)$, which does not generally hold for Riemannian manifolds. Using

$$\exp_I(U) = \operatorname{expm}(U), \quad U \in S^n, \tag{4.6}$$

we pull back J to the vector space S^n,

$$\widetilde{J}\colon S^n \to \mathbb{R}, \qquad \widetilde{J}(U) = J \circ \operatorname{expm}(U), \tag{4.7}$$

and perform ordinary gradient descent:

$$U_{t+1} = U_t - h \partial \widetilde{J}(U_t), \qquad t \in \mathbb{N}, \quad U_0 \in S^n. \tag{4.8}$$

Denote the canonical inner product on S^n by $\langle U, V\rangle = \operatorname{tr}(UV)$. Then the gradient of $\widetilde{J}(U)$ is given by the equation

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \widetilde{J}(U + \tau V)\big|_{\tau=0} = \langle \partial \widetilde{J}(U), V\rangle = d_A J \circ d_U \operatorname{expm}(V), \quad \forall V \in S^n, \tag{4.9}$$

where $A = \operatorname{expm}(U)$.

It remains to evaluate the differential of the matrix exponential on the right-hand side of (4.9). Using the vectorization operator vec(.), that turns matrices into vectors by stacking the column vectors, and we have the identity

$$\operatorname{vec}(CXB^\top) = (B \otimes C)\operatorname{vec}(X). \tag{4.10}$$

Thus, by [9, Thm. 10.13], we conclude

$$\operatorname{vec}\big(d_U \operatorname{expm}(V)\big) = K(U)\operatorname{vec}(V) \tag{4.11a}$$

$$K(U) = (I \otimes e^U)\psi\big(U \oplus (-U)\big), \tag{4.11b}$$

where $\otimes$ denotes the Kronecker matrix product [23], $\oplus$ denotes the Kronecker sum

$$A \oplus B = A \otimes I_n + I_n \otimes B, \tag{4.12}$$

and ψ denotes the matrix-valued function given by the entire function

$$\psi = \frac{e^x - 1}{x} \tag{4.13}$$

with matrix argument x. Applying $\operatorname{vec}(\cdot)$ to the left-hand side of (4.9) and substituting (4.11) in the right-hand side gives

$$\big\langle \operatorname{vec}(\partial\widetilde{J}(U)), \operatorname{vec}(V)\big\rangle = \big\langle \operatorname{vec}(\partial J(A)), K(U)\operatorname{vec}(V)\big\rangle, \qquad \forall V \in S^n. \tag{4.14}$$

Hence, taking into account the symmetry of $K(U)$,

$$\partial\widetilde{J}(U) = \operatorname{vec}^{-1}\big(K(U)\operatorname{vec}(\partial J(A))\big). \tag{4.15}$$

As a result, (4.8) becomes

$$U_{t+1} = U_t - h\operatorname{vec}^{-1}\Big(K(U)\operatorname{vec}\big(\partial J(A_t)\big)\Big), \quad A_t = \operatorname{expm}(U_t), \quad U_0 \in S^n. \tag{4.16}$$

In order to evaluate iteratively this equation, the matrix $K(U)$ given by (4.11b) is never computed. Rather, based on [18], the product $K(U)\operatorname{vec}\big(\partial J(A_t)\big)$ is computed by approximating the product $\psi\big(U \oplus (-U)\big)\partial J(A_t)$ as follows. Using the shorthands

$$C = U \oplus (-U), \qquad b = \partial J(A_t) \tag{4.17}$$

one computes the Krylov subspace

$$\mathcal{K}_m(C, q_1) = \operatorname{span}\{q_1, Cq_1, \ldots, C^{m-1}q_1\}, \quad q_1 = \frac{b}{\|b\|} \tag{4.18}$$

using the basic Arnoldi iteration with initial vector q_1, along with a orthonormal basis $V_m = (q_1, \ldots, q_m)$ of $\mathcal{K}_m(C, q_1)$. This yields the approximation

$$\psi\big(U \oplus (-U)\big)\partial J(A_t) \approx \psi(C)b \approx \|b\|V_m\psi(H_m)e_1, \quad H_m = V_m^\top C V_m, \tag{4.19}$$

where $e_1 = (1, 0, \dots, 0)^\top$ denotes the first canonical unit vector. The right-hand side of (4.19) only involves the evaluation of ψ for the much smaller matrix H_m, which can be savely done by computing

$$\psi(H_m)e_1 = \begin{pmatrix} I_m & 0 \\ 0 & 0 \end{pmatrix} \operatorname{expm} \begin{pmatrix} H_m & e_1 \\ 0 & 0 \end{pmatrix} e_{m+1} \tag{4.20}$$

and using any available routine [15] for the matrix exponential. Putting together, the iteration (4.8) is numerically carried out by computing

$$U_{t+1} = U_t - h \operatorname{vec}^{-1}\Big(\|\partial J(A_t)\| \big(I \otimes \operatorname{expm}(U_t)\big) V_m \psi(H_m) e_1 \Big) \tag{4.21a}$$

$$A_t = \operatorname{expm}(U_t), \quad t \in \mathbb{N}, \quad U_0 \in S^n. \tag{4.21b}$$

In view of (4.6), we replace the overall parametrization (3.5a) by

$$\mathcal{U} := \{c_{[1]}, \dots, c_{[n]}, U_{[1]}, \dots, U_{[n]}\} \in \mathcal{S}_{n,d} \tag{4.22a}$$

$$\mathcal{S}_{n,d} := \mathbb{R}^{t_1(d)} \times \cdots \times \mathbb{R}^{t_n(d)} \times S_{s_1(d)} \times S_{s_n(d)}. \tag{4.22b}$$

Consequently, analogous to (4.7), we denote the pulled back objective function (2.7) by $\widetilde{J}(\mathcal{U})$. Applying the procedure worked out above to each positive definite component of the overall parametrization (4.22) results in Algorithm 2.

Algorithm 2: Exponential SOS Flow

Initialization

Choose $\mathcal{A}_0 \in \mathcal{P}_{n,d}$ such that $T_{[1]} \approx \mathrm{id}$.

$U_{[k]} = \operatorname{logm}(A_{[k]})$, $\forall k \in [n]$.

while ***not converged*** **do**

$(A_{[k]})_t = \operatorname{expm}\big((U_{[k]})_t\big)$

$(U_{[k]})_{t+1} = (U_{[k]})_t - h \operatorname{vec}^{-1}\Big(K\big((U_{[k]})_t\big) \operatorname{vec}\big(\partial_{A_{[k]}} J(\mathcal{A}_t)\big)\Big)$

$(c_{[k]})_{t+1} = (c_{[k]})_t - h\, \partial_{c_{[k]}} J(\mathcal{A}_t), \quad \forall k \in [n].$

Remark 1 (polymomial basis). The framework outlined above does not depend on the specific choice of a *monomial* basis (1.1). For example, replacing $v_d(x)$ by

$$Q v_d(x), \quad Q \in \mathrm{GL}\big(s_n(d); \mathbb{R}\big) \tag{4.23}$$

for some linear regular transformation Q, provides a viable alternative. For instance, a polynomial basis that is orthogonal with respect to a weighted L_2 inner product makes sense, especially if prior information about the support $\operatorname{supp} \nu$ of the target measure is available.

4.3 Application: Sampling from the Target Measure

In this section, we consider the objective function (2.7) for the specific case $\mu = \mathcal{N}(0, I_n)$ and the task to generate samples $y = T_{\mathcal{A}}(x) \sim \nu$ from the estimated target measure, using samples $x \sim \mu$ that are simple to compute.

Taking into account the specific form of μ and the triangular structure of $S_{\mathcal{A}}$, the objective function (2.7) simplifies to

$$J(\mathcal{A}) = \frac{1}{N} \sum_{i\in[N]} \sum_{k\in[n]} \Big(\frac{1}{2} \big(S_{[k]}(y_{i,1}, \ldots, y_{i,k})\big)^2 - \log \partial_k S_{[k]}(y_{i,1}, \ldots, y_{i,k}) \Big). \quad (4.24)$$

Both Algorithm 1 and 2 can be used to minimizer (4.24) numerically. The evaluation of the map $T_{\mathcal{A}} = S_{\mathcal{A}}^{-1}$ makes use of the triangular structure of in order to solve the equations

$$S_{\mathcal{A}}(y) = \begin{bmatrix} S_{[1]}(y_1) \\ S_{[2]}(y_1, y_2) \\ \vdots \\ S_{[n]}(y_1, ..., y_n) \end{bmatrix} = x. \quad (4.25)$$

for $y = T_{\mathcal{A}}(x)$ by computing recursively

$$y_k = \big(S_{[k]}(y_1, ..., y_{k-1}, .)\big)^{-1}(x_k), \quad k \in [n]. \quad (4.26)$$

Each step involves few iterations of the one-dimensional Newton method that converges to the unique solution, thanks to the monotonicity of the triangular maps that holds by construction – cf. (3.6).

5 Numerical Experiments

In this section, we report numerical results as proof of concept and discuss the following two aspects:

- Expressiveness of polynomial SOS maps for measure transport and generative modeling (Sections 5.2 and 5.3);
- performance and comparison of the two geometric flows approximated by Algorithms 1 and 2 (Section 5.4).

We point out that unlike the paper [10], no deep network was used for additional parametrization which would obscure the influence of the SOS-polynomial maps.

5.1 Implementation Details.

We used the three two-dimensional densities *open-ring*, *closed-ring* and *mixture of two Gaussians* for this purpose (Figure 5.1), that play the role of the data measure ν. A sample set $y_i \sim \nu$, $i \in [N]$, with $N = 2.000$, was generated as input data.

Next, either algorithm was applied in order to estimate numerically the *SOS*-parameters $\mathcal{A}$ given by (3.5a), by minimizing the objective function (4.24). We used SOS-polynomials of degrees $2d \in \{2, 4, 6\}$ for parametrizing the maps $T_{\mathcal{A}}(x)$. Taking into account the symmetry of the matrices the corresponding

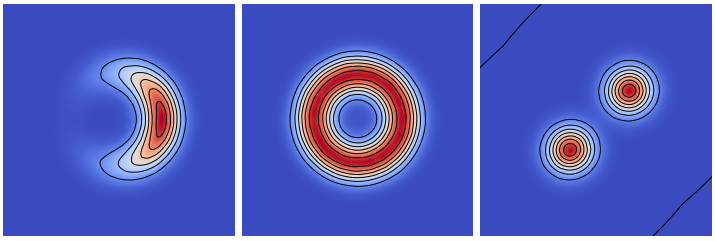

Fig. 5.1: Three non-Gaussian distributions used to evaluate Riemannian SOS-polynomial normalizing flows. From (*left to right*): *open ring*, *closed ring* and *mixture of two Gaussians* distributions.

numbers of variables to be determined are 12, 31, 70. Finally, samples $x_i \sim \mu$ were generated and the map $T_{\mathcal{A}} = S_{\mathcal{A}}^{-1}$ was computed (Section 4.3) in order to generate samples $y_i = T_{\mathcal{A}}(x_i)$. Corresponding kernel density estimates can then be compared to the plots depicted by Figure 5.1.

Both Algorithms 1 and 2 were modified in a stochastic gradient descent like manner: Every update was performed using the gradient with respect to a *single random* index $i \in [N]$ of the objective (4.24), such that each index i was visited after N updates. Thus, even though the considered problem sizes are small, we modified both geometric gradient descent algorithms such that they remain efficient for larger problem sizes [5].

5.2 Riemannian SOS-Polynomial Normalizing Flows

Figure 5.2 displays recovered densities using the procedure described in Section 5.1. See also the figure caption. The low-degree SOS polynomials used to parametrize and estimate the transportation maps $T_{\mathcal{A}}$ suffice to generate samples $y_i = T(x_i)$ by pushing forward samples $x_i \sim \mathcal{N}(0, I_n)$ such that sample y_i follow the ground-truth densities ν depicted by Figure 5.1 quite accurately.

We also checked the influence of changing the polynomial basis according to Remark 1 (page 8). Specifically, Hermite polynomials that are orthogonal with respect to a weighted L_2 inner product were used instead of the canonical monomial basis. Figure 5.4 illustrates that this did not affect the process in a noticeable way. Neither did the result for the Gaussian mixture density show any noticeable effect.

5.3 Exponential SOS-Polynomial Normalizing Flows

We repeated all experiments reported in Section 5.2 using Algorithm 2, instead of Algorithm 1, that is based on the parametrization detailed in Section 4.2. The results are shown by Figure 5.3.

We generally observed fairly good density approximations even for low-degree polynomial parametrizations, that do not achieve the accuracy of the results obtained using the Riemannian flows, however (cf. Figure 5.2). In particular, we

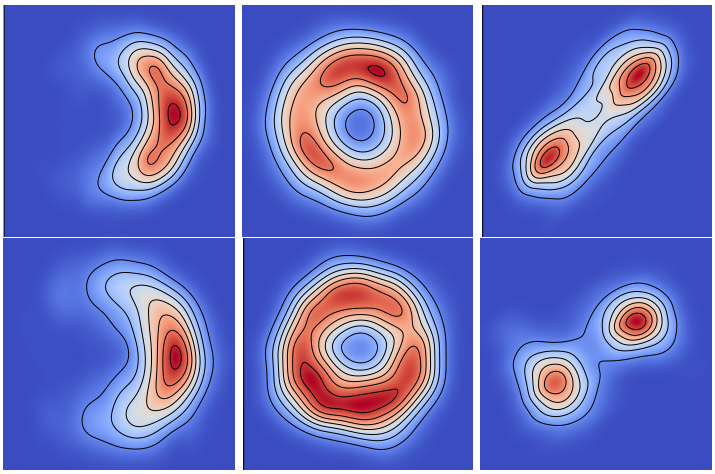

Fig. 5.2: *Riemannian SOS-Polynomial Normalizing Flows.* Kernel density estimate plots based on $N = 2000$ samples $y_i = T_{\mathcal{A}}(x_i) = S_{\mathcal{A}}^{-1}(x_i)$ generated by the transportation maps $T_{\mathcal{A}}$ corresponding to the densities shown by Figure 5.1 and samples $x_i \sim \mathcal{N}(0, I_n)$. The columns correspond from (*left to right*) to the degrees $2d \in \{2, 4, 6\}$ of the SOS-polynomials that were used to compute the increasing triangular maps $T_{\mathcal{A}}$. Except for the mixture of two Gaussians density, low-degree SOS-polynomals suffice to recover the densities quite accurately.

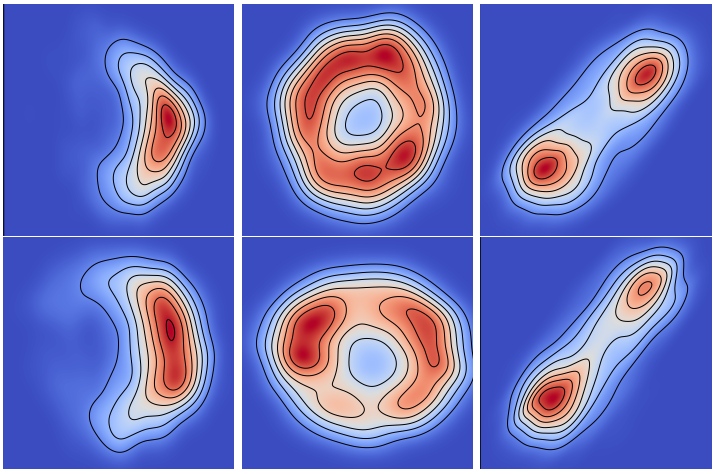

Fig. 5.3: *Exponential SOS-Polynomial Normalizing Flows.* Results of the experiments obtained using Algorithm 2 using the same data as for the experiments illustrated by Figure 5.2. In comparison to the former results, the approximation accuracy deteriorated slightly. In addition, choosing larger polynomial degrees may not improve the result. We attribute this finding to the fact that Algorithm 2 is based on approximating the geometry of the parameter space in various ways (see text).

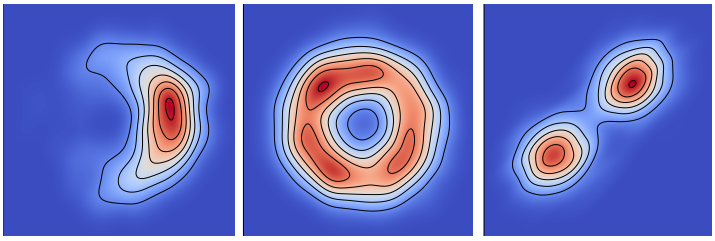

Fig. 5.4: *Riemannian SOS-Polynomial Normalizing Flows* using Hermite polynomials, rather than the canonical monomial basis, and using the same data as for the experiments illustrated by Figure 5.2.

observed that increasing the polynomial degree did *not* systematically improve the approximation.

We attribute this negative finding to two facts: Firstly, Algorithm 2 does not exactly respect the geometry of the parameter space $P_{n,d}$ (3.5b). Secondly, the Krylov subspace approximation underlying the updates (4.21) may also affect the approximation accuracy. We leave a more detailed analysis for future work.

5.4 Comparison Between Riemannian and Exponential SOS Flow

Comparing the results discussed in Sections 5.2 and 5.3 suggests that Riemannian SOS-polynomial normalizing flows should be preferred.

A striking difference concerns the dependency on the polynomial degree. While the Riemannian SOS flow generally yield improved density approximations when the degree is increased, this is hardly the case when using the exponential parametrization. Possible reasons were discussed in the preceding section.

In both cases, however, even small polynomial degrees enable to represent densities by transportation maps quite accurately.

6 Conclusion

We studied transportation maps for generative modeling using Sum-of-Squares polynomials for the construction of increasing triangular maps. Two parametrizations were studied along with two numerical algorithms for estimating the parameters by minimizing a sample-based objective function.

Experiments show that low-degree polynomials suffice to recover basic non-Gaussian distributions quite accurately. Riemannian SOS-polynomial flows that fully respect the geometry of the parameter space perform best, whereas approximations of the geometry may cause detrimental effects.

We merely regard the reported preliminary experimental results as proof of concept, conducted with low-degree parametrizations and small dimension of the underlying domain. Our future work will be devoted to geometric methods for taming the complexity of large degree parametrizations and the representation of high-dimensional generative models.

References

1. Bhatia, R.: Positive Definite Matrices. Princeton Univ. Press (2006)
2. Blekherman, G., Parrilo, P., Thomas, R.R. (eds.): Semidefinite Optimization and Convex Algebraic Geometry. SIAM (2013)
3. Bogachev, V.I., Kolesnikov, A.V., Medvedev, K.V.: Triangular Transformations of Measures. Sbornik: Mathematics **196**(3), 309–335 (2005)
4. Bonnotte, N.: From Knothe's Rearrangement to Brenier's Optimal Transport Map. SIAM J. Math. Anal. **45**(1), 64–87 (2013)
5. Bottou, L., Curtis, F., Nocedal, J.: Optimization Methods for Large-Scale Machine Learning. SIAM Review **60**(2), 223–311 (2018)
6. Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C.J., Schölkopf, B.: From optimal transport to generative modeling: the VEGAN cookbook. CoRR abs/1705.07642 (2017)
7. Brenier, Y.: Polar Factorization and Monotone Rearrangement of Vector-Valued Functions. Comm. Pure Appl. Math. **44**(4), 375–417 (1991)
8. Carlier, G., Galichon, A., 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)
9. Higham, N.: Functions of Matrices: Theory and Computation. SIAM (2008)
10. Jaini, P., Selby, K.A., Yu, Y.: Sum-of-Squares Polynomial Flow. preprint arXiv:1905.02325 (2019)
11. Knothe, H.: Contributions to the Theory of Convex Bodies. The Michigan Math. J. **4**(1), 39–52 (1957)
12. Kobyzev, I., Prince, S.D., Brubaker, M.A.: Normalizing Flows: An Introduction and Review of Current Methods. preprint arXiv:1908.09257 (2019)
13. Marshall, M.: Positive Polynomials and Sum of Squares. Amer. Math. Soc. (2008)
14. Marzouk, Y., Moselhy, T., Parno, M., Spantini, A.: An Introduction to Sampling via Measure Transport. In: Handbook of Uncertainty Quantification, pp. 1–41. Springer (2017)
15. Moler, C., Van Loan, C.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. SIAM Review **45**(1), 3–49 (2003)
16. Papamakarios, G., Nalisnick, E., Rezende, D., Mohamed, S., Lakshminarayanan, B.: Normalizing Flows for Probabilistic Modeling and Inference. preprint arXiv:1912.02762 (2019)
17. Peyré, G., Cuturi, M.: Computational Optimal Transport. CNRS (2018)
18. Saad, Y.: Analysis of Some Krylov Subspace Approximations to the Matrix Exponential Operator. SIAM Journal on Numerical Analysis **29**(1), 209–228 (1992)
19. Santambrogio, F.: Optimal Transport for Applied Mathematicians. Birkhäuser (2015)
20. Tabak, E.G., Turner, C.V.: A Family of Nonparametric Density Estimation Algorithms. Comm. Pure Appl. Math. **66**(2), 145–164 (2013)
21. Tabak, E., Vanden-Eijnden, E.: Density Estimation by Dual Ascent of the Log-Likelihood. Commun. Math. Sci. **8**(1), 217–233 (2010)
22. Talagrand, M.: Transportation Cost for Gaussian and Other Product Measures. Geom. Funct. Anal. **6**, 587–600 (1996)
23. Van Loan, C.F.: The ubiquitous Kronecker product. J. Comput. Appl. Math. **123**, 85–100 (2000)
24. Villani, C.: Optimal Transport: Old and New. Springer (2009)