Input Convex Gradient Networks

Jack Richter-Powell
McGill University
Montréal, Canada
jack.richter-powell@mail.mcgill.ca

Jonathan Lorraine
University of Toronto
Toronto, Canada
lorraine@cs.toronto.edu

Brandon Amos
Facebook AI
bda@fb.com

Abstract

The gradients of convex functions are expressive models of non-trivial vector fields. For example, the optimal transport map between any two measures on Euclidean spaces under the squared distance is realized as a convex gradients via Brenier's theorem, which is a key insight used in recent machine learning flow models. In this paper, we study how to model convex gradients by integrating a Jacobian-vector product parameterized by a neural network, which we call the *Input Convex Gradient Network (ICGN)*. We theoretically study ICGNs and compare them to modeling the gradient by taking the derivative of an input-convex neural network, demonstrating that ICGNs can efficiently parameterize convex gradients.

1 Introduction

The theory of optimal transportation has seen an explosion of computational interest within the machine learning community over the last decade. For example, the Wasserstein metric enables loss functions to leverage the geometry of the underlying space by allowing one to lift any ground cost on a Polish space to the space of measures in a way that metrizes weak convergence. Several works have explored using this distance computationally, such as in Entropic Regularized distance via Sinkhorn divergences [1, 2], or Kantorovich-Rubinstein duality with the W_1 cost [3–5]. The machine learning community has also recently been interested in applications of Brenier's Theorem [6, 7], which states that the optimal map for the W_2 problem is realized as the gradient of a convex function which maximizes the Kantorovich dual problem. This motivates the use of convex gradients for problems such as density estimation and generative modeling, since any density with finite second moment can be realized as the pushforward of a source measure by a convex gradient.

However, in practice, it is difficult to expressively model the gradients of convex functions. The leading approach – the Input Convex Neural Network [8] (ICNN) – models a convex potential which can be differentiated with respect to the inputs to produce a gradient map. Huang et al. [9] combine Brenier's theorem with the ICNN gradients to design flow based density estimators, and Makkuva et al. [10], Korotin et al. [11] use a similar combination to solve high-dimensional barycenter and transport problems. While Huang et al. [9] prove a universal approximation theorem for the ICNN, the result relies on stacking a large number of layers. This detail is not just be theoretical; in Section 4.1, we give a polynomial where a 1-layer ICNN struggles to fit its gradient.

The fundamental difficulty in input-differentiating a neural network to model a gradient is that a product structure emerges [12]. This does not cause issues for training the network on objectives involving the output, like regression, but can emerge in applications where the objectives involve the input-gradient of a network's output [9–11]. Taking the product of activations of a neural network is

Preprint. Under review.

more similar to a polynomial than a neural network, and can suffer from oscillations caused by the Runge phenomena. Instead, it would be desirable to directly model the gradient in a way closer to a feedforward network, while guaranteeing that the model parameterizes a convex gradient.

1.1 Our contributions

For these reasons, we introduce a new class of models which we refer to as *Input Convex Gradient Networks* (ICGN), which is implicitly parameterized by operations on the output of a "hidden" network, similar to the Neural ODE, [13], or Deep Equilibrium Model, [14]. Specifically, we perform a numerical line integral on a symmeterization of the Jacobian, derived from the Gram decomposition of a Positive Semi-Definite (PSD) matrix (see (4)). We take this indirect – and potentially more complex – approach for 2 primary reasons:

1. Modeling the Jacobian directly allows us to enforce constraints on its structure. In Theorem 1, we show our constraints guarantee the model parameterizes a convex gradient. Due to our interpretation via the chain rule, we also believe this avoids having the ill-fated product structure as the ICNN gradient does.
2. Implicitly modeling then integrating allows us to succinctly build complex models, by leveraging the inherent complexity of integration. In this sense, we view our work as a compromise between full ODE models like [13] and a regular feedforward architecture.

The last point relates to what we conjecture is a key benefit of our work. We create a model by integrating a "hidden" network, which we hope to be able to compactly parameterize complex models. In contrast, existing work models gradients by differentiating ICNNs, where the gradient has a similar order of expressiveness to the ICNN itself and is potentially ill-behaved.

As is often said in introductory calculus: "Differentiation is mechanics, but integration is art." Although the gradient of the ICNN is related to the network itself by a relatively simple procedure – symbolically and numerically [15] – the same cannot be said for the integral. It is exactly this lack of a simple relationship we hope will allow us to compactly parameterize complex models with the ICGN.

1.2 Outline

In § 2.1 we motivate the structure of our model, and give an interpretation in terms of the Chain rule. Notably in § 2.3 we describe an operation that computes the line integral of a symmeterized Jacobian, and give conditions when it produces a convex gradient. In § 3 we introduce our model, which consists of applying this operation to a suitable neural network. In § 4 we give two toy experimental implementations, and we round off § 5 by discussing some open problems and directions for future work.

1.3 Related work

Structured higher-order info: Various applications require various guarantees about learned functions. We are primarily interested in convexity, which ICNNs [8] guarantee. [16] look at guaranteeing Lipschitzness, while [17] look at guaranteeing the function is a valid distance. Sobolev networks [18] take the alternative approach of fitting high-order info with a soft penalty. In future work, we are interested in looking at enforcing properties other than convexity with our method.

Implicit Models via integration: Implicit models with no explicit architecture are a powerful tool for generating complex models. One way to generate implicit models is by using the weights are used in some iterative procedure to generate an output – ex., deep equilibrium networks [19]. We are particularly interested in implicit models generated via integration – ex., Neural ODE's [13] have seen recent popularity in applications like spatio-temporal point processes [20]. Jacnet [21] proposed a related method to structure higher-order information by learning them implicitly with integration.

Flow-based models: We are motivated by downstream applications of our methods to flow-based models, which are powerful tools for designing probabilistic models with tractable density.[22] Recently, [23] introduced Convex Potential Flows using the gradient map of an ICNN. Our method could be applied to generate similar convex flows – we are excited for this future work.

Optimal Transport: Makkuva et al. [10] explored using Input Convex Neural Networks for learning transportation maps, while Alvarez-Melis et al. [24] and Mokrov et al. [25] used ICNN's for the Kantorovich dual specifically in the setting of Wasserstein gradient flows [26]. Fan et al. [27] also attempted solving the Wassertein Barycenter [28] problem using ICNNs.

2 Building Convex Gradients

In the following, we give proofs inline when possible, but we defer them to the appendix when they are lengthy.

2.1 A basic structure theorem for convex gradients

We highlight some properties of vector mappings where the Jacobian matrix has a particular structure. Let $G : \mathbb{R}^n \to \mathbb{R}^n$ be a smooth vector field where there exists $V : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ such that

$$DG_x = V^T(x)V(x) \tag{1}$$

where $DG = \frac{\partial G}{\partial x} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the Jacobian matrix of G. Then the following theorem holds:

Theorem 1. *For any G that satisfies (1) there exists a convex function $g : \mathbb{R}^n \to \mathbb{R}$ such that $G = \nabla g$. i.e G is the gradient of convex function.*

The proof follows by combining the following 2 propositions:

Proposition 1. *There exists $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that $\nabla\varphi = G$. i.e, it is a potential field.*

Proof. Treating G as a 1-form by identifying $G = \sum_i G_i dx^i$ (forgiven because the space is flat), G is closed if (see equation 11.21 in [29])

$$\frac{\partial G_i}{\partial x^j} = \frac{\partial G_j}{\partial x^i} \qquad \forall i, j$$

but that is exactly equivalent to the Jacobian of G being symmetric, since the entries are $[DF]_{ij} = \frac{\partial G_i}{\partial x^j}$. But that follows because the product V^TV is symmetric by construction. Now since the domain is the Euclidean space, by the Poincaré lemma [29], all closed 1-forms are exact. Thus there exists a 0-form g such that $dg = G$ however, for 0-forms, the exterior derivative just reduces to the gradient (modulo lowering an index) thus $\nabla g = G$ and so G is conservative. □

Proposition 2. *The g that satisfies $\nabla g = G$ is convex.*

Proof. Since g is smooth, it is sufficient to check that the Hessian of g, $\nabla^2 g$ is Positive Semi-Definite (PSD). But since $\nabla g = G$ it follows that

$$\nabla^2 g = D[\nabla g] = DG = V^TV$$

which is PSD because it is a Gram decomposition. Thus g is convex. □

2.2 Building convex gradients from Gram products

Given the Gram factorization we explored above, it is tempting to ask the following question: given $G : \mathbb{R}^n \to \mathbb{R}^m$ smooth, does there exist $H : \mathbb{R}^m \to \mathbb{R}^n$ such that $DH_{G(x)} = [DG_x]^T$? This question is led by the observation that, by the chain rule, the Jacobian of $H \circ G$ becomes

$$D(H \circ G)_x = DH_{G(x)}DG_x = [DG_x]^T DG_x$$

so by Proposition 1 & 2, $H \circ G$ is the gradient of a convex function. In such a case, we say H *convexifies* G. Here are a few examples of these G, H pairs

1. If $G(x) = Ax$ for some matrix $A \in \mathbb{R}^{d\times d}$, then H is given by $H(x) = A^Tx$.

2. If $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is any smooth, invertible and elementwise function, then it's Jacobian is diagonal of the form (recall x^i is the i'th component of x)

$$D\sigma_x = \begin{pmatrix} \sigma'(x^1) & 0 & \cdots & 0 \\ 0 & \sigma'(x^2) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma'(x^n) \end{pmatrix}$$

   so if $G = \sigma$, the H is given by elementwise γ such that $\gamma' = \sigma' \circ \sigma^{-1}$

3. If A is as in 1. and σ is as in 2., then $G = \sigma \circ A$ is convexified by $H = A^T \circ \gamma$, where γ is such that $\gamma' = \sigma' \circ \gamma$. In that case,

$$DH_{G(x)} = A^T D\gamma_{G(x)} = A^T D\sigma_{\sigma^{-1} \circ \sigma Ax} = A^T D\sigma_{Ax} = [DG(x)]^T$$

   this shows that there are non-trivial non-linear solutions.

Unfortunately, composing these solutions further does not yield G's with closed form solutions.The next proposition,however, provides a characterization in the case where G is assumed to be invertible:

Proposition 3. *Let $G : \mathbb{R}^n \to \mathbb{R}^n$ be an invertible smooth vector field. An H exists such that*

$$DH_{G(x)} = [DG_x]^T \qquad \textit{or equivalently} \qquad DH_x = [DG_{G^{-1}(x)}]^T$$

if and only if

$$\frac{\partial}{\partial x^j}\left(\frac{\partial G_i}{\partial x^k}(G^{-1}(x))\right) = \frac{\partial}{\partial x^i}\left(\frac{\partial G_j}{\partial x^k}(G^{-1}(x))\right)$$

or equivalently, for each 1-form defined by

$$\gamma_k := \sum_{i=1}^{d} \frac{\partial G_i}{\partial x^k}(G^{-1})dx^i$$

γ_k must be closed, i.e $d\gamma_k = 0$.

The proof is given in A, along with a discussion of connections to Brenier's Polar Factorization theorem.

2.3 Gram Products by Integration

In general, even if we can verify that an invertible G satisfies Proposition 3, there is little hope of finding a closed form for H. But given that we know if such an H exists, it must satisfy

$$D(H \circ G)_x = [DG_x]^T DG_x \tag{2}$$

For our purposes, we are interested in computing the composition $H \circ G$, so we will proceed by integrating the right hand side. The following definition makes this precise, and generalizes to the case where G is not assumed invertible:

Definition 1. *Let $G : \mathbb{R}^n \to \mathbb{R}^m$ be a smooth vector field. Assume G satisfies the following partial differential equation:*

$$\frac{\partial^2 G}{\partial x^k \partial x^i} \cdot \frac{\partial G}{\partial x^j} = \frac{\partial^2 G}{\partial x^k \partial x^j} \cdot \frac{\partial G}{\partial x^i} \qquad \forall 1 \leq i, j, k \leq n \tag{3}$$

where the $\cdot$ is a Euclidean dot-product, $\frac{\partial G}{\partial x^i}$ is a vector derivative (i.e a column of the Jacobian). We define the convexification of G to be

$$F(x) = \int_0^1 [DG_{sx}]^T DG_{sx} x ds \qquad F : \mathbb{R}^n \to \mathbb{R}^n \tag{4}$$

The utility of this definition is made clear in the following proposition:

Theorem 2. *The Jacobian of F, DF takes the form*

$$DF = [DG]^T DG$$

by theorem 1, this implies there exists a convex $\varphi : \mathbb{R}^n \to \mathbb{R}$ such that $F = \nabla\varphi$.

Lastly we give an important structural result – a weaker version of the characterization we had in the case where G was invertible, but still sufficient for our needs:

Theorem 3. *Assume a vector field H (in the sense of 2) that convexifies G exists, then $H \circ G = F$ as defined above, i.e*

$$H \circ G(x) = F(x) = \int_0^1 [DG_{sx}]^T DG_{sx} x ds$$

This result can be interpreted as: *if a solution exists, this integral will find it.*

3 Input Convex Gradient Networks

Given a Neural Network $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$ with smooth activations, we can apply the transformation (4) to it. By using automatic differentiation algorithms, we can efficiently compute the Jacobian-vector and Jacobian-Transpose-vector products required for the integrand. In this case we write

$$N_\theta = \int_0^1 [D(M_\theta)_{sx}]^T D(M_\theta)_{sx} x ds$$

When the explicit model M_θ satisfies (3), we refer to the implicit model, N_θ, as an *Input Convex Gradient Network* (ICGN). A few details are in order:

1. To evaluate $N_\theta(x)$ for a point x, we numerically compute the line integral. When considering which quadrature method to use, we have a special constraint – we need the numerical estimator to be unbiased, ruling out deterministic methods like Gaussian Quadrature or Simpson's rule. This is required because the optimizer can learn to modify the network in ways that take advantage of the estimator using the same fixed points.

   In essence, the optimizer fits the quadrature method instead of the function. As a consequence, the output diverges significantly from the true value of the integral. For this reason, we use a Monte Carlo estimator in practice. Despite having poor efficiency per function evaluation, the unbiased nature of the estimator is essential for training.
2. Note that we can have $m > n$ for $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$. In practice this allows us to augment the output dimension, which we can use to increase expressitivity of the network. Additionally, since we are fixing the line integral's path, we are making the constraint $F(0) = 0$. We could also add an explicit constant for $F(0)$ as a parameter.
3. In general, an arbitrary network M_θ will not satisfy the PDE (3). However, M_θ satisfies (3) in the special case where $M_\theta(x) = \sigma(Ax + b)$ – i.e., a 0 or 1 layer ICGN depending on interpretation. This follows by Theorem-3 because by Example 2 we know that a solution exists.

   However, in this 1 layer case it is important to note that because we know H in closed form corresponding to M_θ, the integral is really a proof of concept. We believe it is possible to design constraints for deeper networks so that they satisfy (3). In this case the integral is necessary, but we defer this discussion to 5.
4. We can also work with explicit models M_θ without proving they satisfy (3). Because the path in 4 is fixed, the output of the implicit model N_θ remains well defined. Although we lose the closedness and convexity guarantees, for practical applications they might not be necessary. The result is still a highly compact and expressive model, similar to what was first formulated by Lorraine and Hossain [21].

4 Experiments

We include a Google Colaboratory notebook for reproduction of the experiments here.

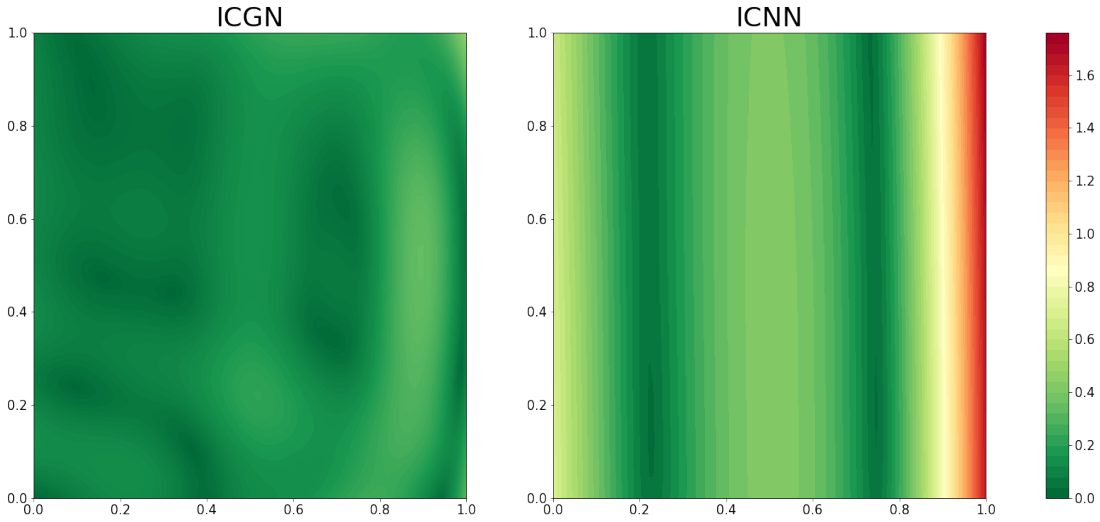

Figure 1: We compare methods for learning convex gradients on the map in Equation 5. Our method – the ICGN – is able to effectively approximate the map with very few parameters, while the moderately sized ICNN struggles.

4.1 A polynomial example

This experiment compares how large of an ICGN we need approximate the gradient of a convex function with Input Convex Neural Networks (ICNN). We approximate the following map as our toy example:

$$T(x, y) = \begin{pmatrix} 4x^3 + \frac{1}{2}y + x \\ 3y - y^2 + \frac{1}{2}x \end{pmatrix} \tag{5}$$

This map is the gradient of the function $\varphi(x, y) = x^4 + \frac{x^2}{2} + \frac{xy}{2} + \frac{3y^2}{2} - \frac{y^3}{3}$, which is convex on $[0, 1]^2$. In Figure 1 we display the error between a ICGN with few parameters and a ICNN with many more parameters. The ICGN has 15 parameters – it is extremely small, with 0 hidden layers i.e $M_\theta = \sigma(Ax + b)$ and an output dimension of 5. The ICNN does not learn reasonable functions with 0 hidden layers, so we display a 1-layer ICNN. We did not find the number of hidden units strongly impacted the 1-layer ICNN performance – we use 25 hidden units in Figure 1 for 78 total parameters. By adding a second layer, the ICNN was able to fit the function correctly.

Takeaway: Our model – the ICGN – effectively approximates the convex gradient with far fewer parameters than the ICNN.

5 Future Directions

Here we list open questions as as interesting directions for future work:

- How can we parameterize deeper networks for M_θ while still ensuring they solve (3)? In 5.1 we discuss a connection between our models structure and a known PDE, which we hope might guide our search for a solution.
- If using more layers isn't practical (i.e., composition), is there another method of generating more expressive solutions that still solve (3)? We explore the this idea in Appendix D.1.
- Can we use a different product structure than the Gram product (i.e $V^T V$)? Other options could be to use either a Hadamard product, or a Kronecker product. We explore this in Appendix D.2.
- Are there applications for this implicit-integration model besides convexity? For example, diffusion modelling. We believe there is great unexplored potential in using integration for modeling.

5.1 Deeper networks and the Beltrami Equation

The ICGNs Achilles heel is currently that although a 1-layer network satisfies (3), a network with more layers does not. Still, 1 layer working at all suggests that it might be possible to extend to deeper networks, given the right constraints on the layers. Choosing these constraints, however, could be extremely difficult.

Fortunately, it seems a similar problem has been investigated before. Specifically, the question of asking when, given a symmetric matrix function $G : \mathbb{R}^n \to \mathbb{R}^{n \times n}$, there exists $f : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$[Df_x]^T Df_x = G(x)$$

is known as the *Beltrami Equation* or *Beltrami System* in the field of geometric function theory and conformal geometry [30]. [31] provides the sufficient condition of the Weyl Curvature [32] of G (interpreted as a metric) vanishing, for a solution to exist for $\mathbb{R}^n, n \geq 4$. This question is phrased somewhat backwards from ours – we want to start with f and make guarantees about G, but we hope further study of this will prove useful in extending our work.

6 Conclusion

We introduced a method for modelling convex gradients by integrating Jacobian-vector products parameterized by a neural networks – the input convex gradient network (ICGN). We provided theoretical results characterizing what we know about how different flavors of our method will work. We demonstrated initial empirical results with our method showing that we can learn non-trivial vector fields with fewer parameters than competing methods. Finally, we presented various exciting directions for future work.

References

[1] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences, 2017.

[2] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving gans using optimal transport, 2018.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[5] Mevlana Gemici, Zeynep Akata, and Max Welling. Primal-dual wasserstein gan, 2018.

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

[7] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN 9780821833124.

[8] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[9] Chin-Wei Huang, Ricky T. Q. Chen, Christos Tsirigotis, and Aaron C. Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *CoRR*, abs/2012.05942, 2020. URL https://arxiv.org/abs/2012.05942.

[10] Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

[11] Alexander Korotin, Vage Egiazarian, Arip Asadulaev, Alexander Safin, and Evgeny Burnaev. Wasserstein-2 generative networks. *arXiv preprint arXiv:1909.13082*, 2019.

[12] Saeed Saremi. On approximating ∇f with neural networks. *arXiv preprint arXiv:1910.12744*, 2019.

[13] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018.

[14] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *CoRR*, abs/1909.01377, 2019. URL http://arxiv.org/abs/1909.01377.

[15] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[16] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

[17] Silviu Pitis, Harris Chan, Kiarash Jamali, and Jimmy Ba. An inductive bias for distances: Neural nets that respect the triangle inequality. *arXiv preprint arXiv:2002.05825*, 2020.

[18] Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. *arXiv preprint arXiv:1706.04859*, 2017.

[19] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

[20] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. *arXiv preprint arXiv:2011.04583*, 2020.

[21] Jonathan Lorraine and Safwan Hossain. Jacnet: Learning functions with structured jacobians. *ICML INNF Workshop*, 2019.

[22] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows, 2018.

[23] Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. *arXiv preprint arXiv:2012.05942*, 2020.

[24] David Alvarez-Melis, Yair Schiff, and Youssef Mroueh. Optimizing functionals on the space of probabilities with input convex neural networks. *arXiv preprint arXiv:2106.00774*, 2021.

[25] Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, and Evgeny Burnaev. Large-scale wasserstein gradient flows, 2021.

[26] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[27] Jiaojiao Fan, Amirhossein Taghvaei, and Yongxin Chen. Scalable computations of wasserstein barycenter via input convex neural networks, 2021.

[28] Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

[29] J.M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York, 2013. ISBN 9780387217529.

[30] Tadeusz Iwaniec, Gaven Martin, et al. *Geometric function theory and non-linear analysis*. Clarendon Press, 2001.

[31] Tao Cheng, Huanhuan Yang, and Shanshuang Yang. Beltrami system and 1-quasiconformal embeddings in higher dimensions, 2018.

[32] J.M. Lee. *Introduction to Riemannian Manifolds*. Graduate Texts in Mathematics. Springer International Publishing, 2019. ISBN 9783319917559. URL https://books.google.ca/books?id=VUOCDwAAQBAJ.

[33] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994. ISBN 9781107392953. URL https://books.google.ca/books?id=ukd0AgAAQBAJ.

A Proof of Prop 3

Here we give the proof of proposition 3:

Proof.
$\implies$

Assume there exists H that satisfies (3). Then since each row of DH is the gradient of a component of H, they are exact as 1-forms. Thus

$$\gamma_k = \sum_{i=1}^{d} \frac{\partial G_i(G^{-1})}{\partial x^k} dx^i = \sum_{i=1}^{d} \frac{\partial H_k}{\partial x^i} dx^i \implies d\gamma_k = 0$$

and by Prop in Lee [29], the 1-forms γ_k are exact on $\mathbb{R}^d$ if and only if in coordinates,

$$\frac{\partial \gamma_k^j}{\partial x^i} - \frac{\partial \gamma_k^i}{\partial x^j} = 0$$

where γ_k^i denotes the ith coefficient (function) of γ_k. Expanding this expression yields

$$\frac{\partial}{\partial x^i}\left(\frac{\partial G_i(G^{-1})}{\partial x^k}\right) - \frac{\partial}{\partial x^j}\left(\frac{\partial G_i(G^{-1})}{\partial x^k}\right) = 0$$

$\impliedby$ Conversely, assume the 1-forms are closed for all k. By the Poincare lemma [29], closed 1-forms on $\mathbb{R}^d : d \geq 2$ are exact, so there exists 0-forms, G_k such that $dG_k = \nabla G_k = \gamma_k$. But then setting

$$G = \begin{pmatrix} G_1 \\ \vdots \\ G_d \end{pmatrix}$$

we have

$$DG = [DG_{G^{-1}(x)}]^T$$

as desired.

□

In the case where G is not invertible, however, it is harder to say whether such an H exists. We leave exploring this to future work.

A.1 Note about polar factorization theorem

Brenier's polar factorization theorem[6] tells us that

Theorem 4. *If $G \in L^2(\mathbb{R}^d; \mathbb{R}^d)$ is a vector valued L_2 mapping, and*

$$\lambda, \nu \in \mathcal{P}_2(\mathbb{R}^d) \qquad \mu := G_{\#}\lambda$$

are two probability measures with finite second moments, then there exists a convex function φ and a map $S : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$G \circ S = \nabla\varphi \qquad (\nabla\varphi)_{\#}(\nu) = \mu \qquad S_{\#}(\lambda) = \nu$$

and S is the unique L_2 projection of G onto the set of maps that pushforward λ onto ν i.e

$$S = \underset{\sigma \in S(\lambda,\nu)}{argmin} \int_{\mathbb{R}^d} |\sigma(x) - H(x)|^2 \qquad S(\lambda,\nu) = \{\sigma \in L^2(\mathbb{R}^d; \mathbb{R}^d) | \sigma_{\#}(\lambda) = \nu\}$$

Given the similarities between the structure of this theorem – the composition of $G \circ S$ versus our construction of finding H such that $H \circ G$ is a convex gradient – we naturally question whether our construction is some sort of special case. The question can be posed as, given G that satisfies (3), is there a measure ν such that S is smooth and $DS_{G(x)} = DG_x^T$? Then G convexifies S in the sense of 2. We hope to explore this connection in future work.

A.2 Open question about convex functions

An interesting question to consider is the following, given any smooth convex $\varphi : \mathbb{R}^d \to \mathbb{R}$, does there exist a pair of vector fields G, H, where H convexifies G and

$$H \circ G = \nabla \varphi$$

B Proof of theorem 2

Here we prove theorem 2,

Theorem 5. *The Jacobian of F, DF takes the form*

$$DF = [DG]^T DG$$

by theorem 1, this implies there exists a convex $\varphi : \mathbb{R}^d \to \mathbb{R}$ such that $F = \nabla \varphi$.

Proof. To start, we derive the PDE (3). This equation comes from interpreting each row of $[DG_x]^T DG_x$ as a 1-form, then checking the closedness condition. Precisely, for the kth row, we have

$$\left[[DG_x]^T DG_x\right]_k = \left(\frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^1} \quad \cdots \quad \frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^i} \quad \cdots \quad \frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^d}\right)$$

so interpreting this as a 1-form leads to

$$\omega_k = \sum_{i=1}^{d} \left(\frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^i}\right) dx^i$$

so then taking the exterior derivative yields

$$\begin{aligned}
d\omega_k &= \sum_{i=1}^{d} d\left(\frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^i}\right) dx^i \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial}{\partial x^j}\left(\frac{\partial G}{\partial x^k} \cdot \frac{\partial G}{\partial x^i}\right) dx^j \wedge dx^i \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d} \left[\left(\frac{\partial^2 G}{\partial x^j \partial x^k}\right) \cdot \frac{\partial G}{\partial x^i} + \frac{\partial G}{\partial x^k} \cdot \left(\frac{\partial^2 G}{\partial x^j \partial x^i}\right)\right] dx^j \wedge dx^i \\
&= \sum_{i=1}^{d}\sum_{j=1}^{i} \left(\frac{\partial^2 G_i}{\partial x^j \partial x^k} - \frac{\partial^2 G_j}{\partial x^k \partial x^i}\right) dx^i \wedge dx^j
\end{aligned}$$

If $d\omega_k = 0$, then we must have

$$\left(\frac{\partial^2 G_i}{\partial x^j \partial x^k} - \frac{\partial^2 G_j}{\partial x^k \partial x^i}\right) = 0 \qquad 1 \leq i, j, k \leq d$$

again, by the Poincare lemma, if $d\omega_k = 0$, then there exists a 0-form F_k such that $\nabla F_k = \omega_k$. In this case, by Stokes theorem, we have

$$\int_0^1 [DG_{sx}]^T DG_{sx} x ds = \int_0^1 \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_d \end{pmatrix} = \int_{[0,1]} \begin{pmatrix} dF_1 \\ \vdots \\ dF_d \end{pmatrix} = \begin{pmatrix} F_1 \\ \vdots \\ F_d \end{pmatrix}\Bigg|_0^1 = \begin{pmatrix} F_1(x) - F_1(0) \\ \vdots \\ F_d(x) - F_d(0) \end{pmatrix}$$

so setting $F = (F_1, \cdots, F_d)$ we have a vector field that satisfies

$$DF_x = [DG_x]^T [DG_x]$$

□

C Proof of Theorem 3

Proof. Assume there exists H that convexifies G. Then by construction, the Jacobian of $H \circ G$ takes the form $D(H \circ G) = [DG]_x^T [DG]_x$. But this means that by Stokes theorem (4) becomes

$$\int_0^1 [DG]_{sx}^T [DG]_{sx} ds = \int_0^1 D(H \circ G)_{sx} x ds = H \circ G(sx)\Big|_0^1 = H \circ G(x)$$

as desired. □

D Further Directions

D.1 Additive Structure

An alternative to developing deeper networks might be to attempt to add solutions that satisfy (3). The construction (4) is highly non-linear, so computing (4) for a sum may be enough to significantly increase the complexity of our model.

It is easy to check that for two solutions F and G,

$$\frac{\partial^2(G+F)}{\partial x^k \partial x^i} \cdot \frac{\partial(G+F)}{\partial x^j} = \frac{\partial^2(G+F)}{\partial x^k \partial x^j} \cdot \frac{\partial(G+F)}{\partial x^i} \qquad \forall 1 \leq i,j,k \leq n$$

reduces to

$$\frac{\partial^2 G}{\partial x^k \partial x^i} \cdot \frac{\partial F}{\partial x^j} + \frac{\partial^2 F}{\partial x^k \partial x^i} \cdot \frac{\partial G}{\partial x^j} = \frac{\partial^2 G}{\partial x^k \partial x^j} \cdot \frac{\partial F}{\partial x^i} + \frac{\partial^2 F}{\partial x^k \partial x^j} \cdot \frac{\partial G}{\partial x^i}$$

so if we could enforce the condition that

$$\frac{\partial^2 G}{\partial x^k \partial x^i} \cdot \frac{\partial F}{\partial x^j} = \frac{\partial^2 F}{\partial x^k \partial x^i} \cdot \frac{\partial G}{\partial x^j} = 0 \qquad \forall i,j,k : i \neq j$$

then $F + G$ would also be a solution to (3). If we could enforce this, in theory we could add up to n solutions in $\mathbb{R}^n$ in this manner, which may be of use in high dimensions. Of course, it's not obvious how to do this in the case where F, G are neural networks, but it is possible a slight modification of the structure would be sufficient.

D.2 Different Products

While we have focused on structuring the integrand structured via the Gram product $[DG]^T DG$ (due to the chain rule interpretation), we could consider other products. The construction changes, however, because for the two other cases below, we need the matrices we combine to be PSD symmetric to begin with. This means that the only obvious way to use another product, if we denote it $a(\cdot,\cdot)$, is to start with two convex gradients F,G (we could model using the constructions in Example 2), and integrate their composition $a(DF, DG)$. A few options are:

- The Hadamard product, i.e

  $$DG \odot DF \text{ for } G = \nabla g \qquad F = \nabla f$$

  could be used. The famous Schur product theorem tells us that the Hadamard product of any two PSD matrices remains PSD. [33] Unfortunately, though, we lose the interpretation as a Jacobian of a composition that the Gram product yields. Furthermore we still have a similar problem in that not any F and G can be used while recovering a result similar to 2, in this case, the PDE we require the pair F, G to satisfy takes the form

  $$\frac{\partial G_k}{\partial x^i \partial x^k}\left(\frac{\partial F_k}{\partial x^j} - \frac{\partial F_k}{\partial x^i}\right) = \frac{\partial F_k}{\partial x^i \partial x^k}\left(\frac{\partial G_k}{\partial x^j} - \frac{\partial G_k}{\partial x^i}\right)$$

  Like (3), it is not easily to consider when a pair will satisfy this equation. We leave questions concerning this construction to future work.
- The Kronecker product could be also be considered. It also has the nice property that the Kronecker product of two PSD matrices remains PSD. [33] However, we still have the issue of making the rows of $DF \otimes DG$ closed as 1-forms. One can derive a similar PDE as above, but the system is even more complex and restrictive, so we omit it.