Input Convex Gradient Networks

Jack Richter-Powell[1], Jonathan Lorraine[2], Brandon Amos[3]

December 9, 2021

1. McGill University
2. University of Toronto
3. Facebook AI Research

Motivation

- From Brenier's theorem [2], we know that for two measures $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^n)$, such that μ does not give mass to small sets, there exists a convex potential φ such that

$$\nabla \varphi = \underset{T : T_{\#}\mu = \nu}{\operatorname{argmin}} \int_{\mathbb{R}^n} |x - T(x)|^2 d\mu$$

Motivation

- From Brenier's theorem [2], we know that for two measures $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^n)$, such that μ does not give mass to small sets, there exists a convex potential φ such that

$$\nabla \varphi = \underset{T: T_{\#}\mu = \nu}{\operatorname{argmin}} \int_{\mathbb{R}^n} |x - T(x)|^2 d\mu$$

- This result motivates use of convex gradients for estimating OT maps [5], more recently for applications such as Wasserstein gradient flows,[4] density estimation, generative modelling [3]...

Setup

- Popular Existing Approach: Model $\varphi : \mathbb{R}^n \to \mathbb{R}$ using an Input-Convex Neural Network[1], then use automatic differentiation to compute $\nabla\varphi$.

Setup

- Popular Existing Approach: Model $\varphi : \mathbb{R}^n \to \mathbb{R}$ using an Input-Convex Neural Network[1], then use automatic differentiation to compute $\nabla\varphi$.
- But if we want $\nabla\varphi$, why not model it directly? i.e using $N_\theta : \mathbb{R}^n \to \mathbb{R}^n$.

Setup

- Popular Existing Approach: Model $\varphi : \mathbb{R}^n \to \mathbb{R}$ using an Input-Convex Neural Network[1], then use automatic differentiation to compute $\nabla\varphi$.
- But if we want $\nabla\varphi$, why not model it directly? i.e using $N_\theta : \mathbb{R}^n \to \mathbb{R}^n$. In general, N_θ is not a conservative vector field...

Setup

- Popular Existing Approach: Model $\varphi : \mathbb{R}^n \to \mathbb{R}$ using an Input-Convex Neural Network[1], then use automatic differentiation to compute $\nabla\varphi$.
- But if we want $\nabla\varphi$, why not model it directly? i.e using $N_\theta : \mathbb{R}^n \to \mathbb{R}^n$. In general, N_θ is not a conservative vector field...
- To constraint N_θ to parameterize a convex gradient, apply the following theorem

Theorem (3 in paper)

For any smooth $G : \mathbb{R}^n \to \mathbb{R}^n$ such that DG_x is symmetric PSD for all x, 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.

Setup

- Popular Existing Approach: Model $\varphi : \mathbb{R}^n \to \mathbb{R}$ using an Input-Convex Neural Network[1], then use automatic differentiation to compute $\nabla\varphi$.
- But if we want $\nabla\varphi$, why not model it directly? i.e using $N_\theta : \mathbb{R}^n \to \mathbb{R}^n$. In general, N_θ is not a conservative vector field...
- To constraint N_θ to parameterize a convex gradient, apply the following theorem

Theorem (3 in paper)

For any smooth $G : \mathbb{R}^n \to \mathbb{R}^n$ such that DG_x is symmetric PSD for all x, 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.

Our proposed model

- Since the condition we want to enforce is on the Jacobian, we work with the Jacobian (or Hessian of the potential) then integrate.

[1] Terms and conditions apply

Our proposed model

- Since the condition we want to enforce is on the Jacobian, we work with the Jacobian (or Hessian of the potential) then integrate.
- So given a suitable network $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$, we compute

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

We call N_θ an *Input Convex Gradient Network*

[1] Terms and conditions apply

Our proposed model

- Since the condition we want to enforce is on the Jacobian, we work with the Jacobian (or Hessian of the potential) then integrate.
- So given a suitable network $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$, we compute

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

We call N_θ an *Input Convex Gradient Network*

  - $[D(M_\theta)]^T D(M_\theta)$ is symmetric PSD by construction, and for suitable[1] M_θ, $DN_\theta = [D(M_\theta)]^T D(M_\theta)$

[1] Terms and conditions apply

Our proposed model

- Since the condition we want to enforce is on the Jacobian, we work with the Jacobian (or Hessian of the potential) then integrate.
- So given a suitable network $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$, we compute

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

We call N_θ an *Input Convex Gradient Network*

  - $[D(M_\theta)]^T D(M_\theta)$ is symmetric PSD by construction, and for suitable[1] M_θ, $DN_\theta = [D(M_\theta)]^T D(M_\theta)$
  - Use Automatic Differentiation to compute vector products for integrand efficiently ($O(n)$).
  - Use any quadrature method to approximate integral.

[1] Terms and conditions apply

Our proposed model

- Since the condition we want to enforce is on the Jacobian, we work with the Jacobian (or Hessian of the potential) then integrate.
- So given a suitable network $M_\theta : \mathbb{R}^n \to \mathbb{R}^m$, we compute

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

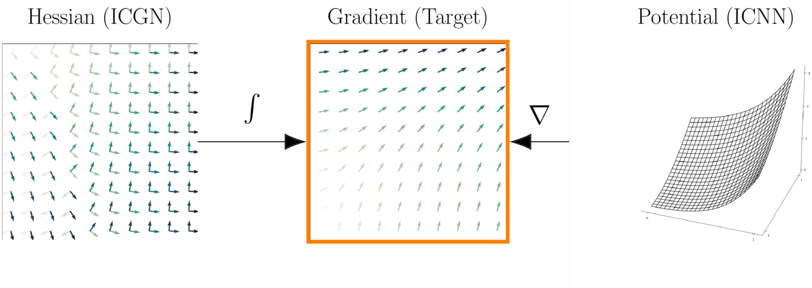
We call N_θ an *Input Convex Gradient Network*

  - $[D(M_\theta)]^T D(M_\theta)$ is symmetric PSD by construction, and for suitable[1] M_θ, $DN_\theta = [D(M_\theta)]^T D(M_\theta)$
  - Use Automatic Differentiation to compute vector products for integrand efficiently ($O(n)$).
  - Use any quadrature method to approximate integral.
  - **Current limitation**: can only use one layer hidden networks.

[1] Terms and conditions apply

Takeaway: ICGN vs ICNN for modelling gradients

As a comparison from our approach to the ICNN gradient:



References

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

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

[3] 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.

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

[5] Subhadip Mukherjee, Sören Dittmer, Zakhar Shumaylov, Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb.

Learned convex regularizers for inverse problems. *CoRR*, abs/2008.02839, 2020. URL
https://arxiv.org/abs/2008.02839.

Thanks to the OTML organizers for a great workshop!