Shape Constraints on Neural Networks

Presenter: Zijing Ou

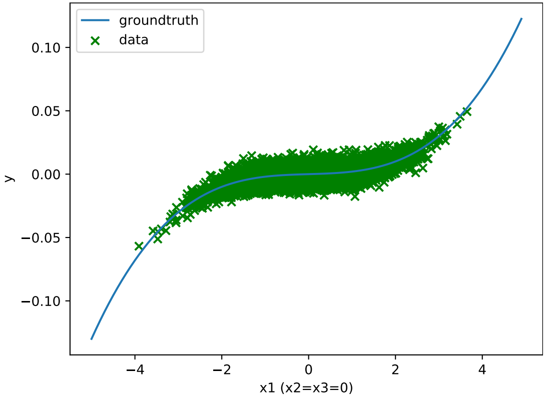
Machine Learning Group, Tencent AI Lab

A Motivating Example

Fitting regression models:

- Dataset: $\mathcal{D} = \{\mathbf{x}, y\}$,
  $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$,
  $h = 0.001(x_1^3 + x_1) + x_2 + \sin(x_3)$,
  $y = h + 0.005\epsilon, \epsilon \sim \mathcal{N}(0, 1)$
- Goal: find $f(\boldsymbol{\theta}) \in \mathcal{H}$ such that

$$y \approx f(\mathbf{x}; \boldsymbol{\theta})$$



A Motivating Example

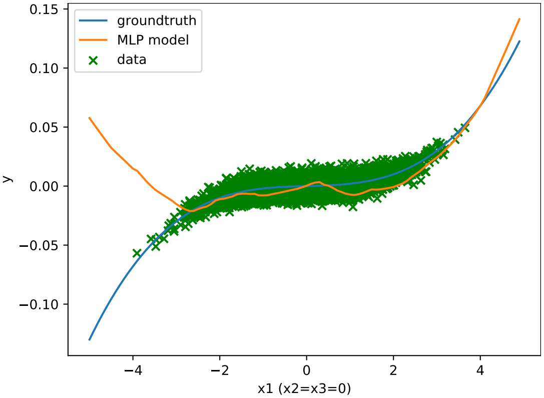
Fitting regression models:

- Dataset: $\mathcal{D} = \{\mathbf{x}, y\}$,
  $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$,
  $h = 0.001(x_1^3 + x_1) + x_2 + \sin(x_3)$,
  $y = h + 0.005\epsilon, \epsilon \sim \mathcal{N}(0, 1)$
- Goal: find $f(\boldsymbol{\theta}) \in \mathcal{H}$ such that

$$y \approx f(\mathbf{x}; \boldsymbol{\theta})$$

- Training:

$$\mathcal{L} = \|\text{MLP}(\mathbf{x}; \boldsymbol{\theta}) - y\|^2$$



A Motivating Example

Fitting regression models:

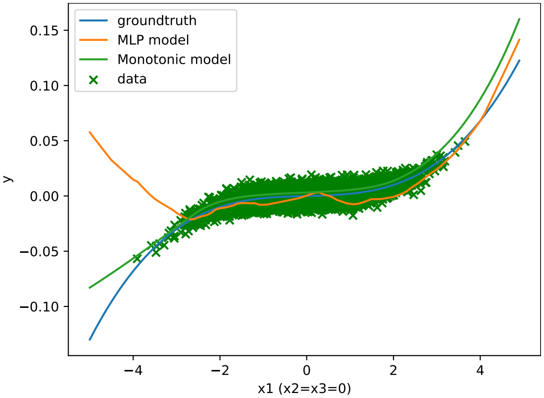
- Dataset: $\mathcal{D} = \{\mathbf{x}, y\}$,
  $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$,
  $h = 0.001(x_1^3 + x_1) + x_2 + \sin(x_3)$,
  $y = h + 0.005\epsilon, \epsilon \sim \mathcal{N}(0, 1)$
- Goal: find $f(\boldsymbol{\theta}) \in \mathcal{H}$ such that

$$y \approx f(\mathbf{x}; \boldsymbol{\theta})$$

- Training:

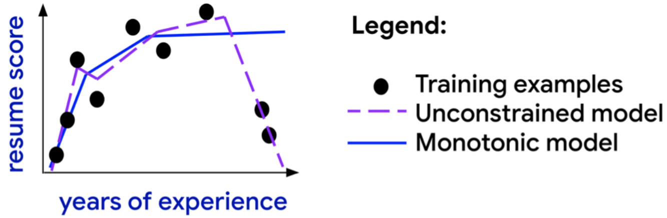
$$\mathcal{L} = \|\text{MLP}(\mathbf{x}; \boldsymbol{\theta}) - y\|^2$$

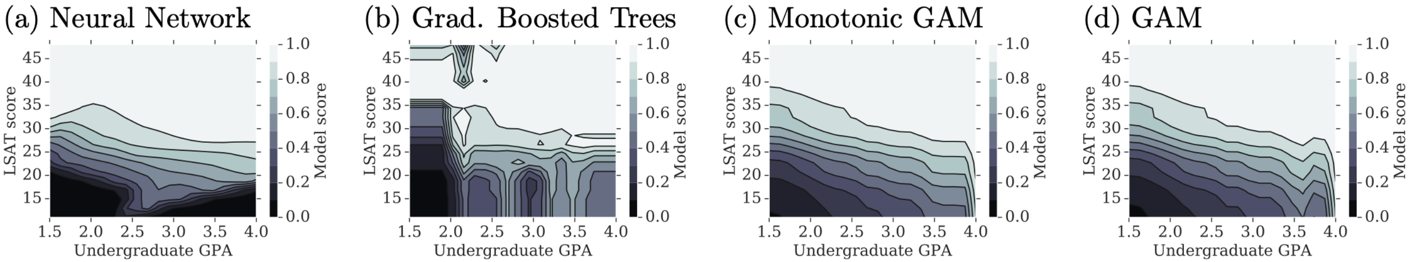
$s.t.$ $\text{MLP}(\mathbf{x}; \boldsymbol{\theta})$ is **monotonic** $w.r.t.$ x_1



Real Life Applications

- Interpretable Machine Learning
- Artificial Intelligent Fairness
- Deontological Ethics





Monotonicity shape constraint

$$f(x, z_1) \leq f(x, z_2), \forall x, z_1 < z_2,$$

where x and z represent the LSAT and GPA score, respectively.

Monotonically Constrained Neural Network

Game plan: learn a monotonic function $F(x;\psi):\mathbb{R}\to\mathbb{R}$

- Imposing its derivative $f(x;\psi) =: \frac{\partial F(x;\psi)}{\partial x} > 0$;
- The monotonic function $F(x;\psi)$ can be parameterized as

$$F(x;\psi) = \int_0^x f(t;\psi)\mathrm{d}t + F(0;\psi),$$

where $f(t;\psi):\mathbb{R}\to\mathbb{R}_+$ is parameterized by neural networks.

Monotonically Constrained Neural Network

Game plan: learn a monotonic function $F(x;\psi):\mathbb{R}\rightarrow\mathbb{R}$

- The monotonic function $F(x;\psi)$ can be parameterized as

$$F(x;\psi)=\int_0^x f(t;\psi)\mathrm{d}t+F(0;\psi), \quad \text{where } f(t;\psi):\mathbb{R}\rightarrow\mathbb{R}_+.$$

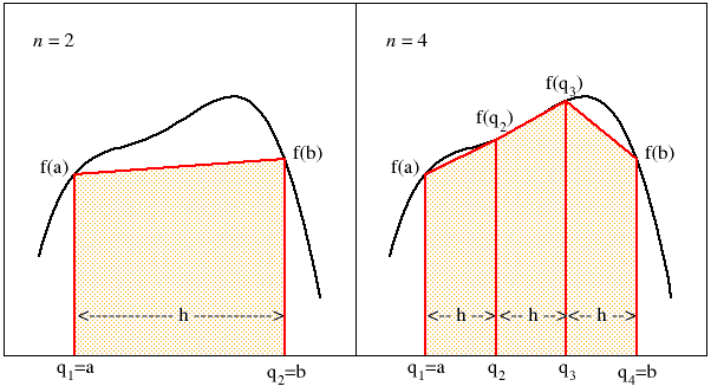
- The derivative of $F(x;\psi)$ *w.r.t.* ψ is

$$\begin{aligned}\nabla_\psi F(x;\psi)&=f(t;\psi)\nabla_\psi t|_{t=0}^{t=x}+\int_0^x\nabla_\psi f(t;\psi)\mathrm{d}t+\nabla_\psi F(0;\psi)\\&=\int_0^x\nabla_\psi f(t;\psi)\mathrm{d}t+\nabla_\psi F(0;\psi),\\ \nabla_x F(x;\psi)&=f(x;\psi).\end{aligned}$$

Monotonically Constrained Neural Network

The computation of $\int_0^x f(t;\psi)\mathrm{d}t$ and $\int_0^x \nabla_\psi f(t;\psi)\mathrm{d}t$:

- Trapezoid quadrature



- More advanced numerical integration methods can be used (*e.g.*, Clenshaw-Curtis quadrature).

Monotonic Flow Based Models

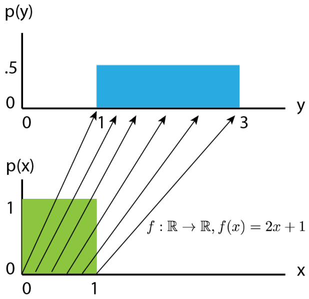
Application Beyond Regression

Flow based models as density estimation

$$\begin{aligned}\text{samples } \boldsymbol{x} &: \mathbb{R}^d;\\ \text{latent variables } \boldsymbol{z} &: \mathbb{R}^d;\\ \text{bijective mapping } \boldsymbol{z} &= g(\boldsymbol{x};\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^d.\end{aligned}$$

According to the change of variables theorem

$$\log p(\boldsymbol{x}) = \log p_z(g(\boldsymbol{x};\boldsymbol{\theta})) + \log\left|\frac{\partial g(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{x}}\right|$$



Monotonic Flow Based Models

Flow based models

$$\log p(\mathbf{x}) = \log p_z(g(\mathbf{x};\boldsymbol{\theta})) + \log\left|\frac{\partial g(\mathbf{x};\boldsymbol{\theta})}{\partial \mathbf{x}}\right|.$$

Notice that two constraints should be satisfied

- The mapping function $\boldsymbol{z} = \log p_z(g(\mathbf{x};\boldsymbol{\theta})$ is invertible;
- The determinant of Jacobian matrix $\left|\frac{\partial g(\mathbf{x};\boldsymbol{\theta})}{\partial \mathbf{x}}\right|$ is tractable.

The monotonically constrained neural networks satisfy these two conditions naturally.

Monotonic Flow Based Models

Consider autoregressive models

$$p(\mathbf{x};\boldsymbol{\theta}) = p(x_1;\boldsymbol{\theta})\prod_{i=1}^{d-1} p(x_{i+1}|\mathbf{x}_{1:i};\boldsymbol{\theta}).$$

Apply flow based transformation for each component

$$\log p(x_i|\mathbf{x}_{1:i-1};\boldsymbol{\theta}) = \log p_z(g(\mathbf{x}_{1:i};\boldsymbol{\theta})) + \log\left|\frac{\partial g(\mathbf{x}_{1:i};\boldsymbol{\theta})}{\partial x_i}\right|.$$

Instantiate $g(\mathbf{x}_{1:i};\boldsymbol{\theta})$ as monotonically constrained neural networks

$$g(\mathbf{x}_{1:i};\boldsymbol{\theta}) = \int_0^{x_i} f(t|\mathbf{x}_{1:i-1};\boldsymbol{\theta})\mathrm{d}t + g(0|\mathbf{x}_{1:i-1};\boldsymbol{\theta}),$$
$$\nabla_{x_i} g(\mathbf{x}_{1:i};\boldsymbol{\theta}) = f(x_i|\mathbf{x}_{1:i-1};\boldsymbol{\theta}).$$

Monotonic Flow Based Models

Monotonic Flow Based Models

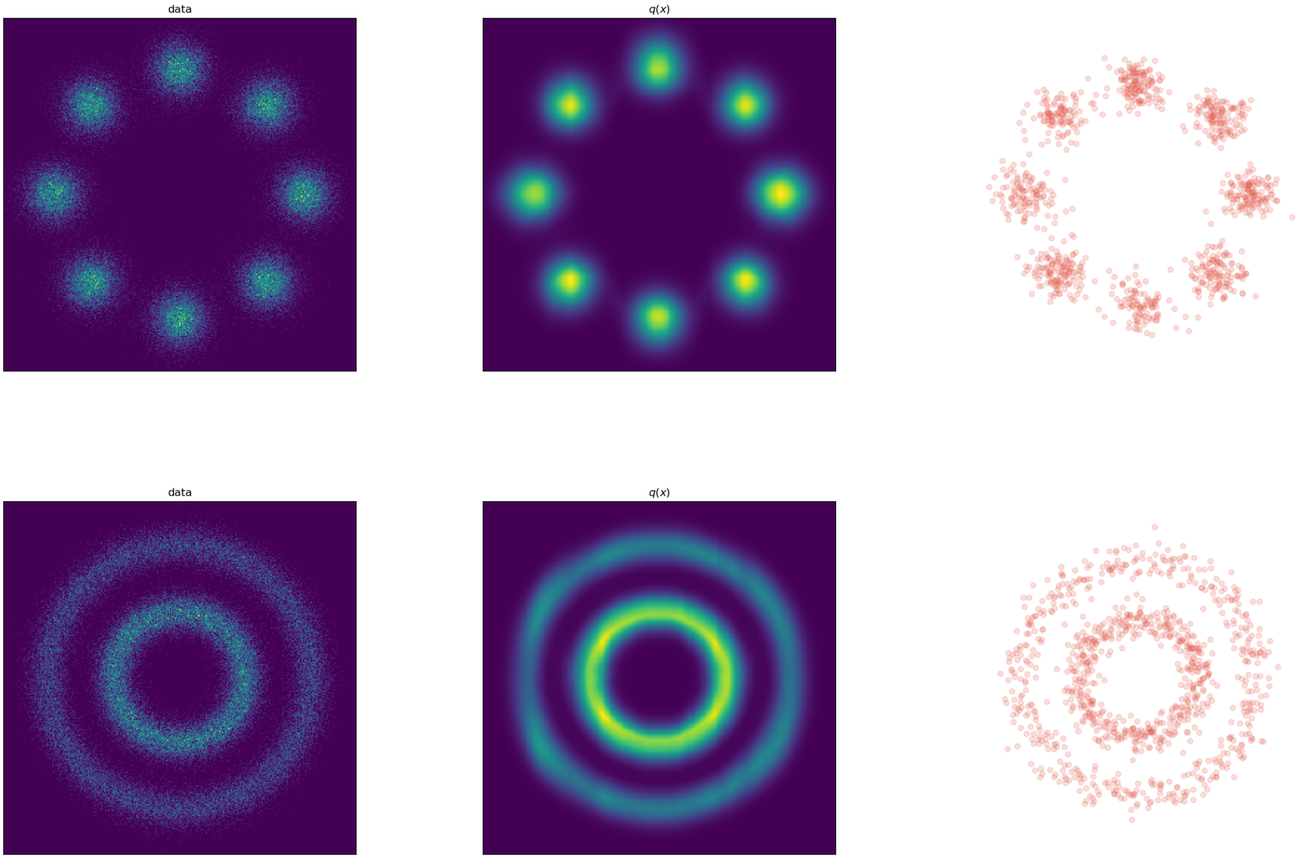
$$\log p(\mathbf{x};\boldsymbol{\theta}) = \log p_z(g(\mathbf{x};\boldsymbol{\theta})) + \sum_{i=1}^{d} \log f(x_i|\mathbf{x}_{1:i-1};\boldsymbol{\theta}),$$

where $p_z(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z};\mathbf{0},\mathbf{1})$.

- **Training.** At each minibatch of samples from $\boldsymbol{x} \sim \mathbf{PoP}_X$
  1. Take transfomation to obtain $\boldsymbol{z_\theta} = g(\mathbf{x};\boldsymbol{\theta})$.
  2. Maximize $\log p_z(\boldsymbol{z_\theta}) + \sum_{i=1}^{d} \log f(x_i|\mathbf{x}_{1:i-1};\boldsymbol{\theta})$ *w.r.t.* $\boldsymbol{\theta}$.
- **Inference.** Given a noisy variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z};\mathbf{0},\mathbf{1})$, compute

$$\boldsymbol{x} = g^{-1}(\boldsymbol{z};\boldsymbol{\theta}).$$

Monotonic Flow Based Models



Shape Constrains Beyond Monotonicity

Dominance: feature A is more important that feature B

- Recent data vs older data in time series models
- Buys vs views on webstores
- A bird in hand vs two birds in the bush

Complements: feature A and feature B are complements

- Guns and ammo
- Vaccine doses and nurses
- Cashiers and cash registers
- CTR (click-through-rate) and # of impressions

Shape Constrains Beyond Monotonicity

Dominance

$$f(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$$
$$\mathbf{x}[d] \in [l_d, u_d]$$
$$L_d(\mathbf{x}) = \mathbf{x} - \boldsymbol{e}_d \odot \mathbf{x} + l_d \boldsymbol{e}_d$$
$$U_d(\mathbf{x}) = \mathbf{x} - \boldsymbol{e}_d \odot \mathbf{x} + u_d \boldsymbol{e}_d$$

- Monotonic Dominance:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[a]} \geq \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[b]} \geq 0$$

- Range Dominance:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[a]} \geq 0, \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[b]} \geq 0$$
$$f(U_a[\mathbf{x}]) - f(L_a[\mathbf{x}]) \geq f(U_b[\mathbf{x}]) - f(L_b[\mathbf{x}])$$

Shape Constrains Beyond Monotonicity

Complements

- Edgeworth:

$$\frac{\partial}{\partial \mathbf{x}[b]}\left(\frac{\partial f(x)}{\partial \mathbf{x}[a]}\right) \geq 0$$

  $f(\mathbf{x})$ is edgeworth $\nRightarrow$ $g(f(\mathbf{x}))$ is edgeworth.

- Trapezoid:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[a]} \geq 0$$

$$\frac{\partial f(L_a(\mathbf{x}))}{\partial \mathbf{x}[b]} \leq 0, \quad \frac{\partial f(U_a(\mathbf{x}))}{\partial \mathbf{x}[b]} \geq 0$$

  $f(\mathbf{x})$ is trapezoid $\Rightarrow$ $g(f(\mathbf{x}))$ is trapezoid, if $g(\cdot)$ is monotonic.

- Joint Monotonicity:

$$\frac{\partial f(x)}{\partial \mathbf{x}[a]} + \frac{\partial f(x)}{\partial \mathbf{x}[b]} \geq 0$$

Shape Constrains Beyond Monotonicity

Increasing Concave Neural Networks

Game plan: learn an increasing concave function $F(x;\psi) : \mathbb{R} \to \mathbb{R}$

- Imposing its 2nd derivative $\frac{\partial^2 F(x;\psi)}{\partial x^2} := -f(x;\psi) \leq 0$;
- The increasing concave function $F(x;\psi)$ can be parameterized as

$$F(x;\psi) = \int_{a=0}^{a=x} \int_{b=a}^{b=\infty} f(b;\psi) \mathrm{d}b \mathrm{d}a,$$

where $f(b;\psi) : \mathbb{R} \to \mathbb{R}_+$ is parameterized by neural networks.

Shape Constrains Beyond Monotonicity

Increasing Concave Neural Networks

Game plan: learn a increasing concave function $F(x;\psi):\mathbb{R}\to\mathbb{R}$

- The monotonic function $F(x;\psi)$ can be parameterized as

$$F(x;\psi)=\int_{a=0}^{a=x}\int_{b=a}^{b=\infty} f(b;\psi)\mathrm{d}b\mathrm{d}a, \quad \text{where } f(b;\psi):\mathbb{R}\to\mathbb{R}_+.$$

- The derivative of $F(x;\psi)$ *w.r.t.* ψ and x is

$$\begin{aligned}
\nabla_\psi F(x;\psi) &= \int_{a=0}^{a=x}\int_{b=a}^{b=\infty} \nabla_\psi f(b;\psi)\mathrm{d}b\mathrm{d}a,\\
\nabla_x F(x;\psi) &= \int_{b=x}^{b=\infty} f(b;\psi)\mathrm{d}b > 0,\\
\nabla_x^2 F(x;\psi) &= -f(x;\psi).
\end{aligned}$$

Shape Constrains Beyond Monotonicity

General Concave Neural Networks

Game plan: learn a general form of concave function $F(x;\psi) : \mathbb{R} \to \mathbb{R}$

- The sign of the 2nd derivative $\frac{\partial^2 F(x;\psi)}{\partial x^2}$ is unknown;
- The general form of concave function $F(x;\psi)$ can be parameterized as

$$F(x;\psi) = \int_{a=0}^{a=x} \int_{b=a}^{b=\infty} f^{+}(b;\psi)\mathrm{d}b\mathrm{d}a + \int_{a=0}^{a=x} \int_{b=0}^{b=a} f^{-}(b;\psi)\mathrm{d}b\mathrm{d}a,$$

where $f^{+}(b;\psi) : \mathbb{R} \to \mathbb{R}_{+}$ and $f^{-}(b;\psi) : \mathbb{R} \to \mathbb{R}_{-}$ are parameterized by neural networks.

Shape Constrains Beyond Monotonicity

General Concave Neural Networks

Game plan: learn a increasing concave function $F(x;\psi):\mathbb{R}\to\mathbb{R}$

- The general concave function $F(x;\psi)$ can be parameterized as

$$F(x;\psi)=\int_{a=0}^{a=x}\int_{b=a}^{b=\infty} f^{+}(b;\psi)\mathrm{d}b\mathrm{d}a+\int_{a=0}^{a=x}\int_{b=0}^{b=a} f^{-}(b;\psi)\mathrm{d}b\mathrm{d}a.$$

- The derivative of $F(x;\psi)$ *w.r.t.* ψ and x is

$$\nabla_{\psi}F(x;\psi)=\int_{a=0}^{a=x}\int_{b=a}^{b=\infty}\nabla_{\psi}f^{+}(b;\psi)\mathrm{d}b\mathrm{d}a+\int_{a=0}^{a=x}\int_{b=0}^{b=a}\nabla_{\psi}f^{-}(b;\psi)\mathrm{d}b\mathrm{d}a,$$

$$\nabla_{x}F(x;\psi)=\int_{b=x}^{b=\infty} f^{+}(b;\psi)\mathrm{d}b+\int_{b=0}^{b=x} f^{-}(b;\psi)\mathrm{d}b,$$

$$\nabla_{x}^{2}F(x;\psi)=-f^{+}(x;\psi)+f^{-}(b;\psi)<0.$$

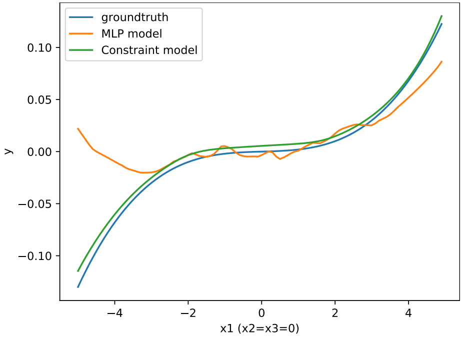
Shape Constrains Beyond Monotonicity[1]

$$\mathcal{D} = \{\mathbf{x}, y\}, \quad \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}), y = f(\mathbf{x})$$

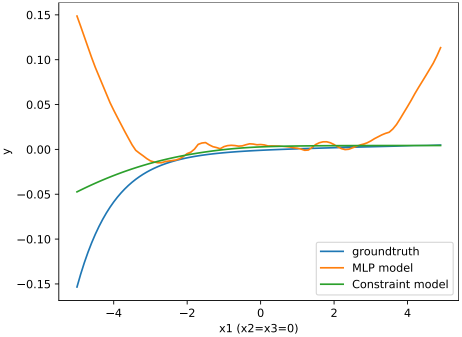
$$\text{monotonic: } f(\mathbf{x}) = 0.001(x_1^3 + x_1) + x_2 + \sin(x_3)$$

$$\text{increasing concave: } f(\mathbf{x}) = 0.001(-e^{-x_1} + x_1) + x_2 + \sin(x_3)$$

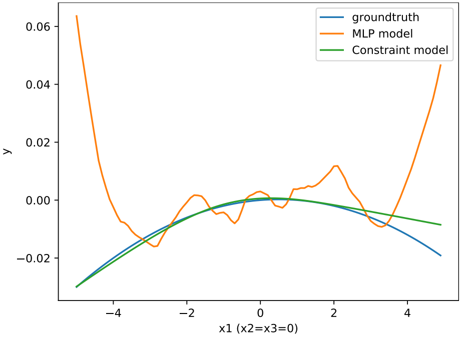
$$\text{general concave: } f(\mathbf{x}) = 0.001(-x_1^2 + x_1) + x_2 + \sin(x_3)$$



monotonic



increasing concave



general concave

[1] code available: https://github.com/J-zin/shape-constraints-on-neural-network

Papers

Incorporating shape constrains via penalized derivatives, Gupta, AAAI 2021

Multidimensional shape constraints, Gupta et al. ICML 2020

Deontological ethics by monotonicity shape constraints, Wang et al. AISTATS 2020

Unconstrained monotonic neural networks, Wehenkel et al. NIPS 2019

Deminishing returns for interpretability and regularization, Gupta et al. NIPS 2018

Deep Lattice Networks, You et al. NIPS 2017

Fast and Flexible Monotonic Functions with Ensembles of Lattices, Canini et al. NIPS 2016