Monotonic Neural Networks

Xing Han

March 2020

1 Introduction

Assume we have a set of monotonically distributed samples $\{x_i, y_i\}_{i=1}^N$ (i.e., $(y_i - y_j)\ .\ (x_i - x_j) > 0$ for $i \neq j$), and wish to obtain monotonic outputs by training on some neural networks g. It's possible that g overfits the data and generates non-monotonic output. We propose a method to solve this problem by combining a linear offset and perform random smoothing to g. Specifically, we have

$$\begin{aligned} f_0(x) &= \mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)}[g(z)] + \omega^\top x + b \\ &= \int C\ .\ g(z)\ .\ e^{-\frac{(z-x)^2}{2\sigma^2}} dz + \omega^\top x + b \end{aligned}$$

then

$$\begin{aligned} \nabla f_0(x) &= C \int \frac{(z-x)}{\sigma^2}\ .\ g(z)\ dz + \omega \\ &= \mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)}\left[\frac{z-x}{\sigma^2}\ .\ g(z)\right] + \omega \\ &\geq -\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}\ .\ \sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)}\left[\frac{(z-x)^2}{\sigma^4}\right]} + \omega \\ &= -\frac{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}}{\sigma} + \omega \geq 0 \\ &\rightarrow \omega \geq \frac{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}}{\sigma}. \end{aligned}$$

Since estimating $\frac{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}}{\sigma}$ not only requires Monte Carlo estimation of the numerator, but the line search of σ affect both numerator and denominator, which is computationally expensive. In this case, we can define a search space Φ and compute $\max_{x\in\Phi} g(x)$ as a simple estimator of the numerator, i.e.,

$$\omega \geq \frac{\left|\max\limits_{x\in\Phi} g(x)\right|}{\sigma} \geq \frac{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}}{\sigma},$$

then σ can be estimated as

$$\sigma = \frac{\left|\max_{x \in \Phi} g(x)\right|}{\omega}. \tag{1}$$

By satisfying Equ (1), we can get $f_0(x) = \mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)}[g(z)] + \omega^\top x + b$ is monotonic.

Monte Carlo Estimation We want to efficiently estimate the term

$$\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2)} g^2(z) = \mathbb{E}_{z\sim\mathrm{Unif}([-a,a])} \frac{\mathcal{N}(z|x,\sigma^2)}{\frac{1}{2a}} g^2(z) = \mathbb{E}(z \sim \mathrm{Unif}([-a,a]))\mathcal{N}(z|x,\sigma^2)(2ag^2(z))$$

So we can uniform sample z from $[-a,a]$, where a is the uniform upper bound of x, and only need to calculate the probability $\mathcal{N}(z|x,\sigma^2)$ with different x and σ.

Assume $\{z_i\}_{i=1}^n$ are uniformly sampled from $\mathrm{Unif}([-a,a])$, we can use the Monte Carlo estimation of the form

$$\frac{1}{n}\sum \mathcal{N}(z_i|x,\sigma^2)(2ag^2(z_i)) = \frac{2a}{\sqrt{2\pi}\sigma}\frac{1}{n}\sum \exp(-\frac{(z_i-x)^2}{2\sigma^2})g^2(z_i)$$

So we can reuse $g^2(z_i)$ for multiple times when changing x and σ, which is desired, as in practice, g can be a large neural network. Inference can take much more time than calculating the Gaussian weights.

2 Experiment

We first perform a simple simulation experiment as following:

1. $X = \{x_i\}_{i=1}^{1000} \sim \mathrm{Unif}[0,20], y = 0.05x + 0.3\sqrt{x} + \epsilon$, where $\epsilon \in \mathcal{N}(0, 0.1)$.
2. $\hat{W} = (X^\top X)^{-1}X^\top y,\ \hat{y} = \hat{W}X$.
3. Fit model g through $\{x_i, y_i - \hat{y}_i\}_{i=1}^{1000}$, where g is a 3-layer feedforward neural network with $Tanh$ non-linearity.
4. Find optimal σ_0 via Equ (1).
5. Get random smoothed output $\mathbb{E}_{z\sim\mathcal{N}(x,\sigma_0^2 I)}[g(z)] + W^\top x$, which is supposed to be monotonic, the result is shown on Figure 1.

Further improvements:

- In this simulation study, the neural network model may itself produce monotonic outputs without linear offset and random smoothing, so we need to design better experiment setting to show our superiority.

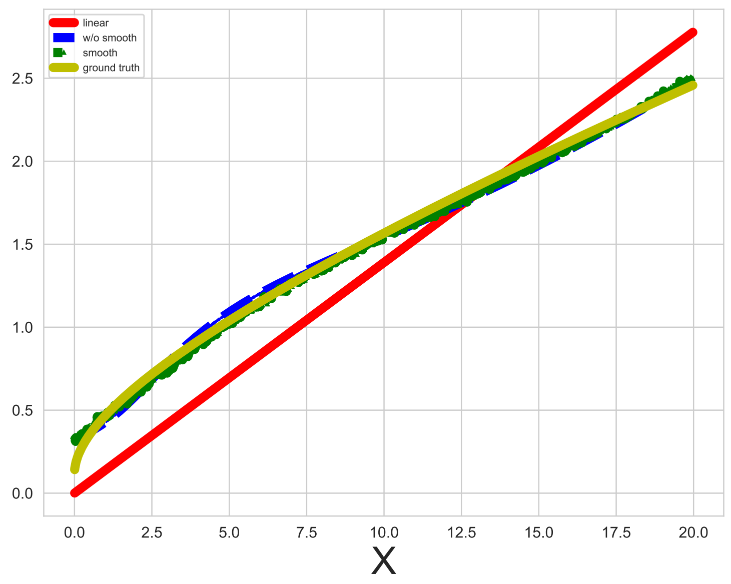

Figure 1: Predicting results for neural network with monotonic constraint (green), ground truth (yellow), linear offset (red), and prediction without random smoothing (blue). We can see the predicting result after random smoothing is monotonic and has better accuracy.

- Need to compare with directly learn a single objective $f_0(x) = g(x) + \omega^\top x + b$ subject to $\omega \geq \frac{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(x,\sigma^2 I)} g^2(z)}}{\sigma}$, versus the above approach, which learns the linear model and neural networks separately. The intuition is, if we jointly train two models, the linear offset may not be optimal.
- Equ (1) is an approximation to simplify the calculation. Make sure if there are other ways to improve this or the original approach performs better.