Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

Towards Sobolev Pruning

Training and pruning surrogate models with sensitivity information

Neil Kichler STCE, RWTH Aachen June 4, 2024

Joint work with:

Sher Afghan (STCE, RWTH Aachen)

Uwe Naumann (STCE, RWTH Aachen)

Motivation

Goal: Find a surrogate model, that is

- accurate
- efficient
- robust.

Motivation

Goal: Find a surrogate model, that is

- accurate
- efficient
- robust.

Typically:

- Fit a (large) neural network
- Prune network into a surrogate model

Motivation

Goal: Find a surrogate model, that is

- accurate
- efficient
- robust.

Typically:

- Fit a (large) neural network
- Prune network into a surrogate model

Shortcoming:

- Surrogate model does not consider sensitivities and uncertainties.
  $\hookrightarrow$ Derivative information may differ drastically.

Motivation

Sensitivity information is often an afterthought, but:

- Captures vital information in many applications
- Crucial in optimization (e.g., Newton's method)
- Helps to learn robust & accurate surrogate models

Motif

Incorporate sensitivity information throughout
the learning and pruning process.

Outline

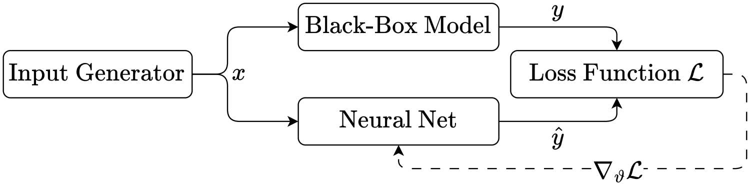
1. Sobolev Training
2. Pruning
3. Case Study
4. Results
5. Conclusion

Setup

In the domain of surrogate modelling with neural networks, let:

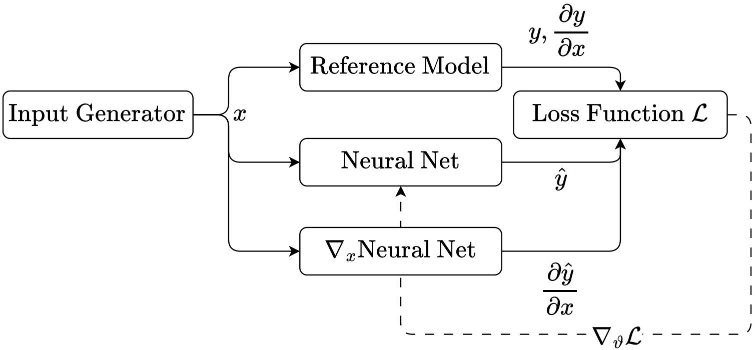
- $\mathcal{N}(\vartheta)$: The surrogate model as a neural network.
  ϑ: The parameters of the neural network.
  f_ϑ: The learned function.
- $\mathcal{S}$: The reference model sampler.
- $\mathcal{L}$: The loss function, e.g. $\mathcal{L} = \|\cdot\|_2^2$.
- (x_i, y_i): An (input, target) sample.

Vanilla Training



Match targets by differentiating the loss and optimize using, e.g., SGD.

Sobolev Training



Match targets and differential targets.

Sobolev Loss

Definition (Sobolev Loss)

Given input $\boldsymbol{x}$, target $\boldsymbol{y}$, predicted output $f_\vartheta(\boldsymbol{x})$, differential target $\nabla_{\boldsymbol{x}}\boldsymbol{y}$, and predicted differential $\nabla_{\boldsymbol{x}} f_\vartheta(\boldsymbol{x})$, the differential loss is defined by:

$$\|\boldsymbol{y} - f_\vartheta(\boldsymbol{x})\|_2^2 + \lambda \|\nabla_{\boldsymbol{x}}\boldsymbol{y} - \nabla_{\boldsymbol{x}} f_\vartheta(\boldsymbol{x})\|_2^2,$$

where $\lambda \in \mathbb{R}_{\geq 0}$ is an added balancing factor.

Sobolev Loss: Interpretation

Srinivas and Fleuret [1] highlight that the Sobolev loss naturally arises when considering pertubations.

Perturbation perspective:

Consider perturbation of input. By Taylor expansion [1]:

$$\mathbb{E}_{\epsilon\sim N(0,\sigma^2)}\left[\sum_{i=1}^{m}(f(\boldsymbol{x}_i+\epsilon)-f_\vartheta(\boldsymbol{x}_i+\epsilon))^2\right]=\sum_{i=1}^{m}(f(\boldsymbol{x}_i)-f_\vartheta(\boldsymbol{x}_i))^2$$
$$+\sigma^2\sum_{i=1}^{m}\|\nabla_{\boldsymbol{x}}f(\boldsymbol{x}_i)-\nabla_{\boldsymbol{x}}f_\vartheta(\boldsymbol{x}_i)\|_2^2+\mathcal{O}(\sigma^4).$$

Sobolev Training

Algorithm Sobolev Training [2].

Require: The following inputs must all be initialized.

- Surrogate model $\mathcal{N}(\vartheta)$ with function f_ϑ and parameters ϑ
- Reference model $\mathcal{S}$
- Optimizer G

while ϑ not converged **do**

$\{(\boldsymbol{x}_i, \boldsymbol{y}_i, \nabla_{\boldsymbol{x}} \boldsymbol{y}_i)\}_{i=1}^{m} \sim \mathcal{S}$ ▷ Sample training data

$\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_\vartheta \sum_{i=1}^{m} \mathcal{L}(f_\vartheta(\boldsymbol{x}_i), \boldsymbol{y}_i) + \lambda \mathcal{L}(\nabla_{\boldsymbol{x}} f_\vartheta(\boldsymbol{x}_i), \nabla_{\boldsymbol{x}} \boldsymbol{y}_i)$

$\vartheta \leftarrow G(\vartheta, \hat{\boldsymbol{g}})$ ▷ Update parameters

end while

return $\mathcal{N}$

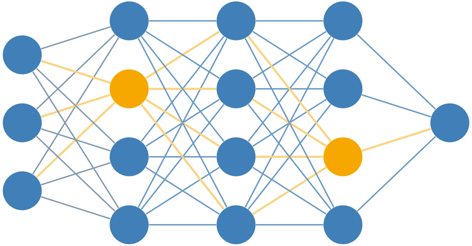
How large should the surrogate model be?

~~How large should the surrogate model be?~~
How small can the surrogate model get?

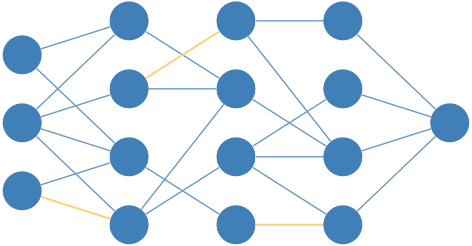
Pruning

Goal

Prune the surrogate model to increase the computational efficiency while retaining accuracy.



Dense Pruning.



Sparse Pruning.

Pruning

Goal

Prune the surrogate model to increase the computational efficiency while retaining accuracy.

Why not start with Sparse Training?

Dynamic Sparse Training (e.g., SET [3], RIGL [4]) works, but:

- are not designed for modern architectures $\rightarrow$ requires masking.
- still requires reasonable starting size guess.
- worse performance, in practice, compared to dense2sparse training.

Pruning

Typically

Magnitude Pruning: $|w|$
Salience Pruning: $|\frac{\partial \mathcal{L}}{\partial w} w|$
...

Pruning

Typically

Magnitude Pruning: $|w|$
Salience Pruning: $|\frac{\partial \mathcal{L}}{\partial w} w|$
$\dots$

Downsides?

- must iterate over training data.
- sensitivity information?
- no global perspective.

Pruning

Can we get a global perspective on weight importance?

Interval arithmetic!

Interval Arithmetic

Fundamentals:

- Considers variables to be inside a fixed range, a trust region.
- Replace $x \in \mathbb{R}$ with $[x] \in \mathbb{IR}$.
- $[x] = [\underline{x}, \overline{x}]$ if $\{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$.
- $f([x]) := \{f(x) \mid x \in [x]\}$.

Fundamental Theorem of Interval Arithmetic

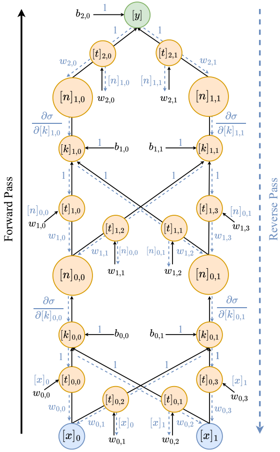
A function $\boldsymbol{f}$ over an interval input box $X = ([x]_0, \cdots, [x]_n)$ is guaranteed to enclose the range of $\boldsymbol{f}$ over those inputs, i.e., $\mathrm{range}(\boldsymbol{f}) \subseteq \boldsymbol{f}(X)$ (Moore et al.).

Interval Adjoints

Algorithmic Differentation (AD) [6] naturally applies to interval arithmetic [7].

AD on Interval Arithmetic

- Apply AD as usual.
- Replace all operations of the source transformed code with the interval arithmetic equivalent.



Interval Adjoint Significance Analysis

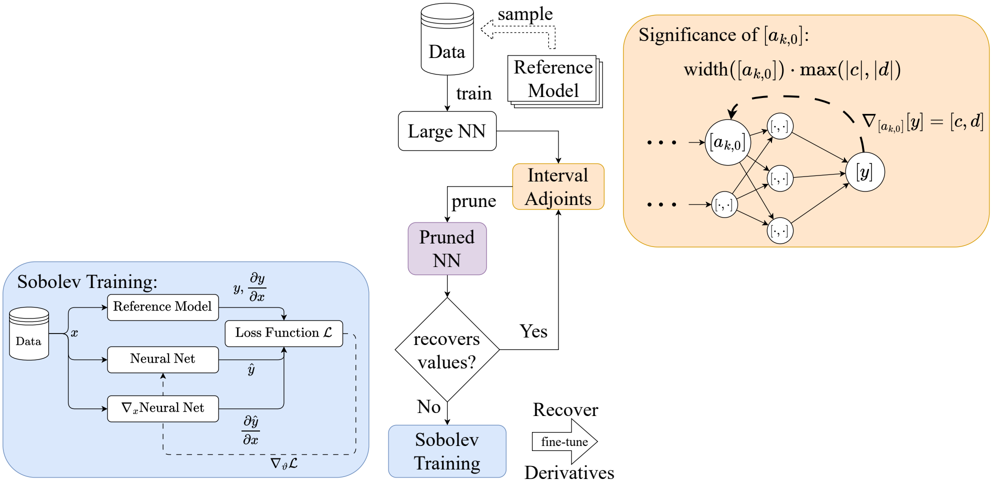
Significance Measure

$$S_{[y]}([n]_{l,i}) = \mathrm{width}([n]_{l,i}) \cdot \max\left(\left|\nabla_{[n]_{l,i}}[y]\right|\right),$$

where:

- $[y]$: interval output
- $\mathrm{width}([x]) = \overline{x} - \underline{x}$
- l: hidden layer index
- i: node index in given layer

General framework



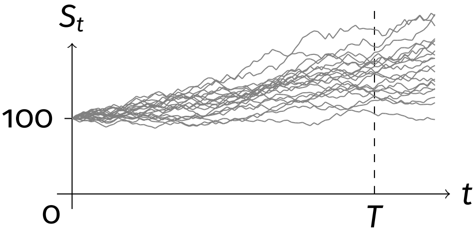
Case Study: Option pricing

Consider the SDE:

$$dS_t = a(S,t)\,dt + b(S,t)\,dW_t,$$

where:

- dW_t is a Wiener process
- $a : \mathbb{R} \times [0,T] \to \mathbb{R}$, the drift
- $b : \mathbb{R} \times [0,T] \to \mathbb{R}$, the vola

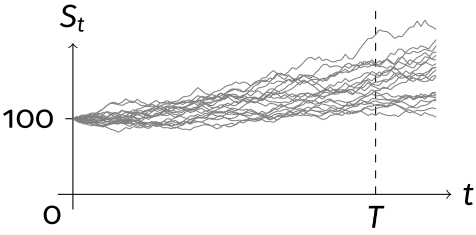


Sample paths of SDE.

Case Study: Option pricing

Consider the SDE:

$$dS_t = a(S, t)\,dt + b(S, t)\,dW_t,$$



Sample paths of SDE.

Interested in the price:

$$V = \mathbb{E}[\nu(S_T, K)],$$

with:

- S_T the price at maturity T
- K the strike price
- $\nu(S_T, K)$ the payoff function

Bachelier

The Bachelier model can be described as a SDE:

$$\mathrm{d}S_t = \mu S_t dt + \sigma \,\mathrm{d}W_t,$$

where:

- t, the time index.
- μ, the constant drift for the interest rate.
- σ, the constant volatility.
- S_t, the underlying asset price at time t.
- $\mathrm{d}W_t$, a Wiener process, i.e. Brownian motion.

Gaussian Basket

Definition (Basket)

A Basket $\mathbf{S}_t \in \mathbb{R}^m$ of m securities $S_t^{[0]}, \ldots, S_t^{[m]}$ has price:

$$\mathbf{S}_t = \sum_{i=0}^{m} \omega_i S_t^{[i]}, \quad \sum_{i=0}^{m} \omega_i = 1,$$

where ω_i is the weight associated with the ith security.

$\Rightarrow$ Directly applicable to the Bachelier model.

Surrogate Objective

Given

$\mathbf{S}_0$: initial spot price from basket

Find

V : option price (Value)

$\frac{\partial V}{\partial \mathbf{S}_0}$: 1st-order price sensitivity (Delta)

$\frac{\partial^2 V}{\partial \mathbf{S}_0^2}$: 2nd-order price sensitivity (Gamma)

$\Rightarrow$ Solved using Least-Squares Monte Carlo.

Least-Squares Monte Carlo

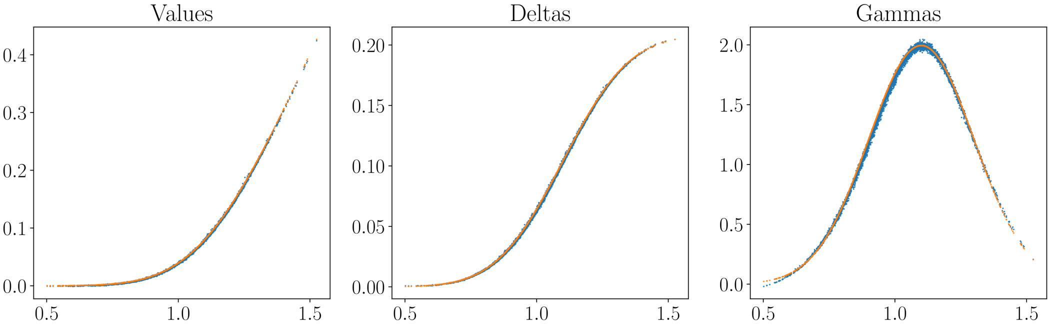
Least-Squares Monte Carlo method is equivalent to optimizing

$$\vartheta^* = \arg\min_{\vartheta} \mathbb{E}_{(\theta,\boldsymbol{z})\sim\Theta_{\text{in}}\times\mathcal{Z}}\left[\|\nu(g(\theta,\boldsymbol{z})) - f_\vartheta(\theta)\|_2^2\right],$$

where:

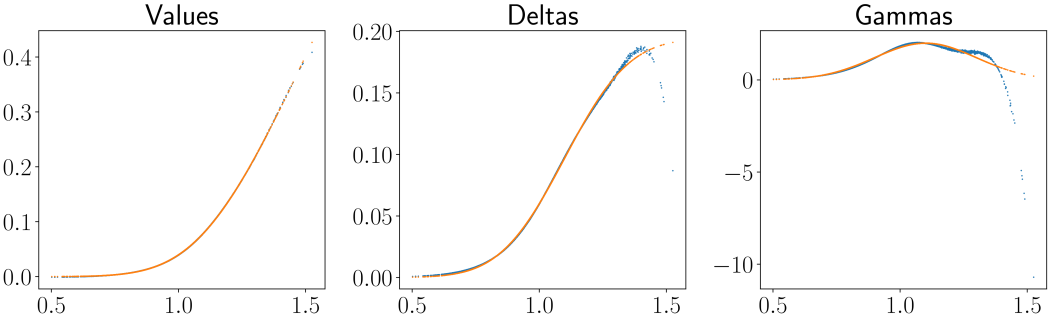
- f_ϑ is the fitted curve with coefficients ϑ
- random input parameters $\theta \sim \Theta_{\text{in}}$ (here: $\theta = \{\mathbf{S}_0\}$)
- random path noise samples $\boldsymbol{z} \sim \mathcal{Z}$
- payoff function ν.

Regression using Neural Networks



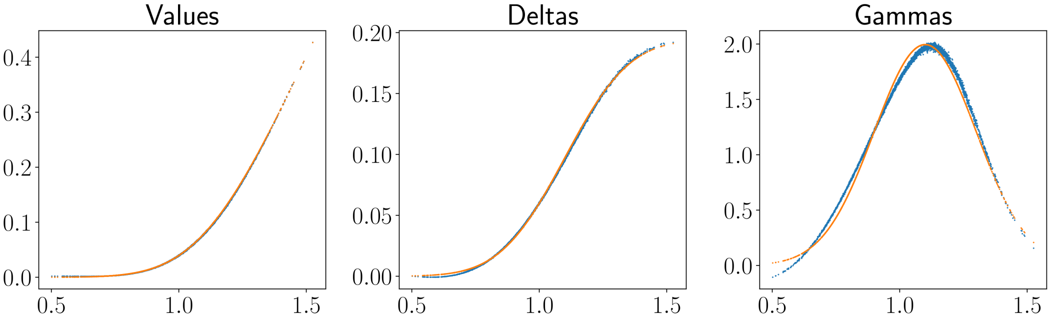
Baseline results (normalized) of Vanilla ML using a basic Multi-Layer Perceptron (MLP).

After Interval Adjoint Significance Pruning



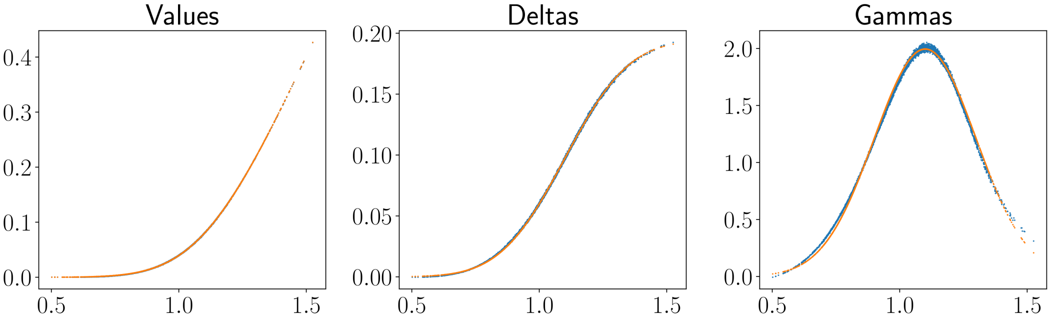
Results of pruned model (and vanilla ML fine-tuning).

After Sobolev fine-tuning



Results after Sobolev fine-tuning on derivative samples from learned NN.

After Sobolev fine-tuning on reference



Results after Sobolev fine-tuning on derivative samples from Bachelier reference model.

Results: Overview

R^2 score of surrogate models for a Bachelier modelled basket option (7 dimensions).

Predict	Oversized NN	Pruned NN	Sobolev fine-tuning NN Data	Sobolev fine-tuning Bachelier
Values	0.999545	0.999296	0.999805	0.999962
Deltas	0.998700	0.996718	0.999479	0.999863
Gammas	0.997033	0.902470	0.987393	0.997374

Limitations & Future Work

Things to keep in mind

It requires:

- AD for interval arithmetic (no built-in support in ML libraries).
- Access to intermediate local partial derivatives.
- Derivative information from reference model

  $\hookrightarrow$ need access to source.

    - Expensive Jacobians? Approximate by sampling `vjps`.

Going beyond

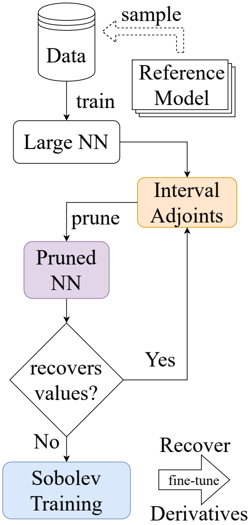
- Add second-order sensitiviy information?

Conclusion

Add sensitivity information by:

- Pruning previously learned network with Interval Adjoint Significance Analysis.
- Using Sobolev Training to improve accuracy and retain sensitivity information.

Paper & Code: github.com/neilkichler/sobolev-pruning



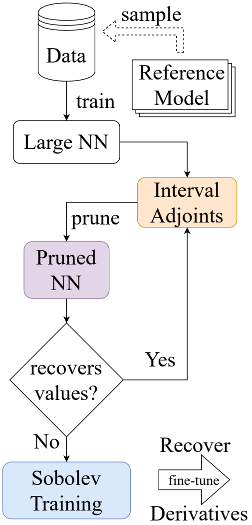
Conclusion

Thank you for your attention!

Add sensitivity information by:

- Pruning previously learned network with Interval Adjoint Significance Analysis.
- Using Sobolev Training to improve accuracy and retain sensitivity information.

Paper & Code: github.com/neilkichler/sobolev-pruning



References I

[1] Suraj Srinivas and Francois Fleuret. “Knowledge Transfer with Jacobian Matching”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, July 2018, pp. 4723–4731. arXiv: 1803.00443.

[2] Wojciech M Czarnecki et al. “Sobolev training for neural networks”. In: vol. 30. 2017. arXiv: 1706.04859.

[3] Decebal Constantin Mocanu et al. “Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science”. In: *Nature communications* 9.1 (2018), p. 2383.

[4] Utku Evci et al. “Rigging the Lottery: Making All Tickets Winners”. In: *Proceedings of Machine Learning and Systems 2020*. 2020, pp. 471–481.

References II

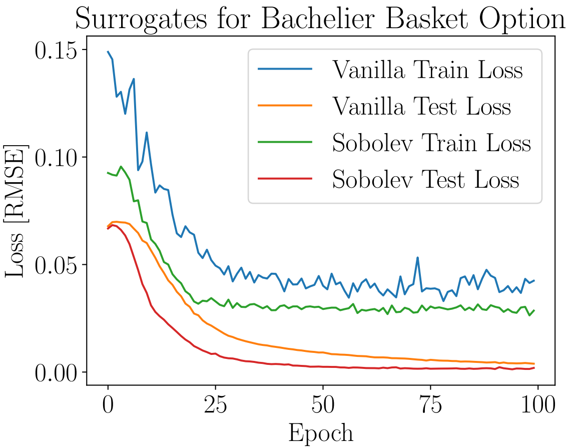
[5] Ramon E. Moore et al. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009. DOI: 10.1137/1.9780898717716.

[6] Uwe Naumann. *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*. Society for Industrial and Applied Mathematics (SIAM), 2012. DOI: 10.1137/1.9781611972078.

[7] Vassilis Vassiliadis et al. “Towards automatic significance analysis for approximate computing”. In: *Proceedings of the 2016 International Symposium on Code Generation and Optimization*. CGO ’16. Barcelona, Spain: Association for Computing Machinery, 2016, pp. 182–193. ISBN: 9781450337786. DOI: 10.1145/2854038.2854058.

Backup Slides

Comparison of Loss Curves



Second-order sensitivity information

Hessian too expensive? $\rightarrow$ sample directions.

Random directions

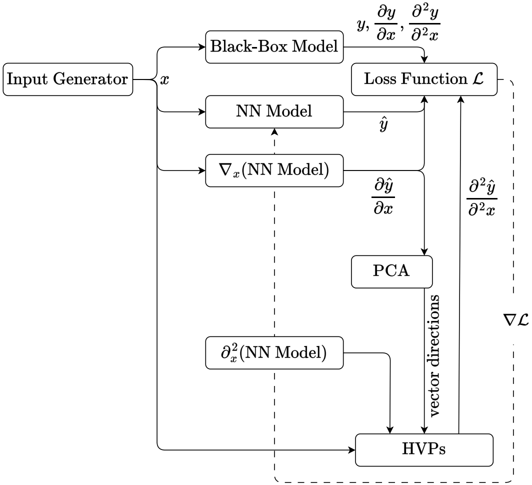
Draw random vectors $\boldsymbol{v}$, s.t. $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\intercal] = \boldsymbol{I}$.

$$\Rightarrow \mathbb{E}\left[\boldsymbol{H}\boldsymbol{v}\boldsymbol{v}^\intercal\right] = \boldsymbol{H}\mathbb{E}\left[\boldsymbol{v}\boldsymbol{v}^\intercal\right] = \boldsymbol{H}.$$

E.g., $N(\mu = 0, \Sigma = \boldsymbol{I})$.

PCA directions

- Take principal components.
- Further reduction by taking k-most significant components.



Pathwise Sensitivities

Fix some random sample path $z \sim \mathcal{Z}$ and input parameters $\boldsymbol{\theta}$.

We have unbiased estimates of, e.g., pathwise deltas, if:

$$\mathbb{E}_{z\sim\mathcal{Z}}\left[\frac{\partial}{\partial S_0}\nu(g(\boldsymbol{\theta},z))\right] = \frac{\partial}{\partial S_0}\mathbb{E}_{z\sim\mathcal{Z}}\Big[\nu(g(\boldsymbol{\theta},z))\Big], \tag{1}$$

i.e. we can interchange the expectation with the derivative operator.

Practical Conditions for (1):

- Payoff ν must be differentiable almost everywhere.
- Payoff ν is Lipschitz continuous.

Smoothing

We perform smoothing between function $f_1 : \mathbb{R} \to \mathbb{R}$ and $f_2 : \mathbb{R} \to \mathbb{R}$ via $\tilde{f} : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ defined as

$$\tilde{f}(x, p, w) = (1 - \sigma(x, p, w))f_1(x) + \sigma(x, p, w)f_2(x),$$

where

$$\sigma(x, p, w) = \frac{1}{1 + e^{-\frac{x-p}{w}}},$$

and p is the position to change between the two functions and w the width of the smoothing.

Smoothing

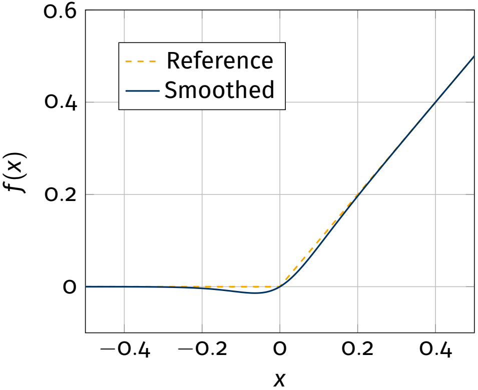
For $\nu = (\cdot)^+$:

Split function into $\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$.

We obtain:

$$\tilde{\nu}(x, w) = \frac{x}{1 + e^{-\frac{x}{w}}}.$$

Equivalent to SiLU if $p = 0, w = 1$.



Smoothing $(\cdot)^+$, with $p = 0, w = 0.05$.