Training a Two-Layer ReLU Network on a 1D Dataset

Due: Monday 8 December, 22:00 (by email)

Submission. Email a **single** `.zip` containing (i) a PDF report (max 2 pages), (ii) a single notebook `ReLU.ipynb`, and (iii) the provided `data/` folder (unchanged) so the notebook runs end-to-end. Send to `scott.pesme@inria.fr` with subject: *"Homework 1 – [Name 1] and [Name 2 (if applicable)]"*. Work alone or in pairs.

Dataset

We provide 1D train and test sets (x_i, y_i) (see Figure 1). Train only on the training set; you may use the test test to compute a test loss.

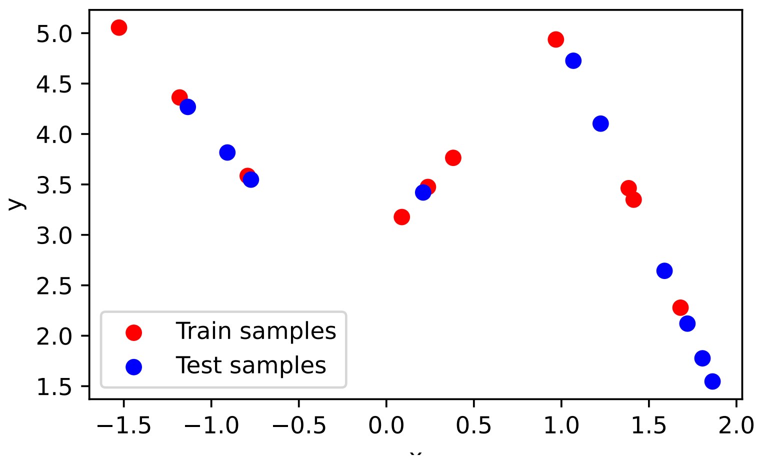

Figure 1: Provided 1D train (red) and test (blue) samples.

Model and Objective

We consider a two-layer ReLU network of width m for scalar input $x \in \mathbb{R}$:

$$f_w(x) = \frac{1}{m} \sum_{j=1}^{m} a_j \max\{0,\ b_j x + c_j\}, \quad w = \big((a_j, b_j, c_j)\big)_{j=1}^{m}. \tag{1}$$

The parameters $w \in \mathbb{R}^{3m}$ are called the *weights* of the neural network. More specifically, $(a_j)_{j=1}^m$ are the outer-layer weights and $(b_j, c_j)_{j=1}^m$ the inner-layer weights.

Given the provided training data $\{(x_i, y_i)\}_{i=1}^n$, we will train the network by minimising the empirical square loss:

$$L(w) = \frac{1}{2n} \sum_{i=1}^{n} \big(f_w(x_i) - y_i\big)^2. \tag{2}$$

You may optionally add *weight decay* (an ℓ_2 penalty on the parameters), e.g. minimise the penalised loss L_λ:

$$L_\lambda(w) \;=\; L(w) + \frac{\lambda}{2}\,\|w\|_2^2, \qquad \lambda \geq 0. \tag{3}$$

Theoretical Questions (to include in the report)

Inside the report, the following questions should be tackled:

Q1. Is the ReLU function $z \in \mathbb{R} \mapsto \max\{0, z\}$ differentiable everywhere? If not, how can one implement gradient-based methods in practice if a non-differentiable point is hit?

Q2. For the given 1D training dataset (plotted Figure 1) and a width $m \geq 1$, does there exist weights $w^\star$ that interpolate the training data, i.e. such that $f_{w^\star}(x_i) = y_i$ for all $i \in [n]$? If yes, provide one explicit set of parameters achieving interpolation. Is such an interpolating $w^\star$ unique?

Q3. Is the training loss L convex? What kind of point can gradient descent be expected to converge to?

Q4. It is common to initialise the network weights randomly as follows: the outer-layer weights are drawn as $a_j \sim \mathcal{N}(0, \alpha_{\text{outer}})$ for $j = 1, \ldots, m$, where $\alpha_{\text{outer}} \geq 0$ is called the *initialisation scale of the outer layer*. Similarly, the inner-layer parameters are drawn as $b_j, c_j \sim \mathcal{N}(0, \alpha_{\text{inner}})$ for $j = 1, \ldots, m$, where $\alpha_{\text{inner}} \geq 0$ denotes the *initialisation scale of the inner layer*. What happens if we pick $\alpha_{\text{inner}} = \alpha_{\text{outer}} = 0$?

Experiments

Implement both **gradient descent (GD)** and **stochastic gradient descent (SGD)** to train the network. Your entire code *must* be in a single notebook, `ReLU.ipynb`, and implemented from scratch (*e.g.*, you may use `numpy` and `matplotlib`, but *not* machine-learning or autodiff libraries such as PyTorch, TensorFlow, or JAX). I will run the notebook with *Run All* to reproduce the figures in your report.

The goal of this homework is to investigate and discuss how different design and algorithmic choices affect training.

Below are examples of aspects you can explore (and you are encouraged to vary several at once to observe interesting phenomena). You are free to choose which to investigate: you may follow items from this list, skip some, or explore additional ideas of your own. For each choice you explore, clearly state what you did and show representative plots (e.g. loss curves, predictions vs. data, or any other quantities you find relevant). Plots must be well-labelled (axes, titles, legends), and every experiment must be clearly explained. Comment on what you observe: what happens, and how can it be interpreted?
Important: The aim is **not** to achieve the lowest possible test loss, but to provide a clear and well-structured analysis of your observations.

Here is a non-exhaustive list of possible directions to explore:

a) **GD vs. SGD.** Compare their behaviour.

b) **Initialisation scale.** Vary the initial scales α_{inner} and α_{outer} and study how this affects training and the learnt function.

c) **Freezing the inner layer.** Instead of training all the weights in both layers, initialise the network but train only the outer layer. What do you observe? How do you interpret this?

d) **Width of the network.** Explore the impact of the network width m on the training dynamics and the learned function.

e) **Step size (learning rate).** Explore various choices of step sizes and describe your observations.

f) **Weight decay.** Observe and comment the impact of weight decay on the training dynamics and on the final predictor.

Report Requirements

- Maximum of 2 pages (single- or double-column, as preferred). LaTeX is preferred, but not mandatory.
- Concise answers to the theoretical questions.
- Clear experimental set-ups (i.e. a description of what you did).
- Plots with readable labels and legends, and captions explaining the main takeaways.
- A discussion of the observed behaviours.
- Reproducibility: every figure in the report must be generated by running your `ReLU.ipynb` end-to-end (use a fixed random seed so that the same plots can be reproduced exactly).

What to Submit

1. A PDF report (maximum 2 pages).
2. A single notebook `ReLU.ipynb` that can be executed with "Run All" to regenerate all figures in the report.
3. The provided `data/` folder, so that the code runs correctly.

Collaboration. You may work alone or in groups of two. Write the names clearly on the first page of the report.

Submission details. Email before **Monday 8 December, 22:00** to `scott.pesme@inria.fr` with subject
"Homework 1 – [Name 1] and [Name 2]". Attach a single `.zip` containing the PDF and `ReLU.ipynb` and the given data folder.