Chair of Decision Sciences and Systems
TUM School of Computation, Information and Technology
Technical University of Munich

Business Analytics & Machine Learning
Homework sheet 12: SGD and Neural Networks – Solution

Prof. Dr. Martin Bichler, Prof. Dr. Jalal Etesami
Julius Durmann, Markus Ewert, Johannes Knörr, Yutong Chao
January 22, 2024

Exercise H12.1 *Training Neural Network with ReLU Activation*

We consider an artificial neural network with ReLU activations:

$$f_W : \mathbb{R}^{d_0} \to \mathbb{R},\ f_W(x) = W_K\sigma(W_{K-1}\sigma(W_{K-2}...\sigma(W_1x))),$$

where $\sigma(z) = \max\{0, z\}$ is the ReLU activation function. The activation function is applied coordinate-wise. W_i is the weight matrix and we denote $W = (W_1, W_2, ..., W_K)$. Let $\{(x_i, y_i)\}_{i=1}^n$ be a dataset and consider the training loss

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

1. Determine whether the following statement is true or false.
   - $L(W)$ is not a differentiable function everywhere in its domain.
   - $f_W(ax) = af_W(x)$ for all $a \in \mathbb{R}$.
   - $\sigma(x)$ is a convex function.
   - $L(W)$ might be non-convex for some dataset $\{(x_i, y_i)\}_{i=1}^n$.

Solution

- True. Since the ReLU activation function is non-differentiable at $x = 0$.
- False. Since $sigma(-1 * 1) \neq -1\sigma(1)$.
- True. This can be proved from the definition.
- True. For dataset $\{(1, 1)\}$, the function $(w_2\sigma(w_1) - 1)^2$ is nonconvex. One can check by applying $(w_1, w_2) = (-1, 1)$ and $(w_1, w_2) = (1, 1)$.

Exercise H12.2 *Implementing a neural network*

This is an optional, advanced programming exercise. You don't need to be able to solve it yourself, but you should nevertheless study the provided solution file to familiarize yourself with the workings of NNs in practice.

a) Implement (in Python via numpy and pytorch) the forward and backward pass for the neural network from Tutorial Sheet 12. Train the neural network on the dataset from Exercise 12.2 c) by performing 100 forward and backward passes and updating the parameters using gradient descent with a learning rate of $\alpha = 1$.

Solution

The solution to this exercise can be found in the file *solution_nn_gradient_descent_numpy* and *solution_-nn_gradient_descent_pytorch* on Moodle.