Homework 3

Zhipeng Huang

03-03-2024

Due date: 24th of March, 2024.
Submission via email: z.huang1@uu.nl.
Submissions should consist of a *short* `.pdf` report of your findings, and an offline, compilable copy of the Jupyter notebook.

Ex. 1 Suppose you have a data set, consisting of data points $(x_i, y_i), i = 1, \ldots, 5000$. The following mapping holds

$$y(x_i) = \tan(x_i), \quad 0 \leq x_i \leq 1.55,$$

where x_i are equally distributed over the domain $[0, 1.55]$. Randomly split the data set into two parts, training (80%) and test (20%). Fix the random seed to make sure the result can be reproduced. Part of the Python code presented in lecture 4 (Seminar_Lecture_4.ipynb) may be useful in finishing this homework.

(a) Train a fully-connected neural network (here one hidden layers with 10 neurons) within Pytorch, and evaluate the trained model $MSE = \sum_i (\hat{y}_i - y_i)^2$, where $\hat{y} = ANN(x)$.

(b) Grid search: Tune the number of neurons in (a), assuming there are 6 candidates, [10, 25, 40, 50, 70, 85], in the search space and using 4-fold cross-validation for model selection. Select the best model and re-train it on the entire training data. Hint: consider early stopping during training.

(c) Automatic differentiation: with the already trained best model in (b), compute the gradient $\hat{y}' = \frac{d\hat{y}}{dx}$ over the test dataset. Compare $\hat{y}'$ with the true gradient $y' = \frac{1}{\cos^2(x)}$ and plot them in a figure. Do they fit well?

(d) Wide vs Deep neural networks: Vary the number of hidden layers and nodes, according to Table 1. For example, the combination $(2, 14)$ in Table 1 means that there are 2 fully-connected hidden layers and each of them consists of 14 neurons, so the structure of ANN is 1→14 →14→1 for input→hidden→hidden→output layer. Follow the steps in (a) and train a neural network for each combination in Table 1.

Table 1: Number of hidden layers and neurons.

		Neurons		
Hidden layers	1	25	50	85
	2	7	10	14
	3	5	7	10
	4	6	8	10

(e) Analyze the results in (d). Does the ANN with four hidden layers perform better? if so, could you explain it? if not, could you improve it?

(f) Computational Graph: Residual Neural Networks (ResNets) allow the gradient information to pass through the intermediate layers, by directing the output of the earlier layers to the input of deeper layers. As shown in Figure 1, skip connections

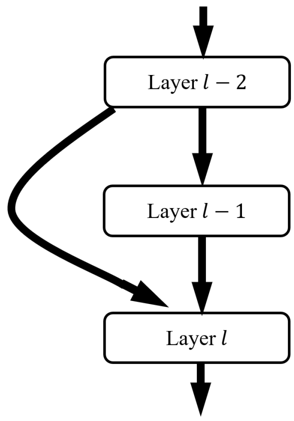

Figure 1: The structure of ResNets.

are introduced into the network structure, as follows,

$$\mathbf{z}^{(\ell)} = \varphi^{(\ell)}(\mathbf{a}^{(\ell)} + \mathbf{z}^{(\ell-2)}) = \varphi^{(\ell)}(W^{(\ell)} \cdot \mathbf{z}^{(\ell-1)} + \mathbf{b}^{(\ell)} + \mathbf{z}^{(\ell-2)}),$$

where $\varphi()$ represents the activation function, W and $\mathbf{b}$ being weights and biases respectively, ℓ being the hidden layer. Please build a ResNet with Pytorch, then train it to approximate $y = \tan(x), 0 \leq x \leq 1.55$. For example, there is one input layer, five fully connected hidden layers (5 neurons per layer), and one output layer.