ASSIGNMENT 2: REGRESSION WITH NEURAL NETWORKS

MA 590 ST: INTRODUCTION TO SCIENTIFIC MACHINE LEARNING

DUE ON OCTOBER 02

Given data pairs $(\mathsf{x}_i, \mathsf{y}_i)_{i=1}^m$, where $\mathsf{y}_\mathsf{i} = \Gamma(\mathsf{x}_i)$ (noise free).

(1) (1D, discontinuous functions) $\Gamma(x)$ Heaviside function on $x \in [-2, 2]$.

(2) (1D) $\Gamma(x) = \sqrt{2-x}$ on $[0, 2]$. This function is not differentiable at $x = 2$ but smooth at any $x < 2$. It is Holder continuous on $[0, 2]$ with exponent 1/2:

$$|f(x) - f(y)| \leq \sqrt{|x-y|},\ x, y \in [0, 2].$$

(3) (2D, multi-modal) $\mathsf{x} = (x, y) \in [-1, 1]^2$.

$$\Gamma(\mathsf{x}) = \Gamma(x, y) = \exp(-k_1(x-0.5)^2 - (y-0.5)^2) + \exp(-k_2(x+0.5)^2 - (y+0.5)^2)$$

Take $k_1 = k_2 = 1$ and $k_1 = k_2 = 10$ (this second one is optional).

(4) Rosebrock function

$$f(\mathsf{x}) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (1-x_i)^2] \quad \text{where} \quad \mathsf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d.$$

Take $d = 2$ and consider the domain $[-2, 2] \times [-2, 2]$.

(5) (high-dimensions, Optional) $\mathsf{x} \in [0, 1]^d$ (We can take $d = 1, 2, \cdots, 8$. Try these different strategies and see how the neural networks keep the accuracy level the same when d increases.)

$$\frac{1}{(2\pi)^{d/2}} \exp(-\sum_{i=1}^{d} x_i^2)$$

Here, you may need to increase the number of layers when d is large.

Deliverables:

(1) plots of the functions and the neural network approximations (in separate pictures, can be put side-by-side), and the absolute errors, and the relative errors. *For each function, list your choices as in the table.* The relative error can be computed as absolute error/(function value $+10^{-10}$) to avoid zeros in the denominators.

(2) your code for the problems

(3) Explain why you chose the settings if you use different ones than those in the table.

(4) Make sure you comment according to your results on which setting performs the best for each function.

functions	choice	Comments
training points	one type of QMC points 100 in 1D and 100×100 in 2D	
test points	100^d ($d = 1, 2$) points are different from training ones	
NN architecture	1) 2 layers, width 40 or 60 2) 4 layers, width 20 or 30	For a fair comparison, we need same number of neurons
activation function	Relu and tanh (use the same activation in one NN)	fundamental ones
Initialization	your choice	
optimizers	ADAM (or AMSgrad) with learning rate $1e-3$	
	(optional) use a scheduler epochs, stopping condition	

Remark 1. The purpose of this exercise is to gain insights into dealing with different data sets: how to pick activation functions and the network architectures. You may also try different optimizers and see if the results are close.

Hints

1. To test your code, you may want to test simple functions such as

$$\Gamma(x) = \sin(x), \Gamma(\boldsymbol{x}) = x^2 + y^2$$

on the unit domain, $x \in [0, 1]$ or $\boldsymbol{x} \in [0, 1] \times [0, 1]$ The functions are simple enough and have been tested many times using feedforward neural networks.

2. You may also use the code (Files/code)

```
mlp_regression_optax.ipynb,   mlp_regression_optax_batch.ipynb,
```

For the quasi-Monte Carlo methods, you may use the code (see Modules/sampling or Files/code)

```
qmc_call.ipynb
```

We will go over these methods next class. You can refer to twodfunction_plot.ipynb for plotting two dimensional functions.

3. Use two layer neural networks with a large width vs deep neural networks using relatively small width. When the numbers of neurons are at the same magnitude, deep neural networks usually performs better. Of course, the performance depends on the trainings (optimizers and hyperparameters).

4. Always normalize your data. In some problems, we need to normalize both the argument x and the function values.

5. (Frequency principle) The neural network initially fits the low frequency parts of the function, and only later in training is it able to capture the small scale details that correspond to higher frequencies. That's why we often use 10K or 20K epochs to check the history of the loss function.