Assignment #1 (Due Dec 20, 2022)

Question #1

Given a *fully connected Neural Network* (MLP) as follows:

a. Input $[x1\ ,x2\ ,\cdots x10\ ]$

b. k-hidden fully connected layers (exclude output layer) with bias and of 24 nodes in each layer

c. Output(Prediction): 1 node;

d. Use the ReLU activation function for all hidden layers.

You are required to do the following:

1. Prepare your data set of 100 data points, by sampling xi ~ N(0,1), y = mean(x1,x2,…x10), consider the mean square error loss function.
2. Build this network using PyTorch with arbitrary k-hidden layers. Use autograd to find the gradient. Plot training curves and testing curves.
3. Build this network from scratch without using PyTorch nn.Linear and ReLU function, without using autograd function. Plot training curves and testing curves. Compare your answers.

Question #2

Generate 100 data points for validation with x~Uniform[0,1] and y = x + e, where e~N(0,0.1). For the training data points, use the following 4 points: [0.1,0.2], [0.2,0.15], [0.3,0.4], [0.4,0.33].

1. Use the mean square error loss.
2. Perform polynomial fit on the 4 training data points with polynomial of degree 0, 1, 2, 3, . . . , 50.
3. Repeat the above steps 10 times with different initialization for your fitting programme to generate different fitting outcomes.
4. Calculate the training error and validation error for each degree of polynomial fit. Since you repeat your steps 10 times, calculate the standard error of your loss.
5. Make a plot of
    a. Training loss versus degree of polynomial
    b. Validation loss versus degree of polynomial
6. Explain your results.

Question #3

Figure shows a computational tree in the forward pass. Three variables are initialized and they are used to compute the output. This tree is generated by an unknown code. Your task is the write a code that generates this tree. For checking if your code is correct, you may use

the following function:

```
from torchviz import make_dot

def print_compute_tree(name,node):
    dot = make_dot(node)
    dot.render(name)

if __name__==’__main__’:
    x = torch.randn([1, 1, 10], requires_grad=True)
    print_compute_tree(‘tree_filename’,x)
```