Github location: <https://github.com/wany0011/BS6207/upload/main/Assignment_1>

**Question #1**: As per instructions, two Linear Networks are created, being of identical structure, please find the respective python files.

1. my\_autograd.ipynb: flexible in configurable ways of constructing the neural network based on parameter of **K and D,** with both forward and backward backpropagation implemented
2. torch\_autograd.ipynb : the structure is hard-coded with no fancy programming skills, since the purpose of coding the neural network is to cross-check the results

# Number of hidden layers

K=10

# Number of input nodes

d=10

the structure is derived as:

# N is batch size; D\_in is input dimension;

# H is hidden dimension; D\_out is output dimension.

N, D\_in, H, D\_out = 1, d, [2\*d+1,K], 1

Training process is built into individual neural network, to prove convergence over iterations. Description of backpropagation is ignored here since many online resources are readily available.

For the above individual neural network, Gradients for weights and bias are generated at each iteration of training, under 2 scenarios:

1. Weights are initialized with ALL Zero and bias ALL One: In such cases, **the gradients are visually identical** (refer to DAT files)across the 2 neural networks over throughout iterations of training.

Gradient of “Weight and Bias” at Layer #5 of Iteration #9

Table

Description automatically generated# torch\_auto # my\_autograd

Table

Description automatically generated with low confidence

1. Randomized Weights and Bias at starting point. The tensor seed torch.manual\_seed(2020) is used along with torch.use\_deterministic\_algorithms(True), and all random samplings follow the same sequences for the 2 neural networks. However, **the initialized weights are still not the same** across the 2 neural networks, that lead to differences in gradients. (If I copy over the initial weights & bias for one to another, the gradient of both networks will be identical. – this step is skipped)

Observation:

Starting from the same initialized weights and bias, ALL the subsequent gradient generated are identical across the 2 neural networks over 10 iterations of training, at the same learning rate of 1e-4. Notably, when all weights initialized with zero, the gradients start to jump out of Zero backward starting from the last layer gradually over the iterations, that could be explained by the mechanism of backpropagation.

Please find the datafile folders as follows:

1. [Init\_0\_Weight\_1\_Bias] : all weights are initialized as Zero ,and 1 for all bias.

Chart, line chart

Description automatically generated

1. [Random\_Weight\_Bias]: all weights and bias are uniformly sampled from [0,1].

Shape

Description automatically generated

**Question #2**

It shed lights on how backpropagation could flow back with weights updated along the DAG originated from Lost function.



