Homework 02-1: due 2022/03/31 14:10 (30%)

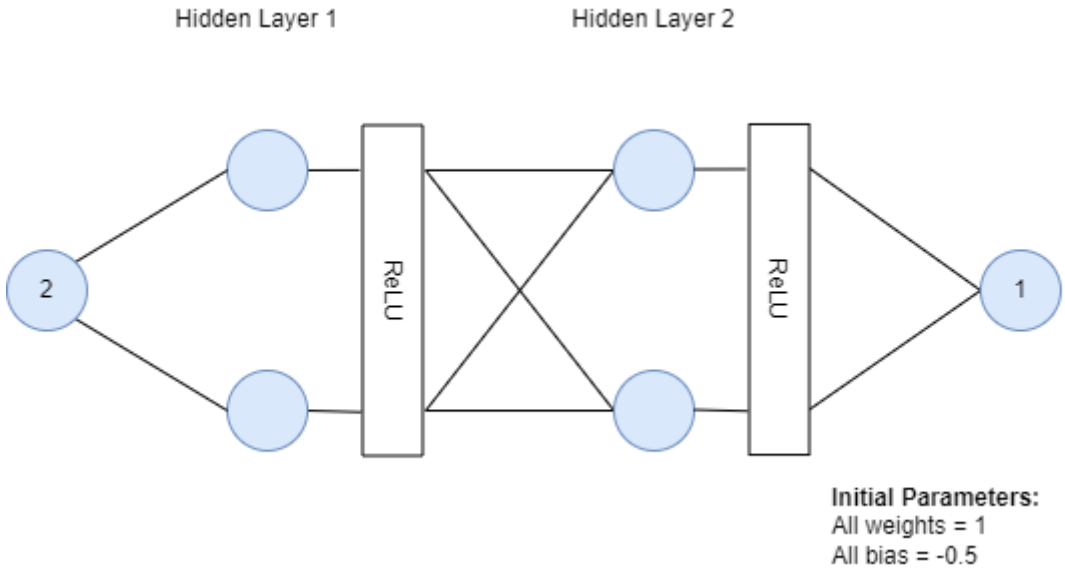
- In this part, you should calculate the forward pass and backpropagation manually and there is no need for any coding.

- Please scan your hand-writting calculation and save it as HW2-1.pdf

- By running the following script, you can check your answer and observe how to do the backpropagation in PyTorch.

- You can change the iterations in script to observe how will the loss change.

1. Please do the forward pass and backpropagation with a neural network as below, the input is 2 and the target is 1. Also, calculate the loss with half and the sum square error. Please update the parameters twice and use the learning rate 0.01.



In [8]:

```
import torch
import torch.nn as nn
import torch.optim as optim
from collections import OrderedDict
```

In [9]:

```
X = torch.tensor([2], dtype= torch.float32)
y = torch.tensor([1], dtype= torch.float32)
```

In [10]:

```
# Half of the sum square error
def loss(y, pred):
    return ((pred-y)**2).sum()/2
```

In [11]:

```
# Show parameters
def show_parameters(i, X, model):
    print(f"Iters {i}")
    print("Input:")
    print(X)
    for layer_name, layers in model.named_modules():
        print("----------------------")
        if not isinstance(layers, nn.Sequential):
            for param_name, param in layers.named_parameters():
                print(f"{layer_name} {param_name} {param}")
                print(f"{layer_name} {param_name} Gradient")
                print(param.grad)
            print(f"{layer_name} output:")
            X = layers(X)
            print(X)

    print("================================================")
```

In [12]:

```
def initialize_weights(model):
    for name, i in model.named_modules():
        if isinstance(i, nn.Linear):
            nn.init.constant_(i.weight.data, 1)
            nn.init.constant_(i.bias.data, -0.5)
```

In [15]:

```
model = nn.Sequential(OrderedDict([("Layer1", nn.Linear(1, 2)),
                                   ("ReLU1", nn.ReLU()),
                                   ("Layer2", nn.Linear(2, 2)),
                                   ("ReLU2", nn.ReLU()),
                                   ("Layer3", nn.Linear(2, 1))]))
initialize_weights(model)
lr = 0.01
n_iters = 10
optimizer = optim.SGD(model.parameters(), lr=lr, momentum=0)
loss_list = []
for i in range(n_iters+1):
    optimizer.zero_grad()
    pred = model(X)
    l = loss(pred, y)
    loss_list.append(l.detach().numpy())
    l.backward()
    show_parameters(i, X, model)
    optimizer.step()
```

```
Iters 0
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[1.],
        [1.]], requires_grad=True)
Layer1 weight Gradient
tensor([[14.],
        [14.]])
Layer1 bias Parameter containing:
tensor([-0.5000, -0.5000], requires_grad=True)
Layer1 bias Gradient
tensor([7., 7.])
Layer1 output:
tensor([1.5000, 1.5000], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([1.5000, 1.5000], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[1., 1.],
        [1., 1.]], requires_grad=True)
Layer2 weight Gradient
tensor([[5.2500, 5.2500],
        [5.2500, 5.2500]])
Layer2 bias Parameter containing:
tensor([-0.5000, -0.5000], requires_grad=True)
Layer2 bias Gradient
tensor([3.5000, 3.5000])
Layer2 output:
tensor([2.5000, 2.5000], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([2.5000, 2.5000], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[1., 1.]], requires_grad=True)
Layer3 weight Gradient
tensor([[8.7500, 8.7500]])
Layer3 bias Parameter containing:
tensor([-0.5000], requires_grad=True)
Layer3 bias Gradient
tensor([3.5000])
Layer3 output:
tensor([4.5000], grad_fn=<AddBackward0>)
================================================
Iters 1
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.8600],
        [0.8600]], requires_grad=True)
Layer1 weight Gradient
tensor([[5.0691],
        [5.0691]])
Layer1 bias Parameter containing:
tensor([-0.5700, -0.5700], requires_grad=True)
```

```
Layer1 bias Gradient
tensor([2.5346, 2.5346])
Layer1 output:
tensor([1.1500, 1.1500], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([1.1500, 1.1500], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9475, 0.9475],
        [0.9475, 0.9475]], requires_grad=True)
Layer2 weight Gradient
tensor([[1.5381, 1.5381],
        [1.5381, 1.5381]])
Layer2 bias Parameter containing:
tensor([-0.5350, -0.5350], requires_grad=True)
Layer2 bias Gradient
tensor([1.3375, 1.3375])
Layer2 output:
tensor([1.6443, 1.6443], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.6443, 1.6443], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.9125, 0.9125]], requires_grad=True)
Layer3 weight Gradient
tensor([[2.4101, 2.4101]])
Layer3 bias Parameter containing:
tensor([-0.5350], requires_grad=True)
Layer3 bias Gradient
tensor([1.4658])
Layer3 output:
tensor([2.4658], grad_fn=<AddBackward0>)
================================================
Iters 2
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.8093],
        [0.8093]], requires_grad=True)
Layer1 weight Gradient
tensor([[2.8667],
        [2.8667]])
Layer1 bias Parameter containing:
tensor([-0.5953, -0.5953], requires_grad=True)
Layer1 bias Gradient
tensor([1.4333, 1.4333])
Layer1 output:
tensor([1.0233, 1.0233], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([1.0233, 1.0233], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9321, 0.9321],
        [0.9321, 0.9321]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.7868, 0.7868],
```

```
        [0.7868, 0.7868]])
Layer2 bias Parameter containing:
tensor([-0.5484, -0.5484], requires_grad=True)
Layer2 bias Gradient
tensor([0.7689, 0.7689])
Layer2 output:
tensor([1.3592, 1.3592], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.3592, 1.3592], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8884, 0.8884]], requires_grad=True)
Layer3 weight Gradient
tensor([[1.1764, 1.1764]])
Layer3 bias Parameter containing:
tensor([-0.5497], requires_grad=True)
Layer3 bias Gradient
tensor([0.8654])
Layer3 output:
tensor([1.8654], grad_fn=<AddBackward0>)
==============================================
Iters 3
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7806],
        [0.7806]], requires_grad=True)
Layer1 weight Gradient
tensor([[1.7852],
        [1.7852]])
Layer1 bias Parameter containing:
tensor([-0.6097, -0.6097], requires_grad=True)
Layer1 bias Gradient
tensor([0.8926, 0.8926])
Layer1 output:
tensor([0.9516, 0.9516], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.9516, 0.9516], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9243, 0.9243],
        [0.9243, 0.9243]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.4595, 0.4595],
        [0.4595, 0.4595]])
Layer2 bias Parameter containing:
tensor([-0.5561, -0.5561], requires_grad=True)
Layer2 bias Gradient
tensor([0.4829, 0.4829])
Layer2 output:
tensor([1.2030, 1.2030], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.2030, 1.2030], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8766, 0.8766]], requires_grad=True)
```

```
Layer3 weight Gradient
tensor([[0.6626, 0.6626]])
Layer3 bias Parameter containing:
tensor([-0.5583], requires_grad=True)
Layer3 bias Gradient
tensor([0.5508])
Layer3 output:
tensor([1.5508], grad_fn=<AddBackward0>)
================================================
Iters 4
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7628],
        [0.7628]], requires_grad=True)
Layer1 weight Gradient
tensor([[1.1615],
        [1.1615]])
Layer1 bias Parameter containing:
tensor([-0.6186, -0.6186], requires_grad=True)
Layer1 bias Gradient
tensor([0.5808, 0.5808])
Layer1 output:
tensor([0.9070, 0.9070], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.9070, 0.9070], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9197, 0.9197],
        [0.9197, 0.9197]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.2864, 0.2864],
        [0.2864, 0.2864]])
Layer2 bias Parameter containing:
tensor([-0.5609, -0.5609], requires_grad=True)
Layer2 bias Gradient
tensor([0.3158, 0.3158])
Layer2 output:
tensor([1.1073, 1.1073], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.1073, 1.1073], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8700, 0.8700]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.4019, 0.4019]])
Layer3 bias Parameter containing:
tensor([-0.5638], requires_grad=True)
Layer3 bias Gradient
tensor([0.3629])
Layer3 output:
tensor([1.3629], grad_fn=<AddBackward0>)
================================================
Iters 5
Input:
tensor([2.])
----------------------
```

```
----------------------
Layer1 weight Parameter containing:
tensor([[0.7512],
        [0.7512]], requires_grad=True)
Layer1 weight Gradient
tensor([[0.7740],
        [0.7740]])
Layer1 bias Parameter containing:
tensor([-0.6244, -0.6244], requires_grad=True)
Layer1 bias Gradient
tensor([0.3870, 0.3870])
Layer1 output:
tensor([0.8779, 0.8779], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.8779, 0.8779], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9168, 0.9168],
        [0.9168, 0.9168]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.1853, 0.1853],
        [0.1853, 0.1853]])
Layer2 bias Parameter containing:
tensor([-0.5641, -0.5641], requires_grad=True)
Layer2 bias Gradient
tensor([0.2111, 0.2111])
Layer2 output:
tensor([1.0457, 1.0457], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.0457, 1.0457], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8660, 0.8660]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.2549, 0.2549]])
Layer3 bias Parameter containing:
tensor([-0.5674], requires_grad=True)
Layer3 bias Gradient
tensor([0.2437])
Layer3 output:
tensor([1.2437], grad_fn=<AddBackward0>)
================================================
Iters 6
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7434],
        [0.7434]], requires_grad=True)
Layer1 weight Gradient
tensor([[0.5231],
        [0.5231]])
Layer1 bias Parameter containing:
tensor([-0.6283, -0.6283], requires_grad=True)
Layer1 bias Gradient
tensor([0.2616, 0.2616])
Layer1 output:
tensor([0.8586, 0.8586], grad_fn=<AddBackward0>)
```

```
----------------------
ReLU1 output:
tensor([0.8586, 0.8586], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9149, 0.9149],
        [0.9149, 0.9149]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.1227, 0.1227],
        [0.1227, 0.1227]])
Layer2 bias Parameter containing:
tensor([-0.5662, -0.5662], requires_grad=True)
Layer2 bias Gradient
tensor([0.1429, 0.1429])
Layer2 output:
tensor([1.0049, 1.0049], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([1.0049, 1.0049], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8634, 0.8634]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.1664, 0.1664]])
Layer3 bias Parameter containing:
tensor([-0.5699], requires_grad=True)
Layer3 bias Gradient
tensor([0.1655])
Layer3 output:
tensor([1.1655], grad_fn=<AddBackward0>)
================================================
Iters 7
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7382],
        [0.7382]], requires_grad=True)
Layer1 weight Gradient
tensor([[0.3567],
        [0.3567]])
Layer1 bias Parameter containing:
tensor([-0.6309, -0.6309], requires_grad=True)
Layer1 bias Gradient
tensor([0.1784, 0.1784])
Layer1 output:
tensor([0.8455, 0.8455], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.8455, 0.8455], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9137, 0.9137],
        [0.9137, 0.9137]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.0825, 0.0825],
        [0.0825, 0.0825]])
Layer2 bias Parameter containing:
tensor([-0.5676, -0.5676], requires_grad=True)
Layer2 bias Gradient
```

```
tensor([0.0976, 0.0976])
Layer2 output:
tensor([0.9775, 0.9775], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([0.9775, 0.9775], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8618, 0.8618]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.1107, 0.1107]])
Layer3 bias Parameter containing:
tensor([-0.5715], requires_grad=True)
Layer3 bias Gradient
tensor([0.1133])
Layer3 output:
tensor([1.1133], grad_fn=<AddBackward0>)
================================================
Iters 8
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7346],
        [0.7346]], requires_grad=True)
Layer1 weight Gradient
tensor([[0.2447],
        [0.2447]])
Layer1 bias Parameter containing:
tensor([-0.6327, -0.6327], requires_grad=True)
Layer1 bias Gradient
tensor([0.1223, 0.1223])
Layer1 output:
tensor([0.8366, 0.8366], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.8366, 0.8366], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9129, 0.9129],
        [0.9129, 0.9129]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.0561, 0.0561],
        [0.0561, 0.0561]])
Layer2 bias Parameter containing:
tensor([-0.5686, -0.5686], requires_grad=True)
Layer2 bias Gradient
tensor([0.0670, 0.0670])
Layer2 output:
tensor([0.9589, 0.9589], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([0.9589, 0.9589], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8607, 0.8607]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.0746, 0.0746]])
Layer3 bias Parameter containing:
tensor([-0.5727], requires_grad=True)
```

```
Layer3 bias Gradient
tensor([0.0778])
Layer3 output:
tensor([1.0778], grad_fn=<AddBackward0>)
================================================
Iters 9
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7322],
        [0.7322]], requires_grad=True)
Layer1 weight Gradient
tensor([[0.1685],
        [0.1685]])
Layer1 bias Parameter containing:
tensor([-0.6339, -0.6339], requires_grad=True)
Layer1 bias Gradient
tensor([0.0842, 0.0842])
Layer1 output:
tensor([0.8305, 0.8305], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.8305, 0.8305], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9123, 0.9123],
        [0.9123, 0.9123]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.0383, 0.0383],
        [0.0383, 0.0383]])
Layer2 bias Parameter containing:
tensor([-0.5692, -0.5692], requires_grad=True)
Layer2 bias Gradient
tensor([0.0462, 0.0462])
Layer2 output:
tensor([0.9461, 0.9461], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([0.9461, 0.9461], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8599, 0.8599]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.0508, 0.0508]])
Layer3 bias Parameter containing:
tensor([-0.5735], requires_grad=True)
Layer3 bias Gradient
tensor([0.0537])
Layer3 output:
tensor([1.0537], grad_fn=<AddBackward0>)
================================================
Iters 10
Input:
tensor([2.])
----------------------
----------------------
Layer1 weight Parameter containing:
tensor([[0.7305],
        [0.7305]], requires_grad=True)
```

```
Layer1 weight Gradient
tensor([[0.1163],
        [0.1163]])
Layer1 bias Parameter containing:
tensor([-0.6347, -0.6347], requires_grad=True)
Layer1 bias Gradient
tensor([0.0581, 0.0581])
Layer1 output:
tensor([0.8263, 0.8263], grad_fn=<AddBackward0>)
----------------------
ReLU1 output:
tensor([0.8263, 0.8263], grad_fn=<ReluBackward0>)
----------------------
Layer2 weight Parameter containing:
tensor([[0.9119, 0.9119],
        [0.9119, 0.9119]], requires_grad=True)
Layer2 weight Gradient
tensor([[0.0263, 0.0263],
        [0.0263, 0.0263]])
Layer2 bias Parameter containing:
tensor([-0.5697, -0.5697], requires_grad=True)
Layer2 bias Gradient
tensor([0.0319, 0.0319])
Layer2 output:
tensor([0.9373, 0.9373], grad_fn=<AddBackward0>)
----------------------
ReLU2 output:
tensor([0.9373, 0.9373], grad_fn=<ReluBackward0>)
----------------------
Layer3 weight Parameter containing:
tensor([[0.8594, 0.8594]], requires_grad=True)
Layer3 weight Gradient
tensor([[0.0348, 0.0348]])
Layer3 bias Parameter containing:
tensor([-0.5740], requires_grad=True)
Layer3 bias Gradient
tensor([0.0371])
Layer3 output:
tensor([1.0371], grad_fn=<AddBackward0>)
================================================
```