Week 3 Experiment

1. Utilizing Numpy to implement Linear model and define loss function and computing loss value
2. Utilizing Numpy to implement Softmax model

The experiment 1

1. import numpy library

In [5]:

```
import numpy as np
```

2. define model

$$f(x) = w * x + b$$

In [6]:

```
def linear_model(x):

    w = np.random.randn(1)
    b = np.random.randn(1)
    output = np.dot(x,w)+b
    return output
```

3. define the loss function

$$Loss = (prediction - y)^2$$

In [7]:

```
def loss_function(y,prediction):
    loss = (prediction - y) * (prediction - y)
    return loss
```

4.define the data x and label y

In [8]:

```
x = np.random.rand(5,1)
y = np.array([1,-1,1,-1,1]).astype('float')

print("The data is as follows:")
for i in range(x.shape[0]):
    print("Item "+str(i),"x:",x[i][0],"y:",y[i])
```

```
The data is as follows:
Item 0 x: 0.9884697980823309 y: 1.0
Item 1 x: 0.33805030874776043 y: -1.0
Item 2 x: 0.1342452050809707 y: 1.0
Item 3 x: 0.47582997431774765 y: -1.0
Item 4 x: 0.3880043849348348 y: 1.0
```

In [10]:

```
prediction = linear_model(x)
loss = loss_function(y,prediction)
print("The all loss value is:")
for i in range(len(loss)):
    print("Item ",str(i),"Loss:",loss[i])
```

```
The all loss value is:
Item  0 Loss: 0.8264405028830231
Item  1 Loss: 1.335495190545648
Item  2 Loss: 0.6791125363466338
Item  3 Loss: 1.303994203029994
Item  4 Loss: 0.7213694623793474
```

The experiment 2

In [11]:

```
import numpy as np
```

In [11]:

```
def softmax(inputs):
    '''
    please refer the content of nndl book and the above example code
    '''
    denominator = 0.0
    for x in inputs:
        denominator += np.exp(x)
    output = []
    for x in inputs:
        numerato = np.exp(x)
        prob = numerato / denominator
        output.append(prob)
    # Returns the indices of the maximum value of all elements
    prediction = np.argmax(output)
    return output,prediction
```

In [12]:

```
out,pred = softmax(np.random.rand(5))
print("the softmax probility:",out)
print("the argmax return value:",pred)
```

```
the softmax probility: [0.13236058458709504, 0.276761002053837, 0.2811
780962412052, 0.1930546102930711, 0.11664570682479175]
the argmax return value: 2
```

please utilize PyTorch to implement the above experiments

Attention:

1. The code logic is the same as the above NumPy code. Therefore, you should understand the above NumPy code, and search the corresponding PyTorch API to rewrite the experiments.
2. Related PyTorch APIs that you might use are as follows:
   - random init tensor: `torch.rand()` , `torch.randn()` , `torch.randint()`
   - matrix multiplication: `torch.mm()` , `torch.dot()` (Be carefull!)
   - exponential operation: `torch.exp()`
   - Returns the indices of the maximum value: `torch.argmax()`
3. For a more detailed API please refer to: https://pytorch.org/docs/stable/torch.html (https://pytorch.org/docs/stable/torch.html)

In [ ]:

```
'''
your code
'''
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```