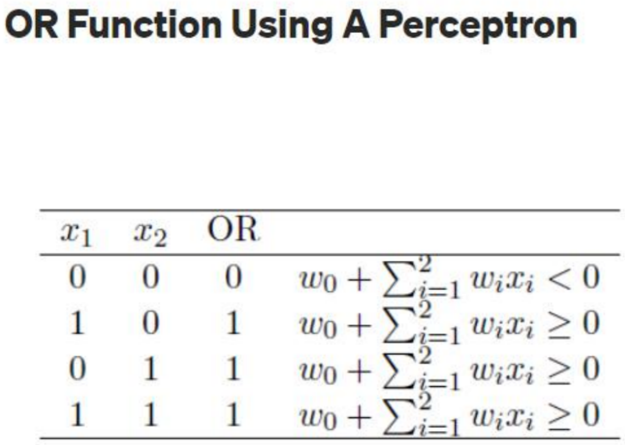
Machine Learning and Artificial Intelligence

Assignment 3

ID: 12232418 Name: Jiang Yuchen

According to the question, we need to use perceptron to learn an OR function. The training data is given as below.



OR Function Using A Perceptron

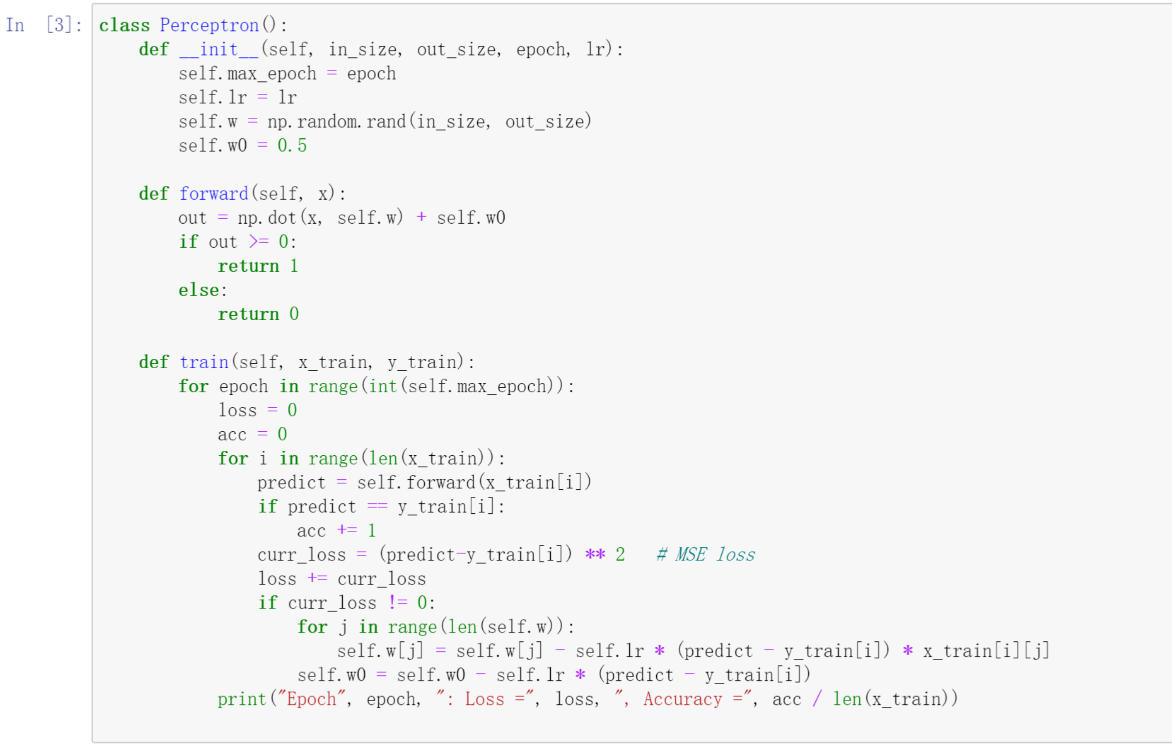
x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

First, we introduce the packages and prepare the data.

```
In [1]: import numpy as np
        import math
```

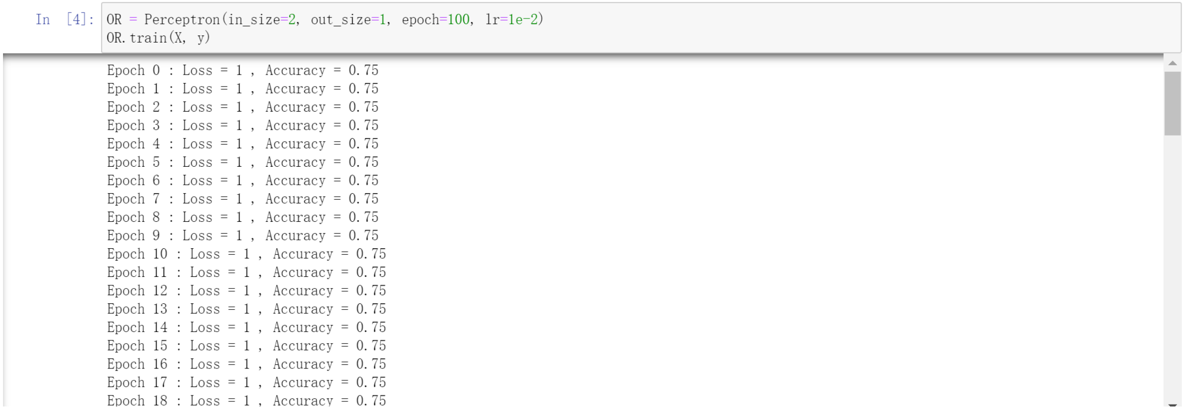
```
In [2]: X = np.array([[0.0, 0.0], [1.0, 0.0], [0.0, 1.0], [1.0, 1.0]])
        y = np.array([0, 1, 1, 1])
```

Then, define the Perceptron class and write forward function and training process. It's hard to decide how to update the parameters since the activation function can't give a non-zero gradient. Thus, we just consider the gradient of other parts. We use Mean Square Error as loss function.

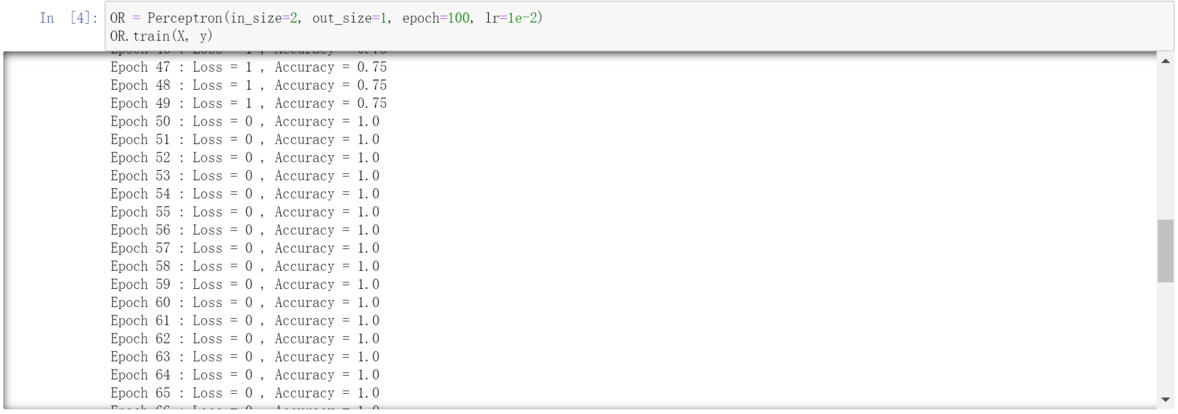


```
In [3]: class Perceptron():
            def __init__(self, in_size, out_size, epoch, lr):
                self.max_epoch = epoch
                self.lr = lr
                self.w = np.random.rand(in_size, out_size)
                self.w0 = 0.5

            def forward(self, x):
                out = np.dot(x, self.w) + self.w0
                if out >= 0:
                    return 1
                else:
                    return 0

            def train(self, x_train, y_train):
                for epoch in range(int(self.max_epoch)):
                    loss = 0
                    acc = 0
                    for i in range(len(x_train)):
                        predict = self.forward(x_train[i])
                        if predict == y_train[i]:
                            acc += 1
                        curr_loss = (predict-y_train[i]) ** 2   # MSE loss
                        loss += curr_loss
                        if curr_loss != 0:
                            for j in range(len(self.w)):
                                self.w[j] = self.w[j] - self.lr * (predict - y_train[i]) * x_train[i][j]
                            self.w0 = self.w0 - self.lr * (predict - y_train[i])
                    print("Epoch", epoch, ": Loss =", loss, ", Accuracy =", acc / len(x_train))
```

Last, train the perceptron with given four pieces of data. Max epoch and learning rate are set to 100 and 0.01 respectively. It converges at epoch 50.



```
In [4]: OR = Perceptron(in_size=2, out_size=1, epoch=100, lr=1e-2)
        OR.train(X, y)

        Epoch 0 : Loss = 1 , Accuracy = 0.75
        Epoch 1 : Loss = 1 , Accuracy = 0.75
        Epoch 2 : Loss = 1 , Accuracy = 0.75
        Epoch 3 : Loss = 1 , Accuracy = 0.75
        Epoch 4 : Loss = 1 , Accuracy = 0.75
        Epoch 5 : Loss = 1 , Accuracy = 0.75
        Epoch 6 : Loss = 1 , Accuracy = 0.75
        Epoch 7 : Loss = 1 , Accuracy = 0.75
        Epoch 8 : Loss = 1 , Accuracy = 0.75
        Epoch 9 : Loss = 1 , Accuracy = 0.75
        Epoch 10 : Loss = 1 , Accuracy = 0.75
        Epoch 11 : Loss = 1 , Accuracy = 0.75
        Epoch 12 : Loss = 1 , Accuracy = 0.75
        Epoch 13 : Loss = 1 , Accuracy = 0.75
        Epoch 14 : Loss = 1 , Accuracy = 0.75
        Epoch 15 : Loss = 1 , Accuracy = 0.75
        Epoch 16 : Loss = 1 , Accuracy = 0.75
        Epoch 17 : Loss = 1 , Accuracy = 0.75
        Epoch 18 : Loss = 1 , Accuracy = 0.75
```



```
In [4]: OR = Perceptron(in_size=2, out_size=1, epoch=100, lr=1e-2)
        OR.train(X, y)

        Epoch 47 : Loss = 1 , Accuracy = 0.75
        Epoch 48 : Loss = 1 , Accuracy = 0.75
        Epoch 49 : Loss = 1 , Accuracy = 0.75
        Epoch 50 : Loss = 0 , Accuracy = 1.0
        Epoch 51 : Loss = 0 , Accuracy = 1.0
        Epoch 52 : Loss = 0 , Accuracy = 1.0
        Epoch 53 : Loss = 0 , Accuracy = 1.0
        Epoch 54 : Loss = 0 , Accuracy = 1.0
        Epoch 55 : Loss = 0 , Accuracy = 1.0
        Epoch 56 : Loss = 0 , Accuracy = 1.0
        Epoch 57 : Loss = 0 , Accuracy = 1.0
        Epoch 58 : Loss = 0 , Accuracy = 1.0
        Epoch 59 : Loss = 0 , Accuracy = 1.0
        Epoch 60 : Loss = 0 , Accuracy = 1.0
        Epoch 61 : Loss = 0 , Accuracy = 1.0
        Epoch 62 : Loss = 0 , Accuracy = 1.0
        Epoch 63 : Loss = 0 , Accuracy = 1.0
        Epoch 64 : Loss = 0 , Accuracy = 1.0
        Epoch 65 : Loss = 0 , Accuracy = 1.0
```

```
In [4]: OR = Perceptron(in_size=2, out_size=1, epoch=100, lr=1e-2)
        OR.train(X, y)
```

```
Epoch 80 : Loss = 0 , Accuracy = 1.0
Epoch 81 : Loss = 0 , Accuracy = 1.0
Epoch 82 : Loss = 0 , Accuracy = 1.0
Epoch 83 : Loss = 0 , Accuracy = 1.0
Epoch 84 : Loss = 0 , Accuracy = 1.0
Epoch 85 : Loss = 0 , Accuracy = 1.0
Epoch 86 : Loss = 0 , Accuracy = 1.0
Epoch 87 : Loss = 0 , Accuracy = 1.0
Epoch 88 : Loss = 0 , Accuracy = 1.0
Epoch 89 : Loss = 0 , Accuracy = 1.0
Epoch 90 : Loss = 0 , Accuracy = 1.0
Epoch 91 : Loss = 0 , Accuracy = 1.0
Epoch 92 : Loss = 0 , Accuracy = 1.0
Epoch 93 : Loss = 0 , Accuracy = 1.0
Epoch 94 : Loss = 0 , Accuracy = 1.0
Epoch 95 : Loss = 0 , Accuracy = 1.0
Epoch 96 : Loss = 0 , Accuracy = 1.0
Epoch 97 : Loss = 0 , Accuracy = 1.0
Epoch 98 : Loss = 0 , Accuracy = 1.0
Epoch 99 : Loss = 0 , Accuracy = 1.0
```