**《深度学习理论与实践》实验报告**

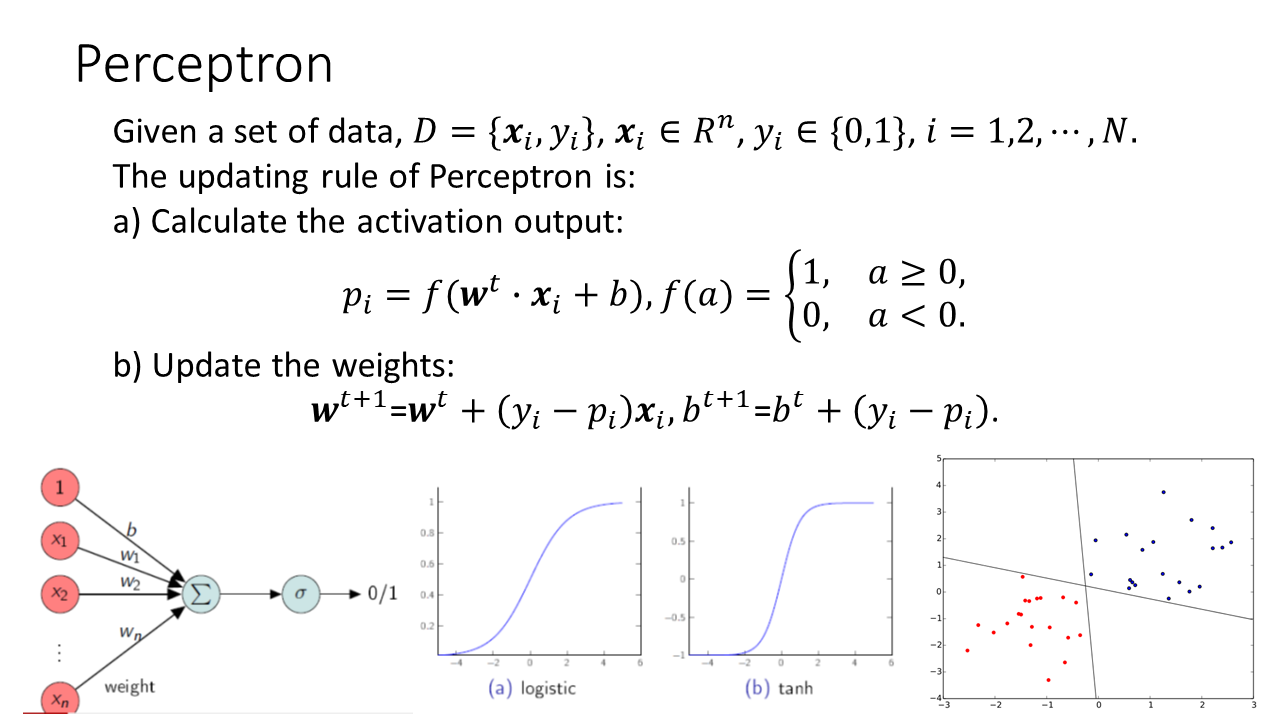
**姓 名：** 徐云哲  **学 号：** 21200211047

**年 级：** 20级 **专 业：** 计算机软件与理论

1. **实验名称**

**Experiment 1 Perceptron**

1. **实验要求**



Define a two-class problem, including 30 positive data and 30 negative data.

Write the code of perceptron using Python.

1. **实验过程**

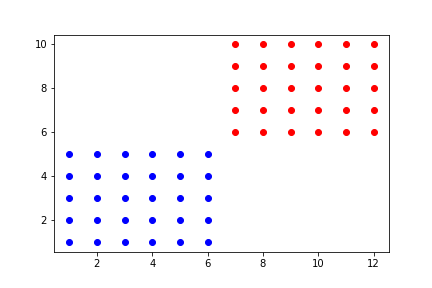
**1、定义一个二分类问题：**

设置数据集有60个点，30个正样本，30个负样本，训练感知机将正样本和负样本正确分类，用一条线可视化分类结果。

简单起见，将坐标都设置为整数，坐标横坐标在6及其以下的为正样本，在图中用蓝色表示，横坐标在7及其以上的为负样本，在图中用红色表示。

正样本标签设为1，负样本标签设为0.

**2、将数据集的点可视化出来：**

****

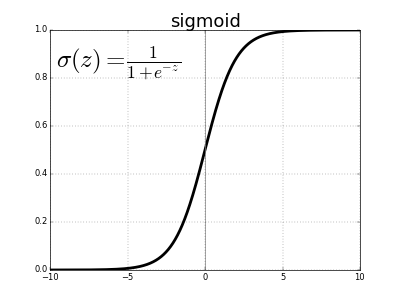
**3、初始化w权重和偏置项b:**

W = [1,0], b = 0

**4、根据实验要求定义目标函数以及权重更新规则:**

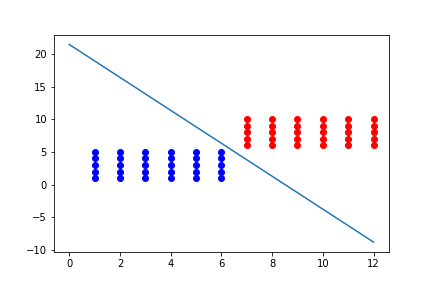
=.

其中，f(a)使用Sigmoid函数激活：



**5、根据目标函数和权重更新规则训练样本，得到训练好的权重和偏置项。**

**6、根据训练好的权重和偏置项分类样本，得到最终分类线：**

****

1. **实验代码**

import numpy as np

import matplotlib.pyplot as plt

def sigmoid(x):

  return 1/(1 + np.exp(-x))

p\_x = np.array([[1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [2, 1], [2, 2], [2, 3], [2, 4], [2, 5], [3, 1], [3, 2], [3, 3], [3, 4], [3, 5], [4, 1], [4, 2], [4, 3], [4, 4], [4, 5], [5, 1], [5, 2], [5, 3], [5, 4], [5, 5], [6, 1], [6, 2], [6, 3], [6, 4], [6, 5], [7, 6], [7, 7], [7, 8], [7, 9], [7, 10], [8, 6], [8, 7], [8, 8], [8, 9], [8, 10], [9, 6], [9, 7], [9, 8], [9, 9], [9, 10], [10, 6], [10, 7], [10, 8], [10, 9], [10, 10], [11, 6], [11, 7], [11, 8], [11, 9], [11, 10], [12, 6], [12, 7], [12, 8], [12, 9], [12, 10]])

y = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

plt.figure()

for i in range(len(p\_x)):

    if y[i] == 1:

        plt.plot(p\_x[i][0], p\_x[i][1], 'bo')

    else:

        plt.plot(p\_x[i][0], p\_x[i][1], 'ro')

w = np.array([1, 0])

b = 0

a = np.zeros(60)

f = np.zeros(60)

p = np.zeros(60)

for j in range(300):

  for i in range(len(p\_x)):

    a[i] = np.dot(w, p\_x[i]) + b

    f[i] = sigmoid(a[i])

    p[i] = f[i]

    w = w + (y[i] - p[i])\*p\_x[i]

    b = b + (y[i] - p[i])

line\_x = [0, 12]

line\_y = [0, 0]

for i in range(len(line\_x)):

  line\_y[i] = (-w[0] \* line\_x[i] - b)/w[1]

plt.plot(line\_x, line\_y)

plt.savefig("picture.png")