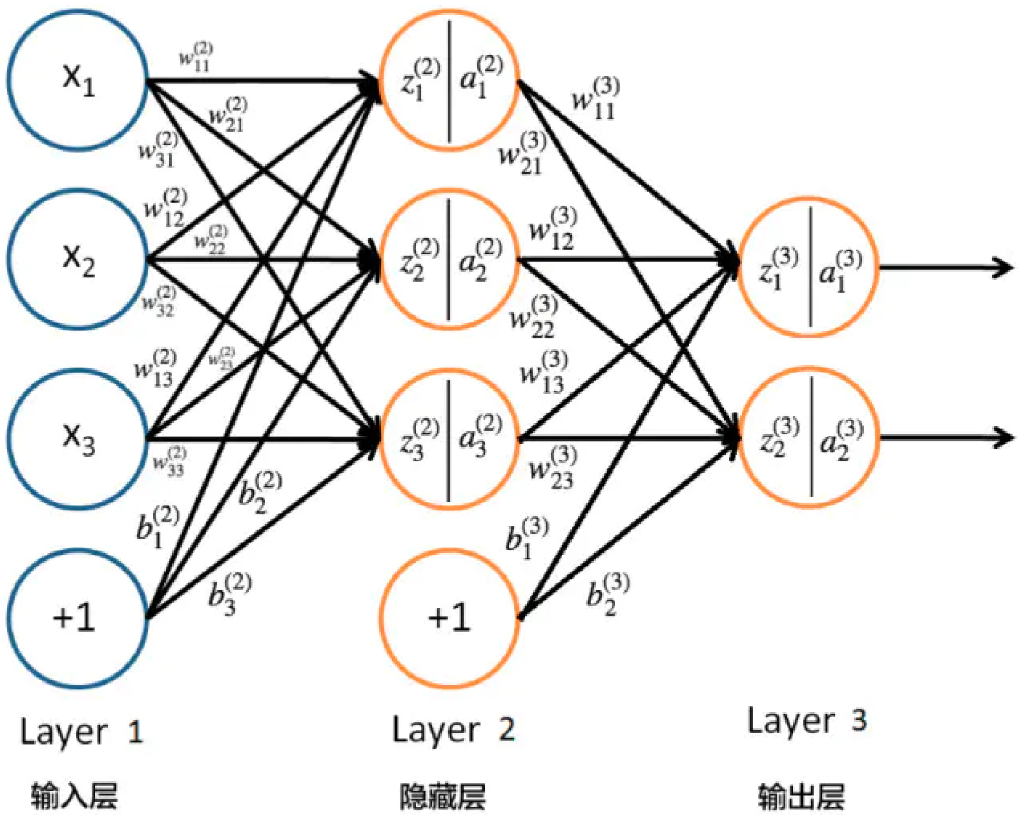
BP神经网络报告

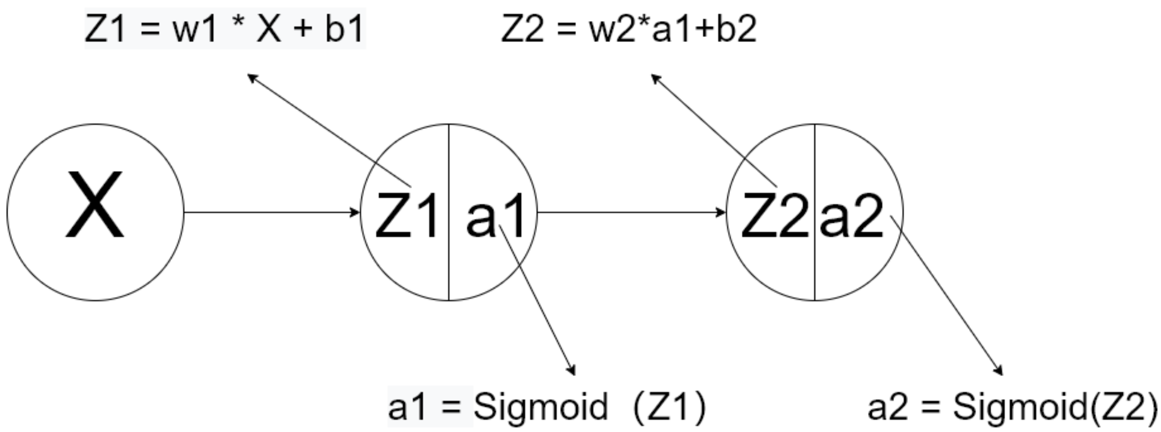
BP神经网络是一种多层的前馈神经网络，其主要的特点是：信号是前向传播的，而误差是反向传播的。具体来说，对于如下的只含一个隐层的神经网络模型：



BP神经网络的过程主要分为两个阶段，第一阶段是信号的前向传播，从输入层经过隐含层，最后到达输出层；第二阶段是误差的反向传播，从输出层到隐含层，最后到输入层，依次调节隐含层到输出层的权重和偏置，输入层到隐含层的权重和偏置。

前向传播

这是一个基础的三层神经网络，按顺序分别是输入层，隐藏层，输出层，传播公式如下图所示。

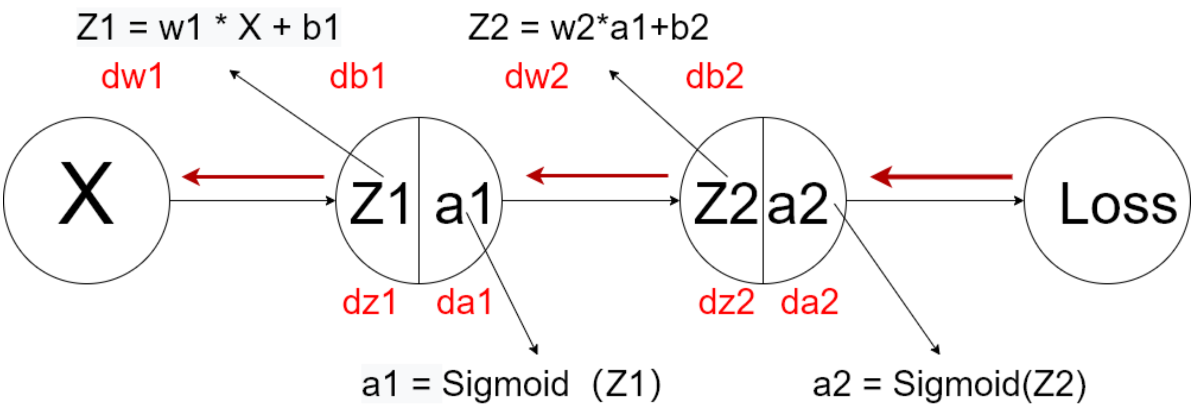


其中X作为输入的一个特征，在这里我们把特征数字化表示成一个数字；w代表权重；b是一个偏差；Loss是损失函数，用于梯度下降计算；a代表节点激活，这里用到了归激活函数sigmoid，sigmoid是一个归一化函数他要把归一化到0-1之间，方便收敛和标签值比较，判断误差，这样通过缩小误差来达到模型的预测功能。最后的得到的a2是前向传播的结果也就是我们模型预测结果分类标签。

```
def forward(x, w1, b1, w2, b2):
    z1 = np.dot(x, w1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, w2) + b2
    a2 = sigmoid(z2)
    return z1, a1, z2, a2
```

反向传播

前向传播相当于是一组用来训练的数据，我们要通过这次前向传播得到的结果和我们的分类标签相减得到误差。我们希望缩小误差让我们的模型有更好的训练效果，这时候就要用到反向传播们这里用的是梯度下降法，让误差按照梯度的方向减小，最后训练打到我们预期的效果。

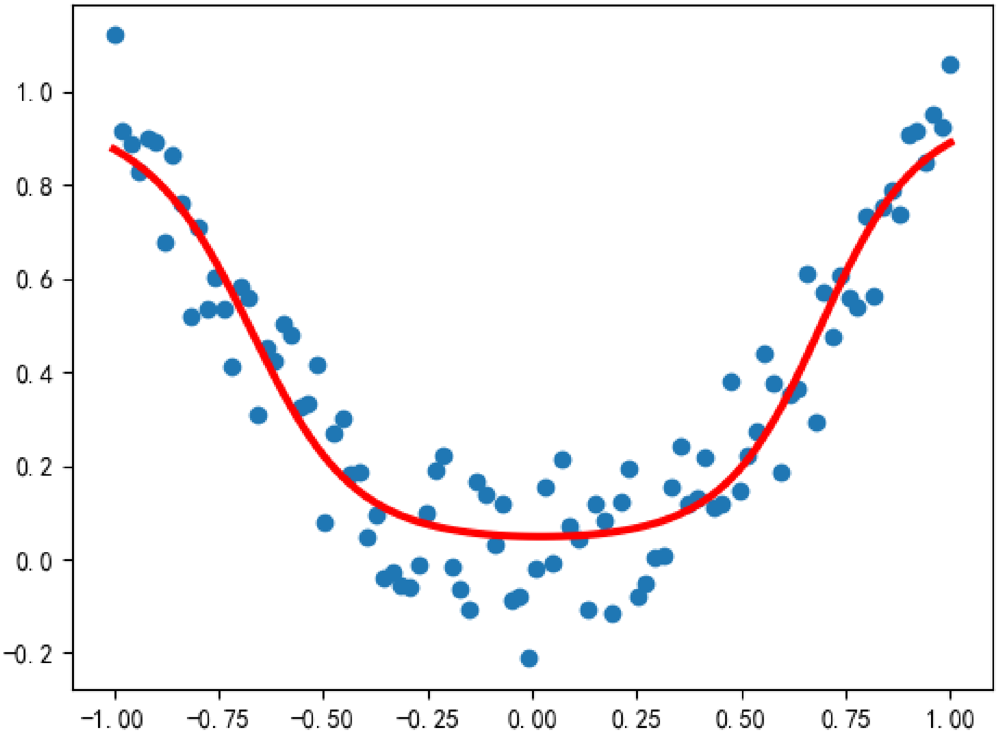



```
def backward(x, y, w1, b1, w2, b2, z1, a1, z2, a2):
    dz2 = (a2 - y) * d_sigmoid(z2)
    dw2 = np.dot(a1.T, dz2)
    db2 = np.sum(dz2, axis=0)
    da1 = np.dot(dz2, w2.T)
    dz1 = da1 * d_sigmoid(z1)
    dw1 = np.dot(x.T, dz1)
    db1 = np.sum(dz1, axis=0)
    return dw1, db1, dw2, db2
```

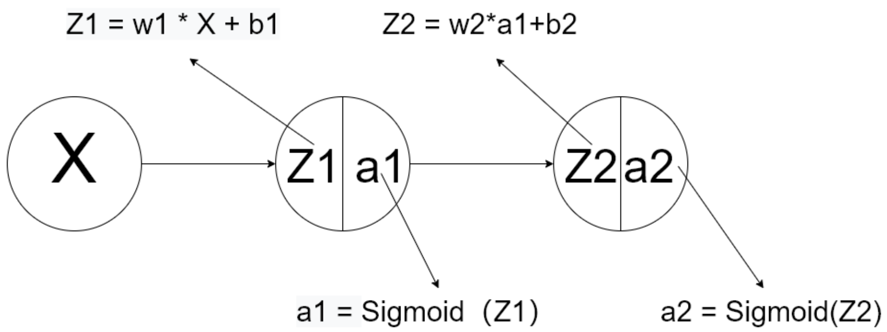
demo代码

```
# BP神经网络拟合非线性曲线
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['axes.unicode_minus'] = False
plt.rcParams['font.sans-serif'] = ['SimHei']
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def d_sigmoid(x):
    return sigmoid(x) * (1 - sigmoid(x))
def forward(x, w1, b1, w2, b2):
    z1 = np.dot(x, w1) + b1
```

```python
    a1 = sigmoid(z1)
    z2 = np.dot(a1, w2) + b2
    a2 = sigmoid(z2)
    return z1, a1, z2, a2
def loss(y, a2):
    return np.mean(np.square(y - a2)/2)
def backward(x, y, w1, b1, w2, b2, z1, a1, z2, a2):
    dz2 = (a2 - y) * d_sigmoid(z2)
    dw2 = np.dot(a1.T, dz2)
    db2 = np.sum(dz2, axis=0)
    da1 = np.dot(dz2, w2.T)
    dz1 = da1 * d_sigmoid(z1)
    dw1 = np.dot(x.T, dz1)
    db1 = np.sum(dz1, axis=0)
    return dw1, db1, dw2, db2
def update(w1, b1, w2, b2, dw1, db1, dw2, db2, lr):
    w1 -= lr * dw1
    b1 -= lr * db1
    w2 -= lr * dw2
    b2 -= lr * db2
    return w1, b1, w2, b2
def train(x, y, w1, b1, w2, b2, lr, epoch):
    for i in range(epoch):
        z1, a1, z2, a2 = forward(x, w1, b1, w2, b2)
        dw1, db1, dw2, db2 = backward(x, y, w1, b1, w2, b2, z1, a1, z2, a2)
        w1, b1, w2, b2 = update(w1, b1, w2, b2, dw1, db1, dw2, db2, lr)
        if i % 10 == 0:
            print('epoch: {}, loss: {}'.format(i, loss(y, a2)))
    return w1, b1, w2, b2
def predict(x, w1, b1, w2, b2):
    z1, a1, z2, a2 = forward(x, w1, b1, w2, b2)
    return a2
if __name__ == '__main__':
    x = np.linspace(-1, 1, 100)[:, np.newaxis]
    noise = np.random.normal(0, 0.1, size=x.shape)
    y = np.square(x) + noise
    w1 = np.random.normal(0, 1, size=(1, 10))
    b1 = np.zeros(10)
    w2 = np.random.normal(0, 1, size=(10, 1))
    b2 = np.zeros(1)
    w1, b1, w2, b2 = train(x, y, w1, b1, w2, b2, lr=0.1, epoch=1000)
    y_pred = predict(x, w1, b1, w2, b2)
    plt.scatter(x, y)
    plt.plot(x, y_pred, 'r-', lw=3)
    plt.show()
```



总结



输入：X
Z1 = w1 * X + b1
a1 = Sigmoid（Z1)
Z2 = w2*a1+b2
a2 = Sigmoid(Z2)
Loss = (y-a2)²

```
正向传播:
z1 = np.dot(x, w1) + b1
a1 = sigmoid(z1)
z2 = np.dot(a1, w2) + b2
```

```
反向传播:
dz2 = (a2 - y) * d_sigmoid(z2)
dw2 = np.dot(a1.T, dz2)
db2 = np.sum(dz2, axis=0)
da1 = np.dot(dz2, w2.T)
dz1 = da1 * d_sigmoid(z1)
dw1 = np.dot(x.T, dz1)
db1 = np.sum(dz1, axis=0)
```

$$\frac{\partial\ \text{Loss}}{\partial z_2} = \frac{\partial\ \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}$$

$$\frac{\partial\ \text{Loss}}{\partial w_2} = \frac{\partial\ \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial\ \text{Loss}}{\partial b_2} = \frac{\partial \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial\ \text{Loss}}{\partial a_1} = \frac{\partial \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial a_1}$$

$$\frac{\partial L_{oss}}{\partial z_1} = \frac{\partial L_{0s}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial a_1}\cdot\frac{\partial a_1}{\partial z_1}$$

$$\frac{\partial\ \text{Loss}}{\partial w_1} = \frac{\partial\ \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial a_1}\cdot\frac{\partial a_1}{\partial z_1}\cdot\frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial\ \text{Loss}}{\partial b_1} = \frac{\partial\ \text{Loss}}{\partial a_2}\cdot\frac{\partial a_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial a_1}\cdot\frac{\partial a_1}{\partial z_1}\cdot\frac{\partial z_1}{\partial b_1}$$

参考