人工智能实验：Topic 5——神经网络

Part 1: 神经网络简介

3190102060　黄嘉欣

一、神经网络的节点

如图 1.1，为一个接受了 3 个输入的节点，其加权输出为 $v=(\omega_1\times x_1)+(\omega_2\times x_2)+(\omega_3\times x_3)+b$，写成矩阵形式即为：$v=\boldsymbol{\omega x}+b$，其中 $\boldsymbol{\omega}=[\omega_1\ \omega_2\ \omega_3]$，$\boldsymbol{x}=[x_1\ x_2\ x_3]^T$。令激活函数为 $y=\varphi(v)$，于是该节点的输出为：$y=\varphi(\boldsymbol{\omega x}+b)$。

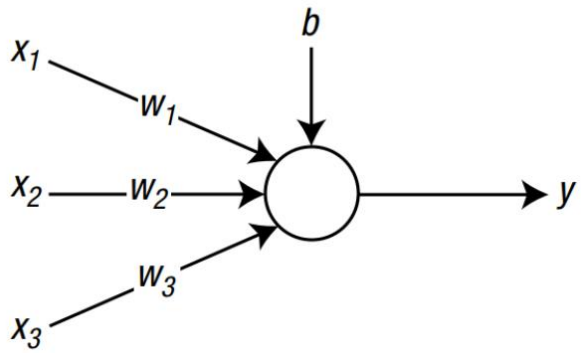

图 1.1 三输入节点示意图

当然，一个或多个节点可以组成神经网络的层，其输入为上一层的输出，输出为下一层的输入，如图 1.2 所示。

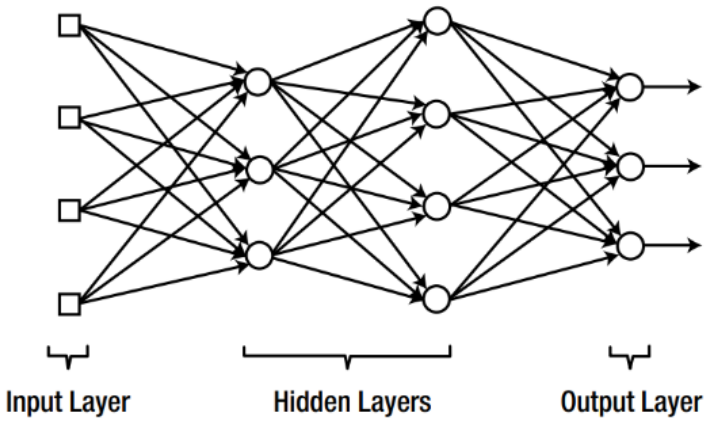

图 1.2 神经网络层示意图

二、神经网络监督学习

神经网络的监督学习过程主要包含以下几个步骤：① 初始化权重；② 通过训练数据对（输入、准确解），计算神经网络的输出和误差；③ 调整权重；④ 重复步骤②和③，直到遍历所有训练数据；⑤ 重复步骤②至④，直到网络达到训练要求。在调整权重

时，我们通常采取 Delta 规则。记x_j为上一级节点j的输出，d_i为第i级节点输出的真值，y_i为第i级节点的输出，$e_i = d_i - y_i$，表示第i级节点的输出误差，ω_{ij}为节点j和节点i之间的权重，$\alpha \in (0,1]$为学习率，则 Delta 规则可表示为$\omega_{ij} = \omega_{ij} + \alpha e_i x_j$，其中$\alpha e_i x_j$为误差更新值，可用$\Delta\omega$表示。为了进一步普适化 Delta 规则，可以将误差更新值定义为$\Delta\omega = \alpha\delta_i x_j$，其中$\delta_i = \varphi'(v_i)e_i$，$v_i$为节点$i$的加权输出。显然，当$\varphi(x) = x$时，有$\delta_i = e_i$。若$\varphi(x)$为 sigmoid 函数，求导很容易得到$\varphi'(x) = \varphi(x)(1 - \varphi(x))$，从而有$\delta_i = \varphi(v_i)(1 - \varphi(v_i))e_i$。通常而言，权重更新计算的规则有 SGD、Batch 和 Mini Batch 几种，其中，SGD 每计算一个训练数据的 error 就立即更新权重；Batch 计算每一个训练数据的 error，根据最后的均值更新权重；Mini Batch 为 SGD 和 Batch 的折中方法，每次根据部分数据调整 weight，直到所有数据都用完。

三、实验 5-1

① 实验题目

考虑一个神经网络，其输入为 3 个节点，输出为 1 个节点，激活函数为 sigmoid 函数。4 组训练数据分别为{0,0,1,0}、{0,1,1,0}，{1,0,1,1}、{1,1,1,1}，其中前三位表示输入，最后一位数字表示正确解。采用 Delta 更新规则。

- 通过 SGD 训练方法及 Delta 规则，对上述神经网络进行训练，并输出训练后的结果；
- 通过 Batch 训练方法及 Delta 规则，对上述神经网络进行训练，并输出训练后的结果；
- 比较 SGD 训练方法及 Batch 训练方法误差(真实结果与输出的 MSE)随 epoch 变化趋势，并可视化结果。

② 实验结果与分析

由二中描述可知，权重的更新与初始权重和学习率有关。令初始权重为[0 0 0]，学习率为 0.9，可以得到 SGD 训练方法及 Batch 训练方法误差随 epoch 的变化趋势如图 3.1 所示。将初始权重改为[1 1 1]，学习率保持不变，得到的图像如图 3.2；保持初始权重为[0 0 0]，学习率修改为 0.1，得到的图像如图 3.3。

比较图 3.1 和图 3.2，可以发现当初始权重发生变化时，训练开始和过程中输出结果与真值的误差也会相应发生变化。比较图 3.1 和图 3.3，可以发现学习率对训练的速度有很大的影响：当学习率较小时，权重的误差更新值较低，训练速度更慢；当学习率较

大时，$\Delta\omega = \alpha\delta_i x_j$更大，权重变化更快，使得训练速度更快。

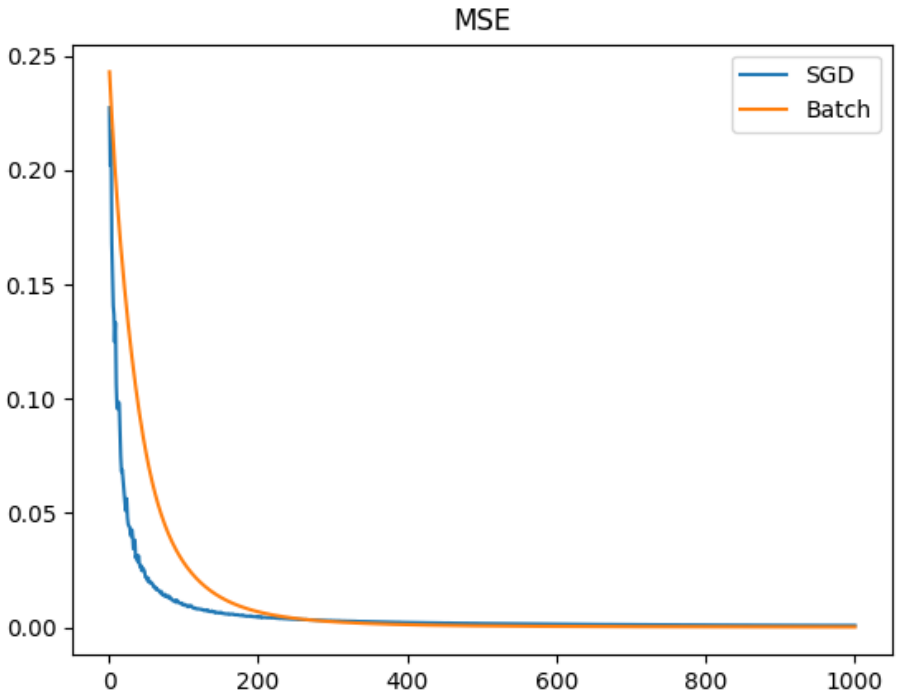

图 3.1 初始权重为[0 0 0]，学习率为 0.9

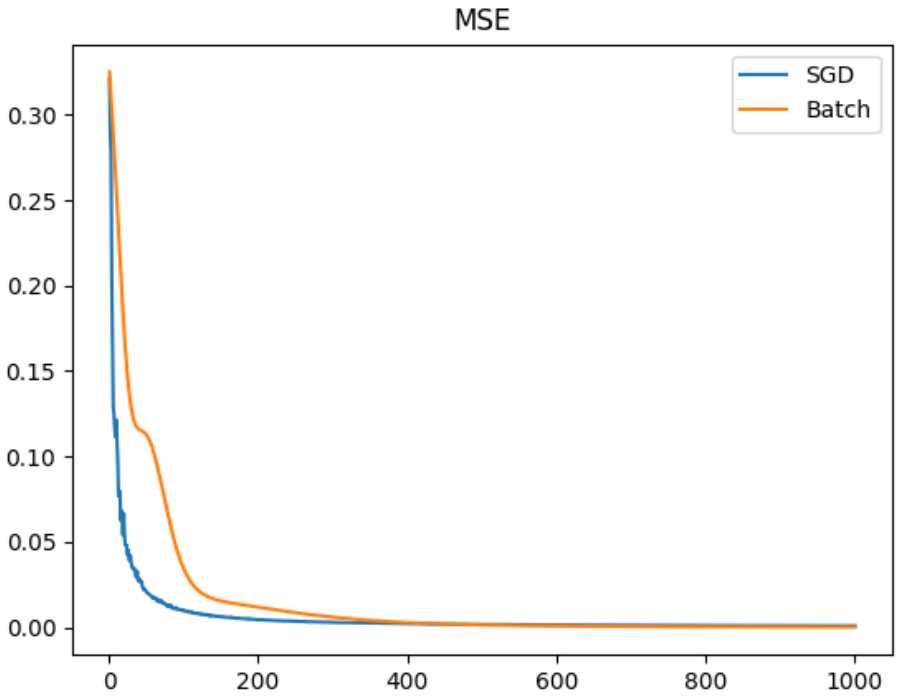

图 3.2 初始权重为[1 1 1]，学习率为 0.9

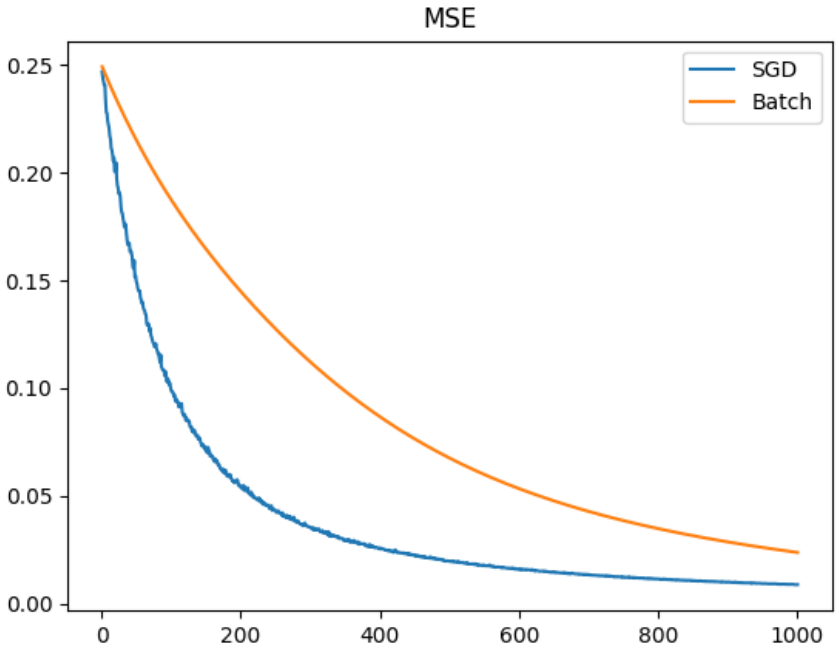

图 3.3 初始权重为[0 0 0]，学习率为 0.1

然而，在上面几张图中，SGD 方法对应的图像呈现锯齿状，根据分析，可能是由于算法从 4 种输入中随机选取训练数据时出现了重复，因此，将随机选取修改为依次选取，得到的图像变得更加平滑，如下所示：

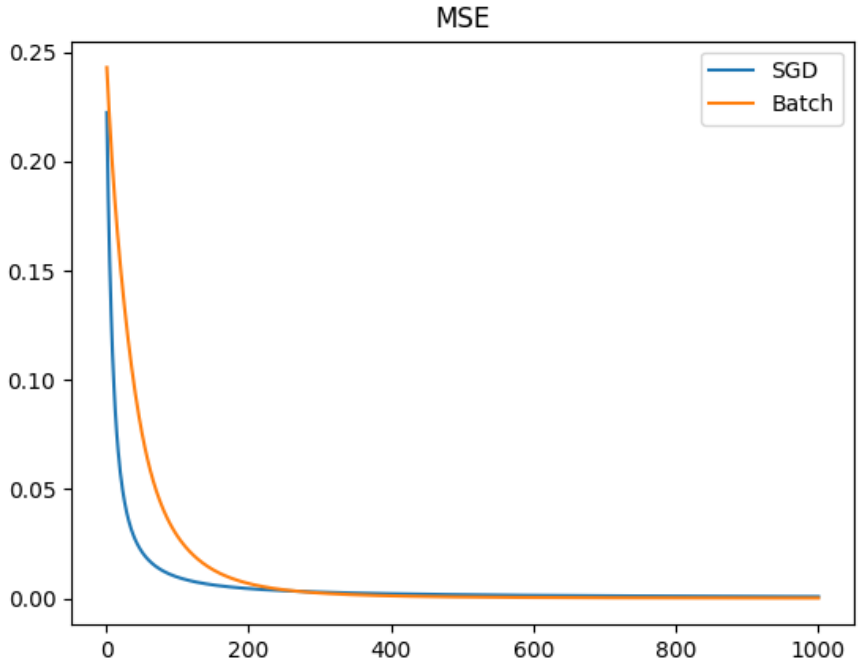

图 3.4 初始权重为[0 0 0]，学习率为 0.9

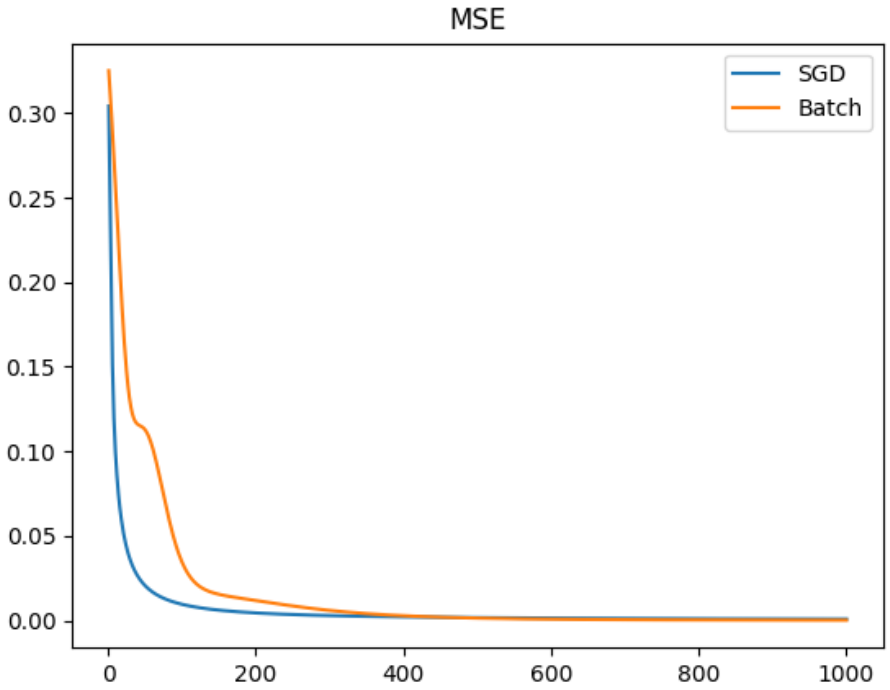

图 3.5 初始权重为[1 1 1]，学习率为 0.9

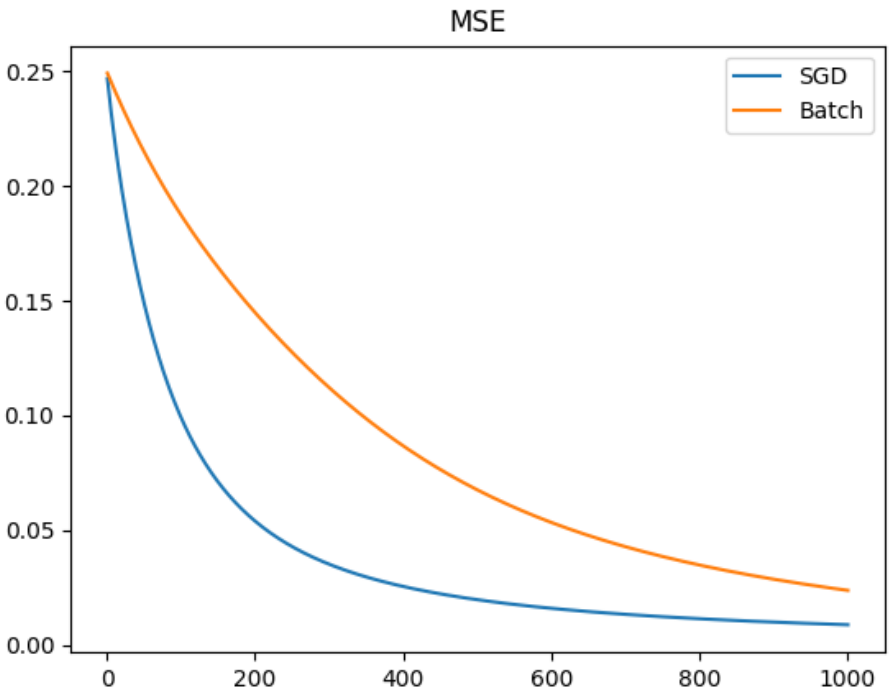

图 3.6 初始权重为[0 0 0]，学习率为 0.1

在初始权重为[0 0 0]，学习率为 0.9，SGD 依次选取训练数据的条件下，得到 SDG 和 Batch 方法训练 999 轮后的结果分别为：

```
Outputs of SGD:
 [[0.50540122 0.50811256 0.56501844 0.56768252]
 [0.50324497 0.50467569 0.61437965 0.6157347 ]
 [0.49526652 0.49270527 0.65069694 0.64836442]
 ...
 [0.033755   0.02721832 0.97796396 0.97263683]
 [0.03373699 0.02720396 0.97797546 0.9726513 ]
 [0.03371901 0.02718963 0.97798694 0.97266574]]
```

```
Outputs of Batch:
 [[0.5        0.5        0.51405879 0.51405879]
 [0.4997969  0.49969534 0.52769043 0.52758919]
 [0.49939122 0.49908733 0.54087612 0.54057425]
 ...
 [0.01479704 0.00828838 0.99577937 0.99244066]
 [0.01477752 0.00827778 0.99578627 0.99245335]
 [0.01475805 0.0082672  0.99579315 0.992466  ]]
```

因此，当训练 999 轮后，SGD 的输出四组结果为[0.0337 0.0272 0.9780 0.9727]，Batch 的四组结果为[0.0148 0.0083 0.9958 0.9925]，与真值[0 0 1 1]已经十分接近，训练结果较好。

四、附录：实验 Python 代码——*nn.py*

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline

def sigmoid(x):
    """
    sigmoid 函数
    Param:
        x: 输入
    Return:
        y: 对应的 sigmoid 函数值
    """
    y = 1./(1+np.exp(-x))
    return y


def mse(x, y):
    """
    计算均方误差
    Param:
        x, y: 输入
    Return:
        x 与 y 的均方误差向量
    """
```

```
    MSEs = []
    for i in range(x.shape[1]):
        xi = x[:,i]                  # 列向量
        MSE = 0                      # 初始化
        for j in range(len(y)):
            square = (xi[j]-y[j])**2 # 平方和
            MSE += square
        MSE /= len(y)                # 取平均
        MSEs.append(MSE)
    return MSEs


def creat_network(net_type, inputs, gts, epoch):
    """
    构造神经网络
    Param:
        net_type: 网络的类型
        inputs: 数据输入
        gts: 正确值
        epoch: 训练轮数
    Return:
        weight: 训练好的权重
    """
    weight = np.array([0,0,0]) # 初始化权重为[0,0,0]
    # weight = np.array([1,1,1]) # 初始化权重为[1,1,1]，行向量
    alpha = 0.9                # 参数
    if net_type == 'SGD':      # 随机梯度下降
        for i in range(epoch):
            for j in range(inputs.shape[0]):     # 每轮需要调整权重的次数
                # index = np.random.randint(inputs.shape[0]) # 随机取一行
                # input = inputs[index]
                input = inputs[j]                           # 依次选取
                output = sigmoid(np.dot(weight, input.T))  # 计算输出
                # e = gts[index] - output
                e = gts[j] - output                        # 输出误差
                delta = alpha*output*(1-output)*e*input    # 误差更新值
                weight = weight + delta                    # 更新权重
    elif net_type == 'Batch':
        deltas = []
        for i in range(epoch):
            for j in range(inputs.shape[0]):               # 每一行
                input = inputs[j]
                output = sigmoid(np.dot(weight, input.T)) # 计算输出
                e = gts[j] - output                        # 输出误差
                delta = alpha*output*(1-output)*e*input    # 误差更新值
```

```
                deltas.append(delta)                        # 保存
            final_delta = sum(deltas)/len(deltas)
            weight = weight + final_delta
    return weight


def main():
    dataset = np.array([[0,0,1,0],[0,1,1,0],[1,0,1,1],[1,1,1,1]])
    inputs = dataset[:,0:3]       # 输入
    gts = dataset[:,3:4]          # ground truth
    outputs_SGD = []              # 初始化输出，利于可视化
    outputs_Batch = []
    epochs = 1000
    for epoch in range(1, epochs):
        # 使用 SGD 构造神经网络
        nn_SGD = creat_network('SGD', inputs, gts, epoch)
        # 使用 Bacth 构造神经网络
        nn_Batch = creat_network('Batch', inputs, gts, epoch)
        output_SGD = sigmoid(np.dot(nn_SGD, inputs.T))     # SGD 的结果
        output_Batch = sigmoid(np.dot(nn_Batch, inputs.T))# Batch 的结果
        outputs_SGD.append(output_SGD)                      # 保存结果
        outputs_Batch.append(output_Batch)
    outputs_SGD = np.array(outputs_SGD)
    outputs_Batch = np.array(outputs_Batch)
    print('Outputs of SGD:\n',outputs_SGD)
    print('Outputs of Batch:\n',outputs_Batch)
    MSE_SGD = mse(outputs_SGD.T, gts)                      # 均方误差
    MSE_Batch = mse(outputs_Batch.T, gts)
    epoch = np.linspace(1,epochs,epochs-1)
    model_SGD = make_interp_spline(epoch, MSE_SGD)       # 绘制平滑曲线
    model_Batch = make_interp_spline(epoch, MSE_Batch)   # 绘制平滑曲线
    y_SGD = model_SGD(epoch)
    y_Batch = model_Batch(epoch)
    l1, = plt.plot(epoch, y_SGD)
    l2, = plt.plot(epoch, y_Batch)
    plt.legend(handles=[l1,l2],labels=["SGD","Batch"],loc='best')
    plt.title('MSE')
    plt.show()


if __name__ == "__main__":
    main()
```