BP 网络的训练与测试

文荟俨 19S103256

一、 实验目的

编程实现 3-3-4 型 (即输入层为 3，隐藏层为 3，输出层为 4)BP 神经网络算法，熟练掌握前向传播和反向传播学习算法，使用 *Softmax* 输出函数，等概率生成 4 类三维数据进行训练。利用训练好的网络估计几个样本的后验概率，和使用贝叶斯公式进行理论计算的置信度作对比，并生成一组测试样本集，计算识别率。

二、 实验计划

1. 生成数据

生成 4000 个模式的训练集 D，每一类均服从高斯分布，各有 1000 个训练样本，按照训练集、测试集 4:1 进行划分。具体参数如表 1 所示，样本维度为三维。

表 1 样本生成参数表

i	μ_i	Σ_i
1	0	I
2	$\begin{pmatrix}0\\1\\0\end{pmatrix}$	$\begin{pmatrix}1&0&1\\0&2&2\\1&2&5\end{pmatrix}$
3	$\begin{pmatrix}-1\\0\\1\end{pmatrix}$	$Diag[2,6,1]$
4	$\begin{pmatrix}0\\0.5\\1\end{pmatrix}$	$Diag[2,1,3]$

生成样本时，我们采用 np.random.multivariate_normal (mean, cov, size=None, check_valid=None, tol=None)方法，它用于生成多元正态分布矩阵。其中 mean 和 cov 为必要的传参，而 size，check_valid 以及 tol 为可选参数。

2.网络结构定义

网络采用 3-3-4 型 BP 网，含偏置项，隐藏层激活函数采用 ReLU，输出层采用 *Softmax*，损失函数 *J*(*W*)采用均方差，定义为输出端期望输出t_k和实际输出z_k的差的平方和，如式(1)所示：

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}||\boldsymbol{t} - \boldsymbol{z}||^2 \quad . \tag{1}$$

其中，***t*** 和 ***z*** 是长为 ***c*** 的目标向量和网络输出向量；***w*** 表示网络的所有权值。本实验采用的网络结构如图 1 所示，这个网络由一个输入层，一个隐含层和一个输出层组成。他们由可修正的权值互连，这些权值由层间的连线表示。除了连接输入单元，每个单元还连接着一个偏置。

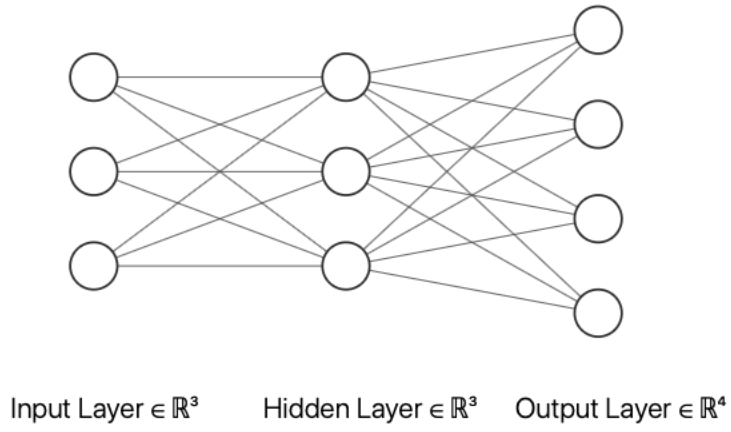

图 1 BP 神经网络结构图(偏置项未展示)

3. 前向传播

每一个三维输入向量都提供给输入层，而每一个输入单元的输出结果则等于输入向量中对应的那个分量。隐单元对它的各个输入进行加权求和运算而形成标量的“净激活”(net activation，简称 net)。也就是说，净激活是输入信号与隐含层权值的内积，它的定义如式(2)所示。

$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} \equiv \mathbf{w}_j^t \mathbf{x} \tag{2}$$

这里下标 *i* 是输入层单元的索引值，*j* 是隐含层单元的索引。w_{ij}表示输入层单元 *i* 到隐含层单元 *j* 的权值。每一个隐含层单元激发出一个输出分量，这个分量是它激活的非线性函数，*f*(*net*)，即式(3)。

$$y_j = f(net_j). \tag{3}$$

本实验中对 *f* 的定义采用整流线性单位函数(ReLU)，如式(4)。

$$f(x)=max(0, x). \tag{4}$$

具体的，它的函数图像如图 2 所示。

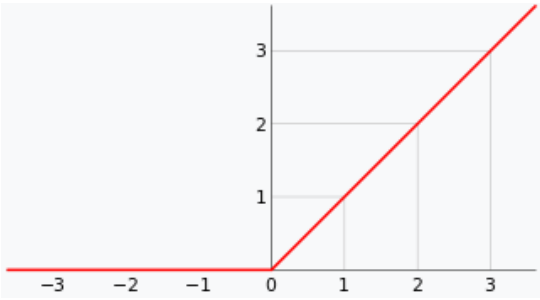

图 2 ReLU 函数示意图

每个输出单元在隐含层单元信号的基础上用类似的方法可以算出它的净激活如式(5)所示。

$$net_k = \sum_{j=1}^{n_H} x_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} \equiv \boldsymbol{w}_k^t \boldsymbol{y} \tag{5}$$

这里下标 k 为输出层的单元索引，n_H表示隐含层单元的数目。我们把偏置单元在数学上看成等价于一个输出恒为y_o=1 的隐含层单元，将输出单元标记为Z_k，这样输出单元对 *net* 的非线性函数就是式(6)。

$$z_k = f(net_k) \tag{6}$$

综合式(2)(3)(5)(6)，可以得到最终的判别函数如式(7)。

$$g_k(\mathbf{x}) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj}\ f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \tag{7}$$

4.反向传播

本实验的损失函数采用式(1)的形式，反向传播学习规则基于梯度下降算法，权值首先被初始化为较小的随机值，然后向误差减小的方向调整，如式(8)。

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}} \tag{8}$$

其中，η是学习率，表示权值的相对变化尺度。我们现在对本实验采用的三层 BP 网络推导分析(8)。考虑第一个隐含层到输出层的权值w_{ij}，由于误差并不是明显决定于w_{jk}，我们需要使用链式求导法则(9)。

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \delta_k \frac{\partial net_k}{\partial w_{kj}} . \tag{9}$$

其中单元 k 的敏感度定义为式(10)。

$$\delta_k \equiv -\partial J / \partial net_k \tag{10}$$

此敏感度描述总误差如何随着单元的激发而变化，对式(1)微分，我们发现对于这样一个输出单元，δ_k为式(11)。

$$\delta_k \equiv -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k)f'(net_k) \tag{11}$$

式(9)的最后一步可由式(5)得到:

$$\frac{\partial net_k}{\partial w_{kj}} = y_j \tag{12}$$

综上所述，我们得到了隐含层到输出层的权值更新规则(13):

$$\Delta w_{kj} = \eta\delta_k y_j = \eta(t_k - z_k)f'(net_k)y_j \tag{13}$$

输入层到隐含层的权值学习规则更微妙，对式(8)再运用链式求导法则计算，得到式(14)。

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} \tag{14}$$

右边的第一项包含了所有的权值w_{jk}，计算过程如式(15)。

$$\begin{aligned}\frac{\partial J}{\partial y_j} &= \frac{\partial}{\partial y_j}\left[1/2\sum_{k=1}^{c}(t_k - z_k)^2\right] \\ &= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j} \\ &= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} \\ &= -\sum_{k=1}^{c}(t_k - z_k)f'(net_k)w_{jk}.\end{aligned} \tag{15}$$

上面的第二部里需要再次用到链式求导法则。式（19）中输出单元的总和可以表示隐单元怎样影响每个输出单元的误差。仿照式(11)，我们可以定义隐单元的敏感度δ_j为式(16)。

$$\delta_j \equiv f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k \quad . \tag{16}$$

一个隐单元的敏感度是各个输出单元敏感度的加权和，权重为隐含层到输出层的权值w_{jk}，然后与$f'(net_j)$相乘。因此得到输入层到隐含层的权值学习规则(17)。

$$\Delta w_{ji} = \eta x_i\delta_j = \eta x_i f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k \tag{17}$$

5.训练算法

根据前向传播和反向传播推导得到的结果，我们定义训练流程如图 3 所示。

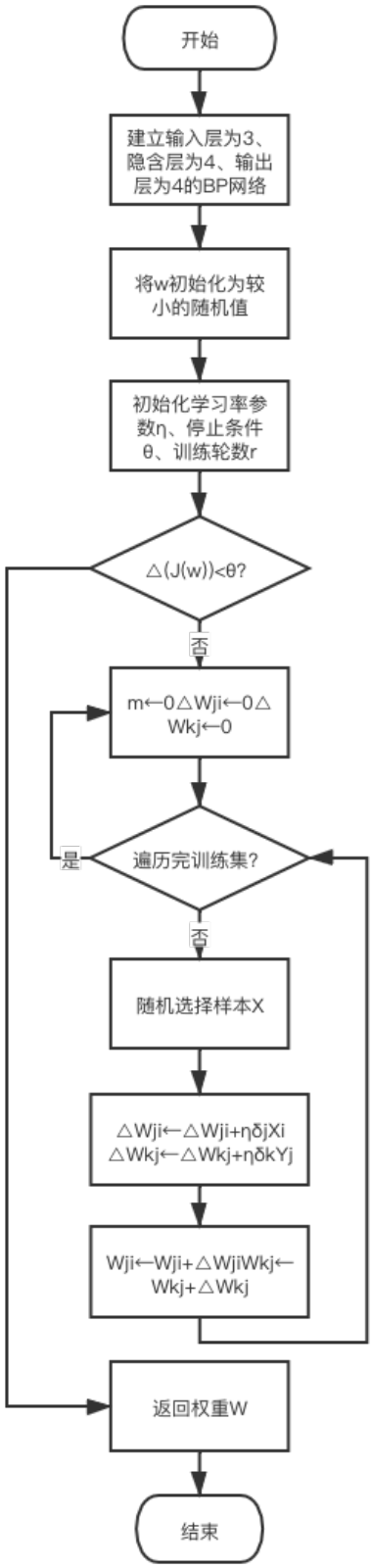

图 3 网络训练流程图

6.样本预测

生成一组测试样本集（400 个模式,每个类别 100 个），计算识别率，并利用训练好的网络来估计如下 5 个模式的每一个后验概率:

$$x_1=(0,0,0)^t, x_2=(-1,0,1)^t, x_3=(0.5,-0.5,0)^t, x_4=(-1,0,0)^t, x_5=(0,0,-1)^t$$

预测算法较为简单，将需要预测的样本的三维信息输入网络，通过训练好的权重前向传播，最后经过 *Softmax* 之后得到输出，求其最大值所在的索引，把它作为预测的标签。规则如式(18):

$$max(Softmax(net_k)) \tag{18}$$

三、 实验过程

该章节对代码具体实现的核心部分作简要说明。均方差的定义如下：

```
def mean_square(output, y):
    return np.sum((y - output) ** 2) / 2.0
```

*Softmax*的定义如下：

```
def softmax(x):
    exps = np.exp(x - np.max(x))
    return exps / np.sum(exps)
```

ReLU函数的定义如下:

```
def ReLuFunc(x):
    x = (np.abs(x) + x) / 2.0
    return x
```

前向传播的定义如下:

```
# Hidden layer
for j in range(0, 3):
for i in range(0, 3):
net1[j] = net1[j] + w1[j][i] * x[i]
    net1[j] += bias1[j]
    fnet1 = ReLuFunc(net1)

# Output layer
for k in range(0, 4):
     for j in range(0, 3):
         net2[k] = net2[k] + w2[k][j] * fnet1[j]
```

```
    net2[k] += bias2[k]
 output = softmax(net2)
```

反向传播的定义如下:

```
# Back
det_k = np.zeros(4)
for k in range(0, 4):
    det_k[k] = (y[k] - output[k]) * (output[k] * (1 -
output[k]))
det_j = np.zeros(3)
for j in range(0, 3):
    sum = 0
    for k in range(0, 4):
        sum = sum + w2[k][j] * det_k[k]
    if net1[j] >= 0:
        det_j[j] = sum
    else:
        det_j[j] = 0

# Update w and b
for j in range(0, 3):
   for i in range(0, 3):
      w1[j][i] = w1[j][i] + learning_rate * \
                            det_j[j] * x[i]
   bias1[j] = bias1[j] + learning_rate * det_j[j]
for k in range(0, 4):
    for j in range(0, 3):
       w2[k][j] = w2[k][j] + learning_rate * \
                            det_k[k] * fnet1[j]
    bias2[k] = bias2[k] + learning_rate * det_k[k]
```

预测准确率计算定义如下:

```
if np.argmax(output) == y:
   acc += 1
```

四、 结果分析

1. 数据生成

初始数据的分布如图 4 所示，可观测到不同类间样本较为集中，不太利于区分。

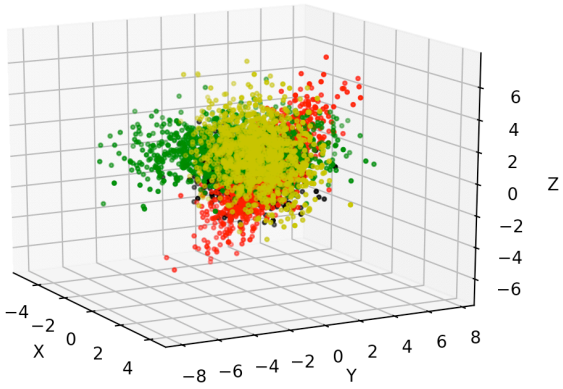

图 4 初始样本分布图

2. 网络训练及测试集预测

实际训练中，我们将学习率设为 0.01，每隔 2 轮降低为原来的 0.97 倍，每轮遍历所有的训练样本，得到结果图 5 所示。其中，浅色的点状虚线是测试集 400 个样本的准确率随训练轮数的变化，最后稳定在 51%左右。

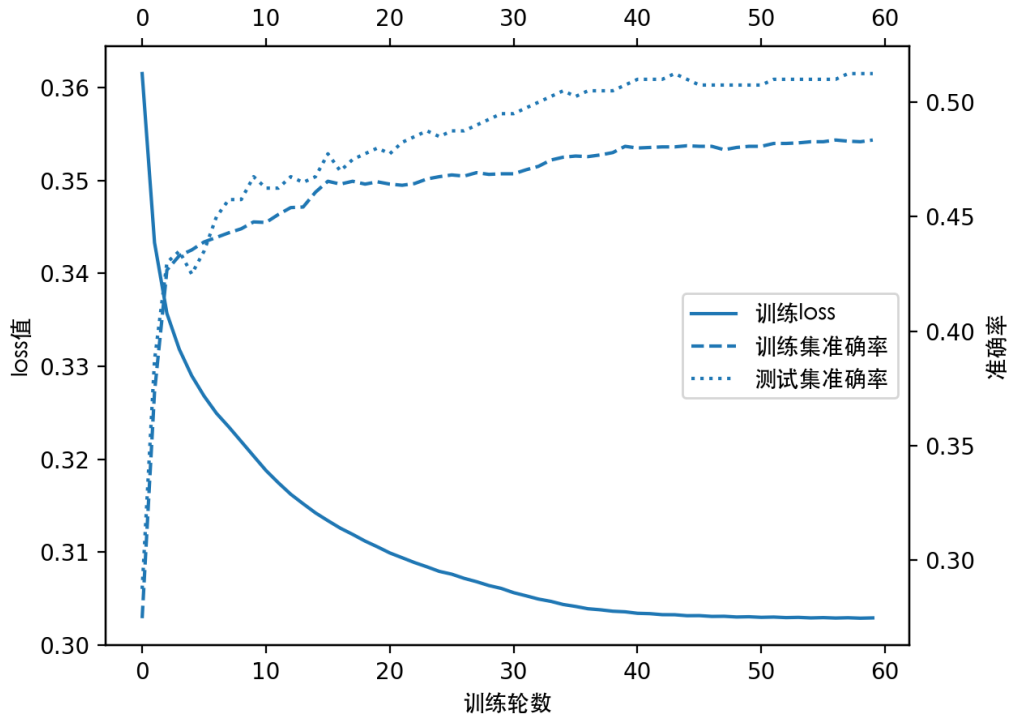

图 5 训练过程各指标曲线图

3. 给定样本预测

(1) 训练网络预测

使用训练好的网络预测如下 5 个样本，结果如图 6 所示。

$x_1=(0,0,0)^t, x_2=(-1,0,1)^t, x_3=(0.5,-0.5,0)^t, x_4=(-1,0,0)^t, x_5=(0,0,-1)^t$

```
Sample 0, inference label:0, and its probability:0.5111009589070249
Sample 1, inference label:3, and its probability:0.3863134716587585
Sample 2, inference label:0, and its probability:0.5496677190355785
Sample 3, inference label:0, and its probability:0.5294875390435843
Sample 4, inference label:0, and its probability:0.36370247601319006
```

图 6 样本预测置信度结果

(2)理论验算

该部分采用贝叶斯函数进行估计，先验概率为 0.25，似然函数采用多维高斯分布概率密度函数，其定义如式(19)：

$$f(x_1,x_2,\cdots,x_p)=\frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})] \tag{19}$$

具体计算的函数定义如下：

```
u = np.asarray([[0, 0, 0],
                [0, 1, 0],
                [-1, 0, 1],
                [0, 0.5, 1]])
sigma = np.asarray([[[1, 0, 0], [0, 1, 0], [0, 0, 1]],
                    [[1, 0, 1], [0, 2, 2], [1, 2, 5]],
                    [[2, 0, 0], [0, 6, 0], [0, 0, 1]],
                    [[2, 0, 0], [0, 1, 0], [0, 0, 3]]])
p = np.zeros(4)
for i in range(0, 4):
    det = np.linalg.det(sigma[i])
    inv = np.linalg.inv(sigma[i])
    xu_ = np.transpose(x - u[i])
    xu = np.transpose(xu_)
    first = np.dot(xu_, inv)
    second = np.dot(first, xu)
    d = len(x)
    p[i] = 1.0/((2*np.pi)**(d/2.0)*np.sqrt(det))*np.exp(-
second/2.0)
prob = max(p)/np.sum(p)
```

得到结果如图 7：

```
Sample 0, inference label:0, and its probability:0.5731991260468056
Sample 1, inference label:0, and its probability:0.3823980301348944
Sample 2, inference label:0, and its probability:0.6136093588062754
Sample 3, inference label:0, and its probability:0.5483914341072756
Sample 4, inference label:0, and its probability:0.5007146343700822
```

图 7 使用贝叶斯公式进行置信度理论计算

对比网络预测结果如表 2 所示：

表 2 BP 网络预测结果与理论计算比较

样本数据	先验概率	理论后验概率(最大值)	理论结果(最大值对应类别)	预测后验概率	预测结果
(0,0,0)	0.25	0.57	0	0.51	0
(-1,0,1)	0.25	0.38	0	0.39	3
(0.5,-0.5,0)	0.25	0.61	0	0.55	0
(-1,0,0)	0.25	0.55	0	0.53	0
(0,0,-1)	0.25	0.50	0	0.36	0

分析：对 5 个样本的预测，BP 网计算得到的样本类别分别是 0、3、0、0、0，而理论计算的结果是 0、0、0、0、0，只有第 2 个样本的结果不一致。