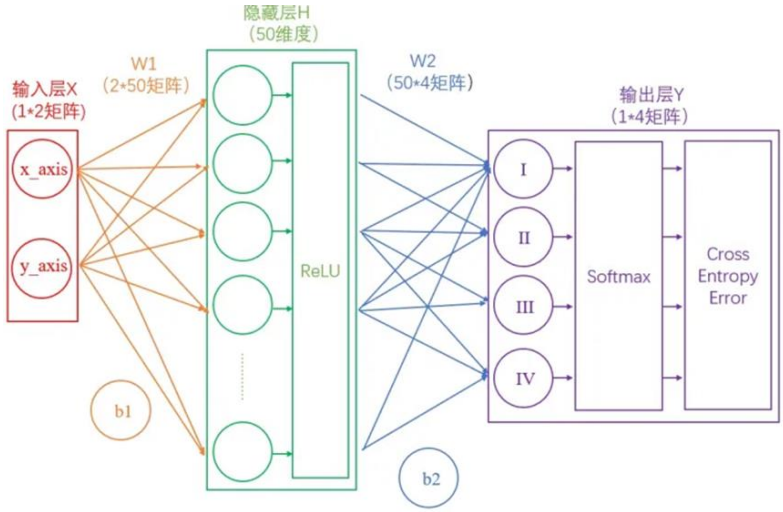
实验一 自动识别技术——语音识别系统的设计与实现（3）

1. 人工神经网络及 MLPClassifier 算法学习

1.1 人工神经网络初步认识

人工神经网络是（Artificial Neural Network, ANN）一种模仿生物神经网络学习模式的机器学习模型。由多个神经元（或称节点）组成，这些神经元通过连接权重相互连接，构成多层的网络结构。每个神经元接收到来自其它神经元的信号，并将这些信号加权线性组合后通过激活函数进行非线性转换，最终输出给下一层神经元或输出层。

下面是一个简单的例子来认识 ANN，从而理解其中的一些概念。



上图是一个两层神经网络，理论上两层神经网络已经可以拟合任意函数。其中：

- 输入层：是坐标值，例如：

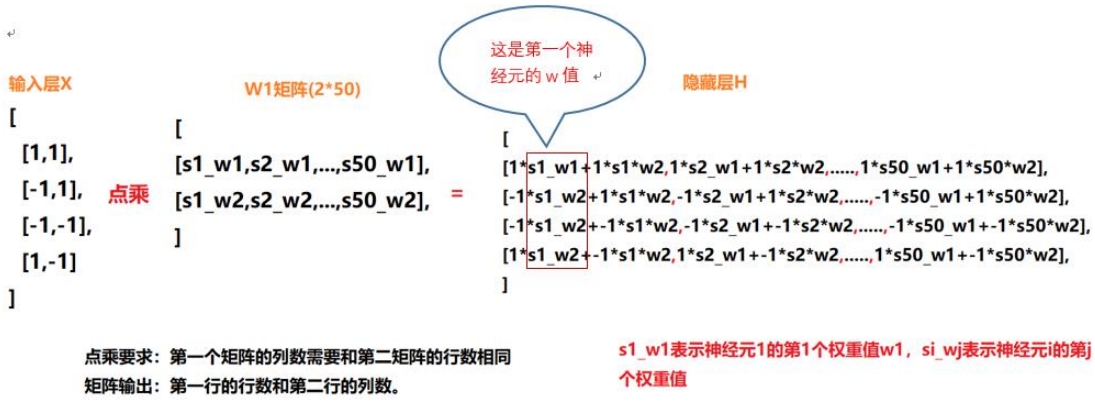
  [
  [1,1],
  [-1,1],
  [-1,-1],
  [1,-1]
  ]

  可以看作是一个 4*2 的矩阵。输入层的元素维度与输入量的特征息息相关，如果输入的是一张 32*32 像素的灰度图像，那么输入层的维度就是 32*32。

- 从输入层到隐藏层

  连接输入层和隐藏层的是 W1 和 b1，W 是一个权重(权重越高，这个特征也就越重要)，b 是一个偏置，如果有多个特征，那么就有多个 w，记作：$w^T * x + b$。

  上图中，隐藏层为 50 维，即有 50 个隐藏神经元，权重矩阵为（2，50）。



➢ 从隐藏层到输出层

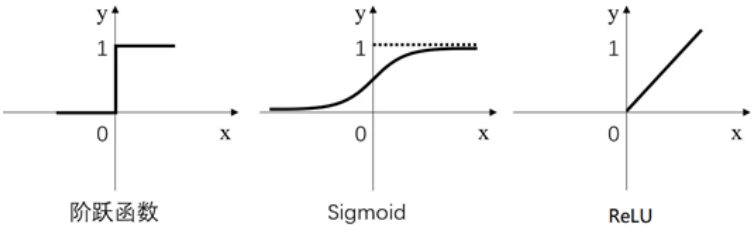
连接隐藏层和输出层的是 W2 和 b2，同样是通过矩阵运算进行的，W2 是一个 50*4 的矩阵，目的是为了将 50 个神经元压缩到 4 个输出特征，也就是每一个数据集在 4 个类别的概率

➢ 激活层

激活层是为矩阵运算的结果添加非线性变换，增加模型的非线性表达能力，使得神经网络可以更好地处理复杂的输入数据，例如图像、文本和语音等。

每个隐藏层计算（矩阵线性运算）之后，都需要加一层激活层，要不然该层线性计算是没有意义的。激活函数的选择也非常重要，不同的激活函数具有不同的特点。

激活层常用的激活函数三种，分别是阶跃函数、Sigmoid 和 ReLU，如下图：



✧ Sigmoid：当输入趋近于正无穷/负无穷时，输出无限接近于 1/0。

✧ ReLU：当输入小于 0 时，输出 0；当输入大于 0 时，输出等于输入。

ReLU 是当前较为常用的激活函数

➢ 输出的正规化

假设某个样本输出 Y 的值可能会是(3,1,0.1,0.5)这样的矩阵，我们当然可以找到里边的最大值“3”，从而找到对应的分类为 I，但是这并不直观。我们想让最终的输出为概率，也就是说可以生成像(90%,5%,2%,3%)这样的结果，这样做不仅可以找到最大概率的分类，而且可以知道各个分类计算的概率值。

我们将输出结果正规化处理的层叫做“Softmax”层。通过 Softmax 层之后，我们得到了 I，II，III 和 IV 这四个类别分别对应的概率。

但是，对于(90%,5%,2%,3%)这样的结果，虽然可以正确分类，但真实结果是

(100%,0,0,0)，其与真实结果之间还是有差别的，我们需要将 Softmax 输出结果的好坏程度做一个“量化”。一种直观的解决方法，是用 1 减去 Softmax 输出的概率，比如 1-90%=0.1。不过更为常用且巧妙的方法是，求对数的负数。还是用 90%举例，对数的负数就是：-log0.9=0.046。可见，概率越接近 100%，该计算结果值越接近于 0，说明结果越准确，该输出叫做“交叉熵损失（Cross Entropy Error）”。

我们训练神经网络的目的，就是尽可能地减少这个“交叉熵损失”。

- 反向传播

算出交叉熵损失后，就要开始反向传播了。其实反向传播就是一个参数优化的过程，优化对象就是网络中的所有 W 和 b（因为其它所有参数都是确定的）。模型训练时，参数的数量有时会上亿，其优化的原理和我们这个两层神经网络是一样的。

神经网络需要反复迭代。如上述例子中，第一次计算得到的概率是 90%，交叉熵损失值是 0.046；将该损失值反向传播，使 W1,b1,W2,b2 做相应微调；再做第二次运算，此时的概率可能就会提高到 92%，相应地，损失值也会下降，然后再反向传播损失值，微调参数 W1,b1,W2,b2。依次类推，损失值越来越小，直到我们满意为止。

此时我们就得到了理想的 W1,b1,W2,b2。

1.2 MLPClassifier 算法介绍

多层感知器（Multilayer Perceptron，简称 MLP）是最基本的人工神经网络模型之一，结构简单，易于理解和实现。可以应用于分类、回归等多种任务。学习神经网络，建议先从 MLP 入手，逐渐深入学习其它类型的神经网络，比如卷积神经网络（Convolutional Neural Networks，简称 CNN）和循环神经网络（Recurrent Neural Networks，简称 RNN），它们分别用于计算机视觉和自然语言处理等特定领域的问题。

MLP 神经网络属于前馈神经网络（Feedforward Neural Network）的一种。在网络训练过程中，需要通过反向传播算法计算梯度，将误差从输出层反向传播回输入层，用于更新权重参数。除了 MLP，其它常见的前馈神经网络包括卷积神经网络（CNN）和循环神经网络（RNN）等。

MLPClassifier 是一种基于多层感知器(MLP)的分类器，可以用于处理各种分类问题。使用 LBFGS 算法或随机梯度下降(SGD)算法来优化损失函数。其函数形式如下：

```
MLPClassifier(solver='sgd',activation='relu',max_iter=10,alpha=1e-4,hidden_layer_sizes = (50,50),random_state = 1, learning_rate_init = .1)
```

MLPClassifier 算法的方法有：

1) fit(X_train,y_train)，拟合训练。
2) get_params（[deep]），获取参数
3) predict(x)，使用多层感知器（MLP）进行预测

4) predict_log_proba（X），就返回对数概率估计

5) predict_proba（X），概率估计

6) score（X，y [，sample_weight]），给定测试数据和标签的平均准确度

7) set_params（** params），设置参数。

sklearn 中已经集成了 MLP 的工具包

2. 实验内容及要求

1) 实验内容：改写下列程序

```
from sklearn.neural_network import MLPClassifier
x=[[0,0],[1,1],[-2,2],[-1,-2],[2,-1],[-3,-3],[3,2]]
y=[1,1,2,3,4,3,1]

clf=MLPClassifier(solver='lbfgs',alpha =1e-
5,hidden_layer_sizes=(5,5),random_state=1)

clf.fit(x,y)

X1=[[2,3],[-1,-2]]

preY1=clf.predict(X1)
print(preY1)
print(clf.predict_proba(X1))
```

实现：输出显示 MLPClassifier 算法的一些重要属性，从而深入理解人工神经网络原理。具体效果如下图：

```
第1层网络：
权重矩阵： (2, 5)
系数矩阵： [[ 0.88961078 -3.63238083 -3.35110327 -1.59751762 -1.90732862]
 [-1.29762168 -2.54525653  2.62482776 -1.231864    0.18424363]]
第2层网络：
权重矩阵： (5, 5)
系数矩阵： [[-1.7544543   1.82472644 -0.65213628 -0.72068278 -0.40046944]
 [-0.9173197   2.41276191 -0.90617516 -0.7561064   1.38585521]
 [-0.32141698 -0.4915977  -0.8450253  -0.50977988  2.2159475 ]
 [-1.80189663  1.94866019 -0.37325009 -0.34821791  1.3696135 ]
 [-0.38328715  0.38988703  0.33047946 -0.74376005  1.41000211]]
第3层网络：
权重矩阵： (5, 4)
系数矩阵： [[ 2.91246136 -1.17003117 -0.70156817 -1.14196552]
 [-1.73260847 -2.89305328  1.10247546  1.96054494]
 [-0.47012996 -0.05367867 -0.27586801 -0.20189471]
 [ 0.4277691  -0.43884194 -0.20167579 -0.53895569]
 [-1.27282923  2.11978898  0.49205375 -2.03463053]]
[1 2 3 4]
0.00011160715076186461
[array([-0.14020761, -0.46643899, -1.64671579,  0.03976439, -0.25112708]), array([ 2.24578927, -0.0042807 , -0.82342525,
18
[1 2]
[[9.99988686e-01 1.21846437e-06 3.07665494e-07 9.78742706e-06]
 [2.52684380e-24 1.00000000e+00 4.97139735e-14 4.12132123e-31]]
```

显示每一层神经网络的权重矩阵大小、系数矩阵

输出类标签

损失值

截距

迭代次数

2） 上传要求

(1) 运行效果图

(2) 源程序文件