项目二：商品分类

2.1 项目介绍与分析

Otto Group 是世界上最大的电子商务公司之一，在全世界范围内，它每天会卖出数百万件商品。每件商品会卖出数百万件商品。每件商品所属的类别分别是 Class1~Class9。对于这家公司来说，货物的供给和需求分析是十分重要的信息。现已有商品的一些特征，要求设计一个算法模型来判断一个商品所属的类别。

本项目是一个典型的多分类问题，商品从属 9 类，给定的数据集中训练集样本共有 49502 条，每条样本存在 93 维特征，数据量相对可观。因此，我组集体考虑采用了**多层感知机（Multi-Layer Perceptron，MLP）**模型解决这一问题。

2.2 MLP 模型

大脑是由生物神经元构成的巨型网络，它在本质上不同于计算机，是一种大规模的并行处理系统，它具有学习，联想记忆，综和等能力，并有巧妙处理信息的方法。而人工神经网络是模拟人脑思维方式的数学模型，从微观结构和功能上对人脑进行抽象与简化，模拟人类智能。人工神经网络也是由大量的，功能比较简单的形式神经元互相连接而构成的复杂网络系统，用它可以模拟许多大脑的基本功能和简单的思维方式。

感知机是最早被设计并被实现的人工神经网络。感知机是一种十分特殊的神经网络。其有两层神经元组成，是结构最为简单的人工神经网络，如图 1 所示

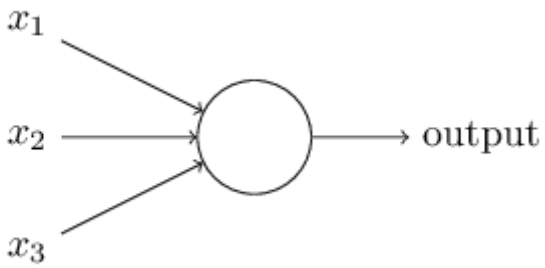

图 1.三输入神经元的感知机网络结构示意图

输入层接受外界输入信号后传递给输出层，输出层是 M-P 神经元，也称“阈值逻辑单元”。感知机能够容易地实现逻辑与、或、非运算。输出 $y = f(\sum_i w_i x_i - \theta)$，假定函数 f 是单位阶跃函数，即函数值大于等于零时输出为 1，反之输出为 0。

（1） **“与”（$x_1 \& x_2$）：** 令$w_1 = w_2 = 1, \theta = 2$,则$y = f(1 * x_1 + 1 * x_2 - 2)$。可知只有在输入同时为 1 时，输出 y 为 1，当任一输入为零时，感知机的输出均为 0，这符合“与”运算的操作。

（2） **“或”（$x_1 || x_2$）：** 当$w_1 = w_2 = 1, \theta = 0.5$，则$y = f(1 * x_1 + 1 * x_2 - 0.5)$，

当任一输入为 1 时，输出均为 1，当输入均为 0 时，输出方为 0，这显然符合“或”操作运规律。

（3） **“非”：** 当$w_1 = -0.6, w_2 = 0, \theta = -0.5$，则$y = f(-0.6 * x_1 + 0.5)$，当输入为 1 时输出为 0，输入为 1 时输出为 0。

上述问题之所以能够使用感知机将其模拟出来，原因是上述的在二维平面上是线性可分的，即存在一条直线（平面或超平面）将它们分开。

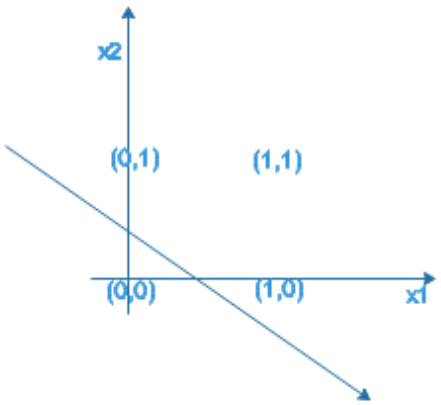

图 2.“或”问题

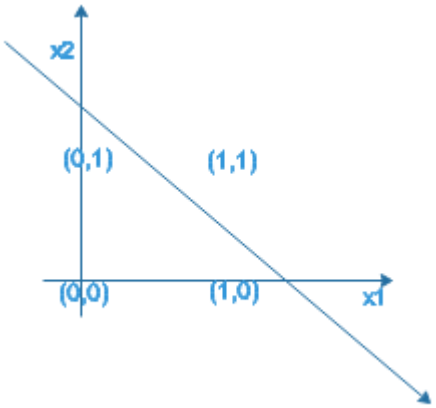

图 3.“与问题

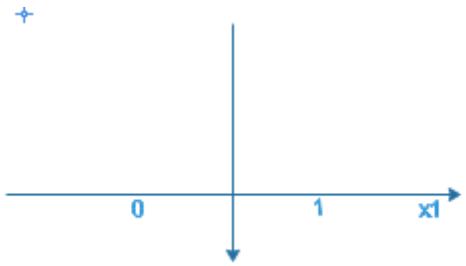

图 4.“非”问题

但要注意的是在存在着先线性可分的同时必然会出现线性不可分的情况，这时感知机对于此类情况显得无能为力。例如“异或”问题，在输入为全 1 或者全 0 时输出为 0，反之则会输出 1。此类情况就属于线性不可分。其平面图如图所示：

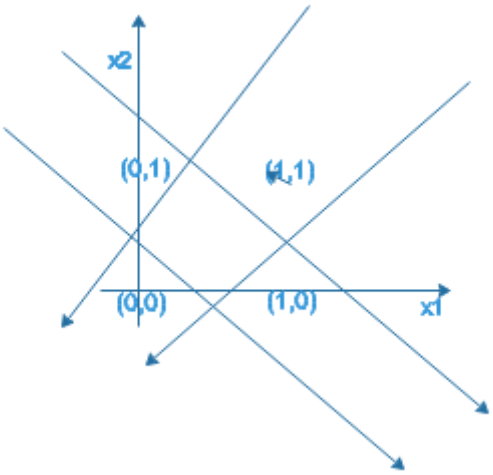

图 5.“异或”问题

此时在平面上找不出一条直线可以将两类完美地分开。要解决非线性可分的问题，需考虑使用多层功能神经元，即多层感知机。这是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP 可以被看做是一个有向图，由多个节点层组成，每一层全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）。一种被称为反向传播算法的监督学习方法常被用来训练 MLP。MLP 是感知机的推广，克服了感知机无法实现对线性不可分数据识别的缺点。

最简单的 MLP 是三层结构（输入层-隐含层-输出层），其结构如图 6 所示：

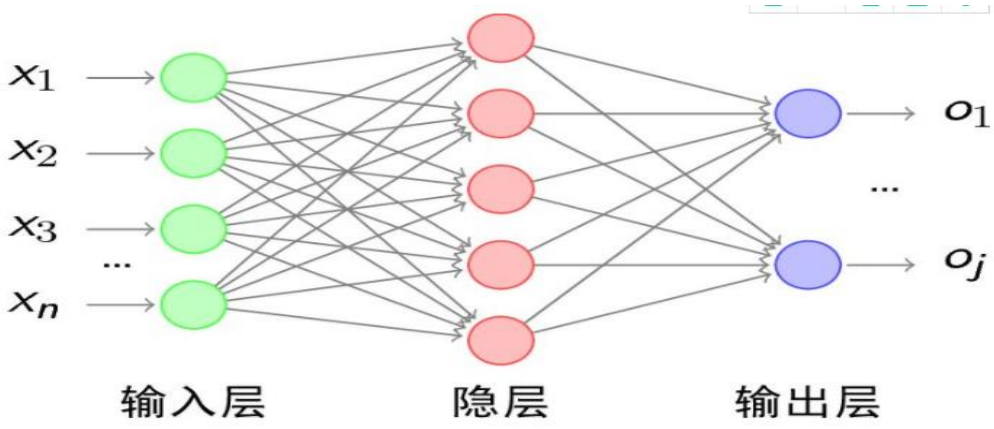

图 6.三层神经网络

从输入层到隐层，其中用 **X** 代表输入，h 代表隐含层输出，则

$$h = \sigma(\boldsymbol{W_1 X} + B_1)$$

式中W_1代表权重，B_1代表偏置。函数f通常为非线性，称为激活函数。

2.3 激活函数

下面给出了几种常用的激活函数及其图像

（1） Sigmoid 函数

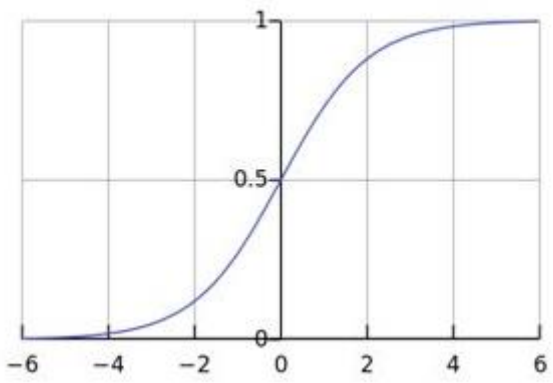

图 7.sigmoid 函数

（2） 双曲正切函数

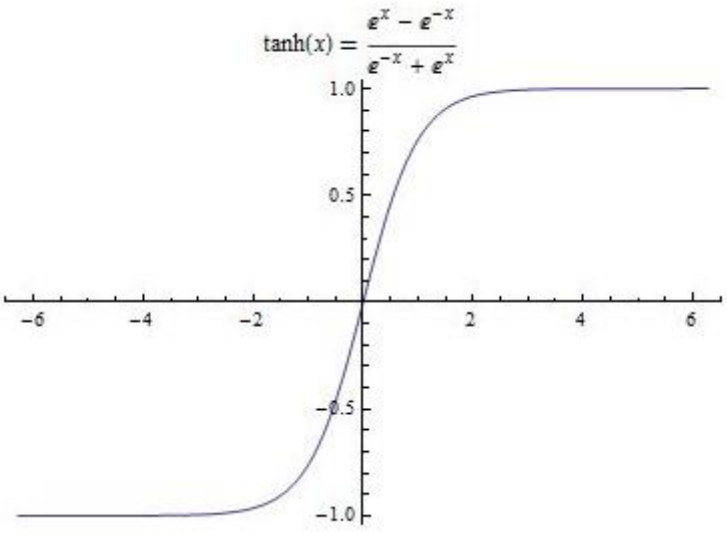

图 8.双曲正切函数

（3） softmax 函数

softmax 函数也称作归一化指数函数，多用于多分类问题，其定义如下：

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1,2, \dots \dots K$$

本项目要求模型输出商品属于各类的概率，所以多层感知机的最后一层选择的式 softmax 函数作为激活函数，以使得输出结果满足 Lentcode 网站的要求，从而参与排名。

2.4 BP 算法

通过构建多层感知机，就可以得到输入输出关系，然后需要通过监督学习求得W_1, W_2, B_1, B_2的最佳参数估计值。通常是利用 **反向传播（Back Propagation，BP）算法** 和最优化算法中的 **梯度下降法** 对权重进行迭代更新，直到满足某个条件为止。

BP 算法的基本思想是：反向传播时，将输出误差(期望输出与实际输出之差)按原通路反传计算，通过隐层反向，直至输入层，在反传过程中将误差分摊给各层的各个单元，获得各层各单元的误差信号，并将其作为修正各单元权值的根据。

这一计算过程使用梯度下降法完成，在不停地调整各层神经元的权值和阈值后，使误差信号减小到最低限度。以三层 BP 网络为例，各层分别为输入层（I），隐含层（H），输出层（O）。虽然只有三层，但足以完成任意的 m 维到 n 维的映射，示例网络如图 9 所示：

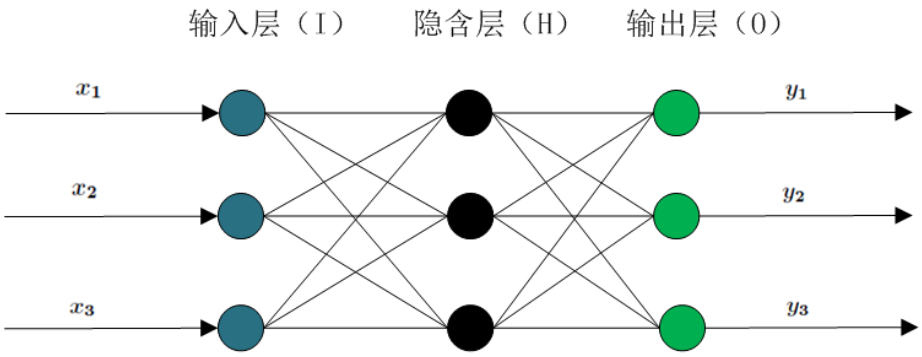

图 9 三层 BP 网络示意图

在 BP 神经网络中，输入层和输出层的节点个数是确定的，而隐含层节点个数不确定。隐含节点数目可以由经验公式确定：

$$\mathrm{h} = \sqrt{m+n} + a$$

其中 h 为隐含层节点的数目，m 为输入层节点数目，n 为输出层节点数目，a 为 1 到 10 之间的调节常数。

BP 算法由信号的正向传播和误差的反向传播两个过程组成。

（1） 正向传递子过程

设节点 i 和节点 j 之间的权值为 w_{ij}，节点 j 的阈值为 b_j,每个节点的输出值

为x_j，而每个节点的输出值是根据上层所有节点的输出值，当前节点与上一层所有的权值和当前节点的阈值还有激活函数来实现的，具体计算法如下：

$$S_j = \sum_{i=0}^{m-1} w_{ij} x_i + b_j$$

$$x_j = f(S_j)$$

其中 f 为激活函数，一般选取 s 型函数或者线性函数，输入层节点没有阈值。

（2） 反向传递子过程

在 BP 神经网络中，误差信号传递子过程是基于 Widrrow-Hoff 学习规则的，假设输出层的所有结果为d_j,误差函数如下：

$$E(w,b) = \frac{1}{2}\sum_{j=0}^{n-1}(d_j - y_j)^2$$

BP 神经网络是通过反复修正权值和阈值使得误差函数值达到最小。Widrow-Hoff 学习规则是通过沿着相对误差平方和的最快速下降方向，连续调整网络的权值和阈值，根据梯度下降法，权值矢量的修正正比于当前位置上 E(w,b)的梯度，对于第 j 个输出点有：

$$\Delta w(i,j) = -\eta \frac{\partial E(w,b)}{\partial w(i,j)}$$

选取激活函数为：

$$f(x) = \frac{A}{1+e^{-\frac{\infty}{B}}}$$

对 f(x)求导得到：

$$f'(x) = \frac{Ae^{\frac{-\infty}{B}}}{B(1+e^{-\frac{\infty}{B}})^2} = \frac{f(x)[A-f(x)]}{AB}$$

对w_{ij}求导有：

$$\frac{\partial E(w,b)}{\partial w_{ij}} = \frac{1}{\partial w_{ij}} \cdot \frac{1}{2}\sum_{j=0}^{n-1}(d_j - y_j)^2$$

$$= (d_j - y_j) \cdot \frac{f(S_j)[A-f(S_j)]}{AB} \cdot \frac{\partial S_j}{\partial w_{ij}}$$

$$= \delta_{ij} \cdot x_i$$

$$\delta_{ij} = (d_j - y_j) \cdot \frac{f(S_j)[A - f(S_j)]}{AB} \quad , \mathrm{x}_i = \frac{\partial S_j}{\partial w_{ij}}$$

根据梯度下降法，对于隐含层和输出层的权值和阈值调整如下：

$$\mathrm{w}_{ki} = w_{ki} - \eta_1 \cdot \frac{\partial E(w,b)}{\partial w_{ki}} = w_{ki} - \eta_1 \cdot \delta_{ki} \cdot x_k$$

$$\mathrm{b}_i = b_i - \eta_2 \cdot \frac{\partial E(w,b)}{\partial b_i} = b_i - \eta_2 \cdot \delta_{ki}$$

2.5 模型实现

2.5.1 基本实现

根据以上理论分析，需要构建的 MLP 模型包括：1 层输入层、若干隐含层和 1 层输出层。考虑到过拟合问题，需要进行正则化，即调整惩罚因子。隐含层和惩罚因子的选择不同都会导致模型预测的精度有所差异。基本模型实现依靠经验对二者进行取值，具体步骤如下：

第一步：读取数据集，将输入特征与输出结果分离，再将输入数据标准化；
第二步：创建 MLP 分类器，其中惩罚因子$\alpha = 1 \times 10^{-5}$，隐含层取 5 层，各层节点数分别是 90，70，50，30，10；

```
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(90,70,50,30,10), random_state=1)
clf.fit(train_X, train_Y)
r = clf.score(train_X, train_Y)
print("R值(准确率):", r)  # R值(准确率): 0.8149771726394893
```

图 10 模型构建

第三步：输出训练数据集，即特征矩阵和输出进行模型训练；
第四步：标准化测试集数据后，利用 MLP 模型进行分类预测；
第五步：将预测结果写入需提交的模板文件，上传 Lentcode 进行评分和排名。

2.5.2 算法提升

利用程序生成一些组特定的隐层的数目，在这些组隐层数目中让模型去决定哪一组是最优的参数。主要用到 GridSearch 网格搜索方法。

```
hl_sizes = []
for i in range(10):
    n = nd // 10 -1
    item = []
    for j in range(1, n):
        p = np.random.rand()
        if p < 0.6:
            a = 5+10*j
            b = a+10*j+1
            if b > nd-5:
                break
            subitem = np.random.randint(a, b)
            if subitem not in item:
                item.append(subitem)
    if item and len(item)>2:
        hl_sizes.append(sorted(item, reverse=True))
hl_sizes.append([80, 60, 40, 20])
```

图 11.生成隐层数目组

最终生成的的隐层数目组为

```
[50, 46, 42, 22],
 [79, 28, 23],
 [74, 63, 37, 20],
 [78, 52, 43, 19],
 [80, 60, 40, 20]]
```

图 12.隐层数目组

然后使之在给定的参数组合下找出能够训练出最佳模型的参数组合。

```
# 构建模型
clf = MLPClassifier(solver='lbfgs', random_state=1)
# 自动调参
param_grid = {'alpha':[3, 2, 1, 1e-1, 1e-2],'hidden_layer_sizes':hl_sizes}
grid_search = GridSearchCV(clf, param_grid, n_jobs = 1, verbose=10)
grid_search.fit(train_X[5000:6000,:], train_Y[5000:6000])
alpha, hl_sizes = grid_search.best_params_['alpha'], grid_search.best_params_['hidden_layer_sizes'];alpha,hl_sizes
```

图 13.自动调参

最终找出的最佳参数为(0.1, [74, 63, 37, 20])，其中 0.1 为惩罚因子，[74,63,37,20]为隐含层的节点数目。由此构建 MLP 分类器进行商品预测。

比较两次提交的结果，未调参时提交得分为 0.60535，排名第十七；使用网格搜索自动调参后得分为 0.57771，排名第十三。

2.5.3 降维考虑

由于原始数据所形成的矩阵包含的零数量个数远大于非零数目，所以该矩阵是稀疏矩阵，可以利用 PCA 适当降维降低矩阵的稀疏性。我们决定利用 PCA 的方法将数据降维，保留原始数据 95%的信息量。通过降维后，我们将原来的 93 个特征降至 57，将少了大约一半的维数。从而使得隐含层结构得以简化，模型训练速度大幅提升，预测处理性能也得以加强。

利用降维后的数据并通过网格搜索进行自动调参，又一次训练出新的模型，利用新的模型进行预测，提交 lentcode 后发现分类效果有所减弱，该模型得分为 1.23000，排名第 25。小组讨论后得出结果是降维造成了数据损失，造成了预测结果不准确。

2.6 PCA 降维

PCA（principal components analysis）即主成分分析技术，主要是在一定的信息损失范围内，把多指标转化为少数几个综合指标，经常用于减少数据集的维数，同时保持数据集的对方差贡献最大的特征。

主要思想是通过线性变换把数据变换到一个新的坐标系统中，使得任何数据投影的第一大方差在第一个坐标(称为第一主成分)上，第二大方差在第二个坐标(第二主成分)上，依次类推。

PCA 降维的优点包括：① 使得数据集更易使用；② 降低算法的计算开销；③ 去除噪声；④ 使得结果容易理解。

数据降维以向量形式表示为：

$$\boldsymbol{X'} = \boldsymbol{XP}$$

式中：

- $\boldsymbol{X} = [\boldsymbol{x_1}\ \boldsymbol{x_2}\ \cdots\ \boldsymbol{x_m}]^{\boldsymbol{T}}$，$\boldsymbol{X}\ \boldsymbol{\epsilon}\ \mathbb{R}^{\boldsymbol{m\times n}}$ 表示原特征矩阵，共 m 条样本，每条样本维度为n；
- $\boldsymbol{X'} = [\boldsymbol{x_1}\ \boldsymbol{x_2}\ \cdots\ \boldsymbol{x_m}]^{\boldsymbol{T}}$，$\boldsymbol{X'}\ \boldsymbol{\epsilon}\ \mathbb{R}^{\boldsymbol{m\times d}}$ 表示降维后的特征矩阵，共 m 条样本，每条样本维度为d，显然$d < m$；
- 可以证明，其中的 $\boldsymbol{P}$ 矩阵是由样本特征矩阵 $\boldsymbol{X}$ 的协方差矩阵的最大的 d 个特征值对应的单位正交的特征向量组成，即：

$$\boldsymbol{P} = [\boldsymbol{v_1}\ \boldsymbol{v_2}\ \cdots \boldsymbol{v_d}], \qquad \boldsymbol{P}\ \boldsymbol{\epsilon}\ \mathbb{R}^{\boldsymbol{n\times d}}$$

PCA 降维的伪代码实现如下：

（1） 将原始数据按列组成 m 行 n 列矩阵 X；
（2） 将 X 的每一行(代表一个属性字段）进行零均值化；
（3） 计算协方差矩阵$\mathrm{C} = \frac{1}{\mathrm{m}}\mathrm{X}\mathrm{X}^{\mathrm{T}}$；
（4） 计算协方差矩阵的特征值及对应的特征向量；
（5） 最大的 d 个特征值对应的单位正交的特征向量组成矩阵 P，其中 d 个特征值的贡献率在 95％以上；
（6） 根据$\boldsymbol{X'} = \boldsymbol{XP}$计算得到降维后的矩阵。