ML Lab 5

综合实验

王世炟

PB20151796

January 16, 2023

Part 1: 实验要求

本次实验考察完整进行实验的能力。具体要求是通过完整的实验流程，分析所提供的数据，解决多分类问题。具体步骤如下：

- **数据预处理**。需要综合运用所学的知识进行数据降维、降噪、补缺、特征提取、编码以及必要的其他数据预处理工作。
- **数据划分**。需要将所提供的 train 数据集按照所学的方法拆分成训练集以及测试集。
- **模型训练**。需要分别使用本课程所学习的线性回归模型、决策树模型、神经网络模型、支持向量机以及 XGBoost 等分类模型来完成标签预测任务。
- **模型验证**。需要将 test_feature.csv 的数据输入到一个你认为性能最佳的模型中，然后仿照 train_label.csv 的文件格式生成对应标签数据文件。
- **实验分析**。需要仔细撰写实验报告以及相关分析。

Part 2: 数据集介绍

本次实验的数据集分别为 train_feature.csv,train_label.csv,test_feature.csv。
其中 train_feature.csv,train_label.csv 为本次的训练数据集，train_feature.csv 中有 10000 条数据，每个数据有 120 个特征;train_label.csv 中有 10000 个对应类别，共有 0,1,2,3 四个类别；train_label.csv 中为 3000 条待分类的数据。

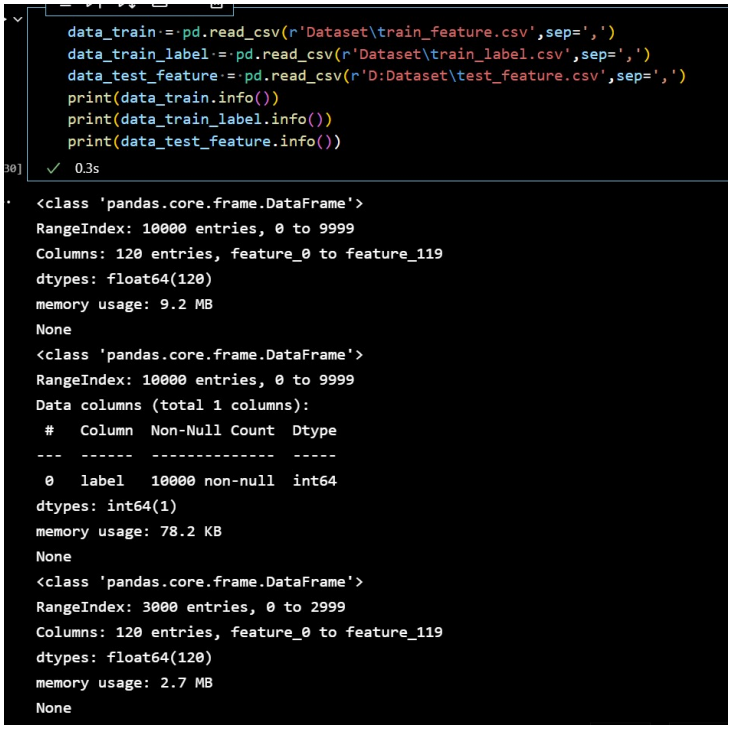

图 1: Dataset

可以看出，Aggregation 数据集大致可分为 7 个簇，D31 数据集大致可分为 31 个簇，R15 数据集大致可分为 15 个簇。

Part 3: 算法简介

本次实验用到线性回归模型、决策树模型、神经网络模型、支持向量机以及 XGBoost 共 5 种模型，先对他们做简要的介绍：

3.1 线性回归模型

Logistics Regression 模型中, 利用了 sigmoid 函数来估计概率

$$P(\boldsymbol{Y}=1)=\frac{1}{1+e^{\boldsymbol{X\beta}}}$$

似然函数:

$$\mathcal{L}(\beta)=\prod_{i=1}^{n}P(y_i|x_i,\beta)=\prod_{i=1}^{n}\left(\frac{1}{1+e^{-x_i\beta}}\right)^{y_i}\left(1-\frac{1}{1+e^{-x_i\beta}}\right)^{1-y_i}$$

对其取对数即得到对数似然函数:

$$\log\mathcal{L}(\beta)=\sum_{i=1}^{n}\left[y_i\log\left(\frac{1}{1+e^{-x_i\beta}}\right)+(1-y_i)\log\left(1-\frac{1}{1+e^{-x_i\beta}}\right)\right]$$

我们可以将它的相反数并取平均值当做损失函数:

$$J(\beta) = -\frac{1}{n}\log \mathcal{L}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(P(y_i)\right) + (1-y_i)\log\left(1-P(y_i)\right)\right]$$

为了用梯度法优化参数, 应当将损失函数对参数 β 求导:

$$\frac{\partial J(\beta)}{\partial \beta_j} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i,\beta))\cdot x_{ij} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-x_i\beta}} - y_i\right)\cdot x_{ij}$$

然后根据梯度下降法的原理:

$$\beta_{t+1} \leftarrow \beta_t - \alpha\nabla J(\beta)$$

即可进行迭代优化。

3.2 决策树模型

这里直接给出决策树模型的算法:

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
属性集 $A = \{a_1, a_2, \ldots, a_d\}$.
过程: 函数 TreeGenerate(D, A)
生成结点 node;
if D 中样本全属于同一类别 C **then**
  将 node 标记为 C 类叶结点; **return**
end if
if $A = \varnothing$ **OR** D 中样本在 A 上取值相同 **then**
  将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
end if
从 A 中选择最优划分属性 a_*;
for a_* 的每一个值 a_*^v **do**
  为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
  if D_v 为空 **then**
    将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
  else
    以 TreeGenerate(D_v, $A \setminus \{a_*\}$)为分支结点
  end if
end for
输出: 以 node 为根结点的一棵决策树

图 2: 决策树模型算法

3.3 神经网络模型

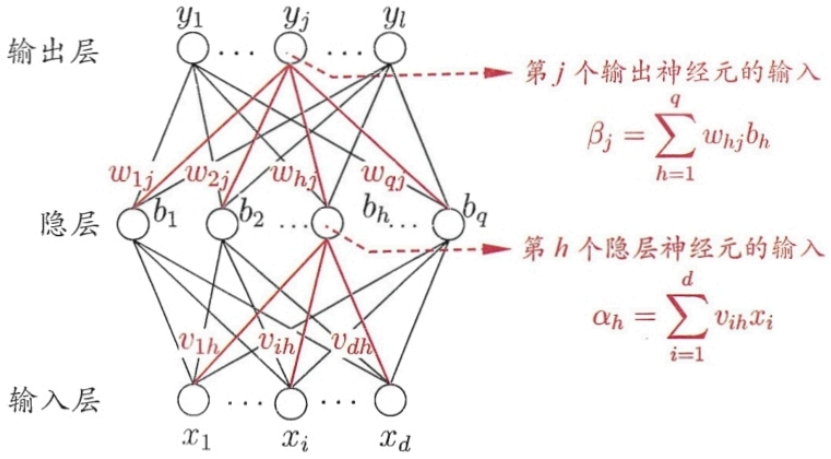

图 3: BP 网络及算法中的变量符号

在神经网络中计算误差，然后利用逆误差传播算法更新网络中的参数，直至达到收敛条件停止更新，

输入: 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$;
学习率 η.

过程:

在$(0,1)$范围内随机初始化网络中所有连接权和阈值
repeat
  for all $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
    根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
    根据式(5.10) 计算输出层神经元的梯度项 g_j;
    根据式(5.15) 计算隐层神经元的梯度项 e_h;
    根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
  end for
until 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络

图 4: 逆误差传播算法

3.4 支持向量机

软间隔 SVM 标准问题:

$$\begin{aligned} min_{w,b,\xi} \quad & \frac{1}{2}\boldsymbol{w^T w} + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i(\boldsymbol{w^T \phi(x_i}+b)) > 1-\xi_i, \quad i=,2,\cdots,m \\ & \xi_i \geq 0, \quad i=,2,\cdots,m \end{aligned}$$

使用合页损失的软间隔 SVM:

$$min_{w,b} \quad \frac{1}{m}\sum_{i=1}^{m} max(0, 1-y_i(\boldsymbol{w^T x_i}+b)) + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

是个无约束问题，可直接采用梯度下降法求解。
对偶问题:

$$\begin{aligned} min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{m}\alpha_i \\ s.t. \quad & 0 \leq \alpha_i \leq C, \quad i=1,2,\cdots,m \\ & \sum_{i=1}^{m}\alpha_i y_i = 0 \end{aligned}$$

在之前的实验中实现了两种方法解决 svm 问题，分别是 Pegasos 算法和 DCD 算法，此处不在赘述。

3.5 XGBoost

XGBoost 是 boosting 族中的算法，遵从前向分步加法，是由多个基模型组成的一个加法模型，假设第 k 个基本模型是 $f_k(x)$, 那么前 t 个模型组成的模型的输出为

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

其中 x_i 为第表示第 i 个训练样本，y_i 表示第 i 个样本的真实标签; $\hat{y}_i^{(t)}$ 表示前 t 个模型对第 i 个样本的标签最终预测值。
在学习第 t 个基模型时，XGBoost 要优化的目标函数为:

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} penalty(f_k) \\
&= \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{k=1}^{t} penalty(f_k) \\
&= \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + penalty(f_t) + constant
\end{aligned}
$$

其中 n 表示训练样本的数量, $penalty(f_k)$ 表示对第 k 个模型的复杂度的惩罚项, $loss(y_i, \hat{y}_i^{(t)})$ 表示损失函数,
将 $loss(y_i, y_i^{(t-1)} + f_t(x_i))$ 在 $y_i^{(t-1)}$ 处泰勒展开可得

$$
loss(y_i, y_i^{(t-1)} + f_t(x_i))\ loss(y_i, y_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)
$$

其中 $g_i = \frac{\partial\ loss(y_i, y_i^{(t-1)})}{\partial\ y_i^{(t-1)}}$, $h_i = \frac{\partial^2 loss(y_i, y_i^{(t-1)})}{\partial\ (y_i^{(t-1)})^2}$
，即 g_i 为一阶导数，h_i 为二阶导数。
此时的优化目标变为

$$
Obj^{(t)} = \ _{i=1}^{n}[loss(y_i, y_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + penalty(f_t) + constant
$$

去掉常数项 $loss(y_i, y_i^{(t-1)})$ (学习第 t 个模型时候，$loss(y_i, y_i^{(t-1)})$ 也是一个固定值) 和 constant，可得目标函数为

$$
Obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + penalty(f_t)
$$

3.6 二分类器解决多分类问题

由二分类器实现多分类器的方法有很多，比如 OvO、OvR、MvM 等，本次实验采取 OvO 方法，下面做简要介绍：
给定数据集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}, y_i \in \{C_1, C_2, \cdots, C_N\}$.OvO 将这 N 类别两两配对，从而产生 $\frac{N(N-1)}{2}$ 个二分类任务，例如 OvO 将为区分类别 C_i 和 C_j 训练一个分类器，该分类器把 D 中的 C_i 类样例作为反例。在测试阶段，新样本同时交给所有分类器，我们将得到 $\frac{N(N-1)}{2}$ 个分类结果，最终可以通过投票产生结果。

Part 4: 实验步骤

4.1 数据预处理

首先利用 pandas 库中的 describe() 函数来查看数据的有关特征：

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	
count	9960.000000	9946.000000	9955.000000	9952.000000	9954.000000	9.955000e+03	9956.000000	9956.000000	99
mean	1381.896486	58.715384	11.157463	215.609224	104.244826	5.159948e+03	1418.185014	193.551326	
std	20342.092034	3067.018960	213.548216	3254.119426	1809.139214	8.325174e+04	21803.434424	3078.353897	10
min	26.000000	-72400.900016	0.000004	3.000000	0.000000	0.000000e+00	27.000000	3.000000	
25%	45.000000	-5.442525	0.248601	7.000000	3.000000	1.620000e+02	45.000000	7.000000	
50%	50.000000	-0.060218	0.504924	8.000000	4.000000	1.940000e+02	50.000000	8.000000	
75%	55.000000	5.482893	0.751965	8.000000	5.000000	2.220000e+02	55.000000	8.000000	
max	526876.000000	105832.015208	8448.902842	79794.000000	57936.000000	2.072000e+06	505945.000000	81060.000000	478

8 rows × 120 columns

图 5: 数据集有关特征

4.1.1 异常值处理

由上图可见，以 feature0 为例，在 75% 位置的值为 55 的情况下，max 达到 526876，这显然是有问题的，我们需要做异常值的处理。

首先，我们假设每一个特征符合正态分布。为了验证这个假设，我们采用了 Kolmogorov–Smirnov 检验来判断假设是否正确。Kolmogorov–Smirnov 检验通过计算 p-value，当其小于 0.05 时认为该特征符合正态分布。

```
from scipy.stats import kstest

flag = 1
for col in data_train.columns:
    if kstest(data_train[data_train[col].isnull()==False][col],cdf = "norm"
        ).pvalue > 0.05:
        # 在kstest api中，p_value 小于0.05代表可以视为其服从正态分布
        print(col,' 不服从正态分布,p_value=',kstest(data_train[data_train[
            col].isnull()==False][col],cdf = "norm").pvalue ,'\n')
        flag = 0
if flag:
    print("全部特征服从正态分布\n")
```

可以得到只有 feature41 不服从正态分布,p-value= 0.2665972004397823 。

为了处理方便，暂且将所有特征均视为符合正态分布。

那么，利用 3σ 原则，一个数据在 $[\mu-2\sigma,\mu+2\sigma]$ 范围内的概率为 0.9545，即我们可以将区间外的点视为离群点，将其替换为 NaN 交由缺失值处理步骤解决。

```
df_zscore=data_train.copy()
cols=data_train.columns
for col in cols:
    df_col = data_train[col]
    z_score = (df_col - df_col.mean()) / df_col.std()
    df_zscore[col] = z_score.abs() > 2 # 选择2sigma以外的作为离群点 数值分
        布在该范围内的概率为0.9545
```

```
for col in cols:
    for i in range(len(df_zscore[col])):
        if df_zscore[col][i] == True:
            data_train.loc[i,col] = np.nan
```

4.1.2 缺失值处理

通过观察数据可知，这些特征均为连续值，在处理了异常值之后选择使用平均值来填充比较合适。

```
data_train.fillna(data_train.mean(skipna=True),inplace=True)
```

4.1.3 重复值处理

如果有重复数据，相当于人为对于数据加上权重，那么会对于之后模型训练的效果有影响。所以要检验是否有重复数据，可以用 pandas 的 duplicated() 函数。
得到的结果是无重复值。

4.1.4 数据变换

数据的量纲不同，数量级差别很大，经过标准化处理后，原始数据转化为无量纲化指标测评值，各指标值处于同一数量级别，可进行综合测评分析。
采用 Z-score 法进行数据的标准化，公式为 $x_{new} = \frac{x-\bar{x}}{\sigma}$

```
cols=data_train.columns
for col in cols:
    df_col = data_train[col]
    z_score = (df_col - df_col.mean()) / df_col.std()
    data_train[col] = z_score
```

4.1.5 特征选择

首先，如果两个特征线性相关的话，那么他们可以用一个特征代替，所以先对特征两两计算相关系数，用来筛选特征。

```
x,y=np.where(data_train.corr().values>0.6)
index=[]
for i in range(len(x)):
    if x[i]==y[i]:
        continue
    else:
        if set([x[i],y[i]]) not in index:
```

```
8                 index.append(set([x[i],y[i]]))
```

得到的结果是 feature 5,69,73,92,50，feature 37，84 线性相关，所以删除特征 69，73，92，84。

然后使用 f 检验选取了前 30 个得分较高的特征。

4.2 数据集划分

采用五折交叉验证的方法来划分数据集，训练集：测试集 =4:1。

```
1 from sklearn.model_selection import KFold
2
3 x = np.arange(len(data_train))
4 kf = KFold(n_splits=5,shuffle=True)
5 for train_index, test_index in kf.split(x):
6     print(train_index,test_index)
```

4.3 模型训练

4.3.1 线性回归模型（逻辑回归）

在逻辑回归中如果迭代可以收敛，那么学习率只是影响迭代次数而不影响最终模型，所以只要找到一个可以使迭代收敛的学习率即可，此处取学习率 lr=1。

下面对于正则化系数进行调整，正则化是为了防止过拟合，让 w 尽量小，得到如下结果：

表 1: γ 取不同值时的结果

γ	0.001	0.01	0.1	1	5	10	100
accuracy	0.2595	0.2585	0.2585	0.2595	0.261	0.258	0.2615
micro-F1	0.2595	0.2585	0.2585	0.2595	0.261	0.258	0.2615

这里选择两个值都最高的 gamma=5 进行模型训练，得到五折交叉验证结果如下：

表 2: $\gamma = 5$ 时的逻辑回归五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.236	0.268	0.2465	0.2545	0.2445	0.2499
micro-F1	0.236	0.268	0.2465	0.2545	0.2445	0.2499

给出其中一折的训练过程的损失函数变化图：

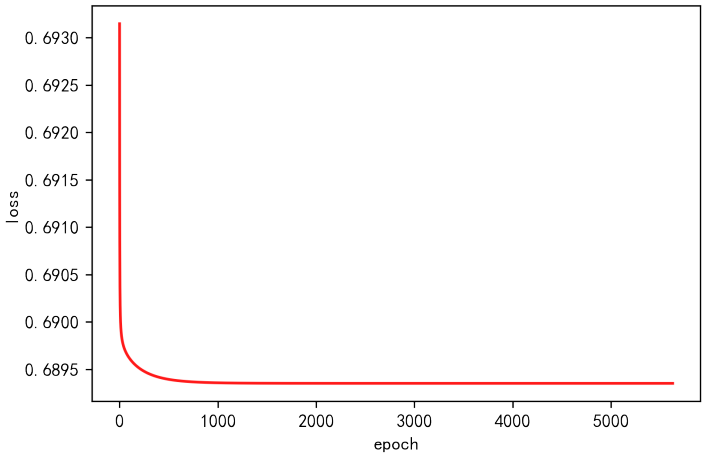

图 6: 逻辑回归 loss 曲线

4.3.2 **支持向量机**

在之前的实验中实现了 svm 的两种算法，分别是 DCD （梯度下降）和 Pegasos ，可调且有重要意义的参数仅有松弛系数 C ，这个参数影响模型对于错误分类的容忍程度，换句话说可以防止过拟合。所以对参数 C 进行调整 (DCD 算法中的 $\lambda = \frac{1}{C}$)。

表 3: λ 取不同值时 DCD 算法的结果

λ	0.001	0.01	0.1	1	10	100
accuracy	0.2535	0.253	0.2525	0.2525	0.2525	0.244
micro-F1	0.2535	0.2535	0.2525	0.2525	0.2525	0.244

选择 $\lambda = 0.001$ 进行模型训练，得到五折交叉验证结果如下：

表 4: $\lambda = 0.001$ 时的 SVM-DCD 算法五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.2415	0.237	0.249	0.2515	0.2515	0.2461
micro-F1	0.2415	0.237	0.249	0.2515	0.2515	0.2461

表 5: C 取不同值时 Pegasos 算法的结果

λ	0.001	0.01	0.1	1	10	100
accuracy	0.248	0.248	0.253	0.253	0.2515	0.2525
micro-F1	0.248	0.248	0.253	0.253	0.2515	0.2525

选择 $C = 1$ 进行模型训练，得到五折交叉验证结果如下：

表 6: $C = 1$ 时的 SVM-Pegasos 算法五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.263	0.2475	0.254	0.256	0.2375	0.2516
micro-F1	0.263	0.2475	0.254	0.256	0.2375	0.2516

4.3.3 **决策树**

这里采取的是 CART 决策树，特征划分标准选择 random，随机的在部分划分点中找局部最优的划分点，以加快训练速度。

可调参数有:

- max_depth：决策树最大深度
- min_samples_leaf：叶子节点（即分类）最少样本数

而 min_samples_leaf 将其设为数据量的 1%，所以只需要对于 max_depth：决策树最大深度进行调参即可，该参数过小会导致欠拟合，过大会过拟合，需要一个适中的值。

对于 max_depth，使其由 1 变化至 100，可得到如下结果：

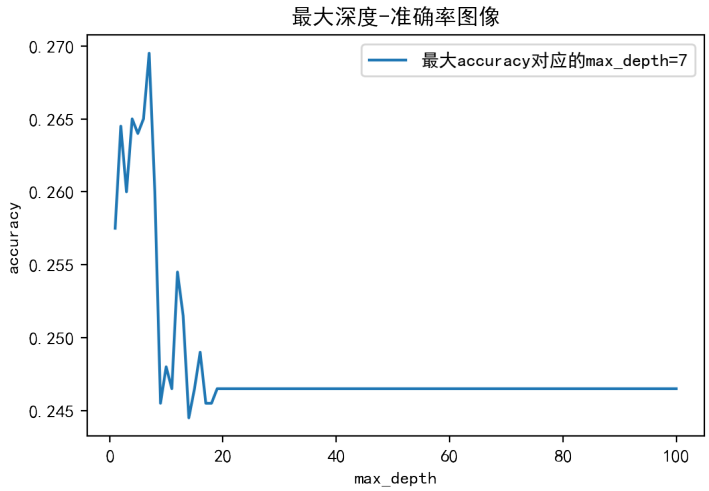

图 7: 最大深度-准确率变化曲线

可得到最佳的最大深度为 7。选择 $max\_depth = 7$ 进行模型训练，得到五折交叉验证结果如下：

表 7: $max\_depth = 7$ 时的决策树算法五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.2685	0.2415	0.253	0.2545	0.265	0.2565
micro-F1	0.2685	0.2415	0.253	0.2545	0.265	0.2565

4.3.4 **神经网络**

神经网络使用的是 sklearn.neural_network 中的 MLPClassifier,solver 参数选取的是 adam，adam 在相对较大的数据集上效果比较好，本次实验数据集有 10000 条数据，所以采用该方法；激活函数选取的是 relu 函数，它可以弥补 sigmoid 函数和 tanh 函数的缺陷：克服梯度消失的问题、加快训练速度；那么可调的参数为隐藏层的层数和每一层神经元的个数。

而在教材 P105 提到：[Hornik et al.,1989] 证明，只需要一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数。然而，如何设置隐层神经元的个数仍是个未决问题，实际中通常靠试错法（trail-by-error）调整。即我们只需要设置一层隐藏层，对神经元个数进行调参。

对神经元个数从 5-200 进行调参，可以得到如下结果：

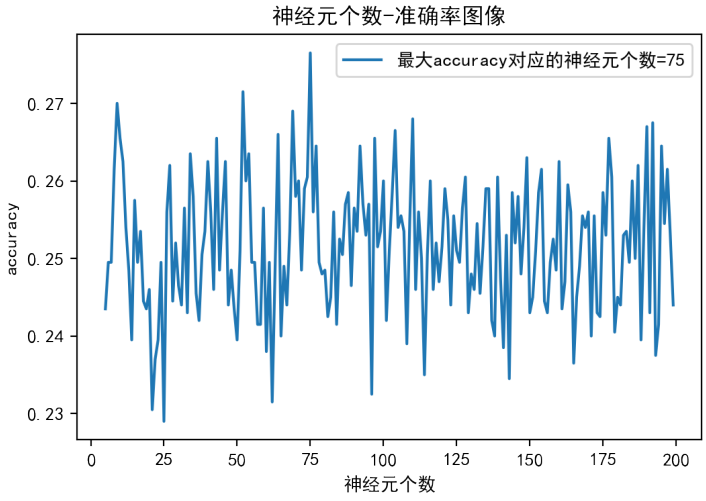

图 8: 神经元个数-准确率变化曲线

可选出最佳的神经元个数为 75，进行模型训练，得到五折交叉验证结果如下：

表 8: 神经元个数为 75 时的神经网络算法五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.249	0.2595	0.2485	0.2675	0.2365	0.2522
micro-F1	0.249	0.2595	0.2485	0.2675	0.2365	0.2522

并给出其中一折的 loss 曲线:

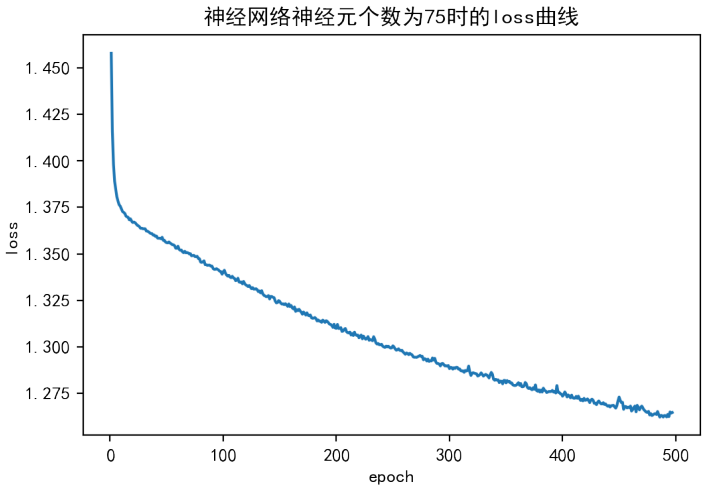

图 9

4.3.5 XGBoost **决策树**

本次实验使用的是基于分类树的 XGBoost 模型, 可调参数有:
- max_depth: 树最大深度, 这里采取与之前决策树相同的最优深度: 7
- n_estimators: 树的个数
- gamma: 惩罚项系数, 指定节点分裂所需的最小损失函数下降值。在节点分裂时, 只有分裂后损失函数的值下降了, 才会分裂这个节点。gamma 指定了节点分裂所需的最小损失函数下降值。这个参数的值越大, 算法越保守。

下面现在 gamma 为默认值的情况下对于 n_estimators 从 1 至 100 进行调参, 得到如**图** 9 结果:

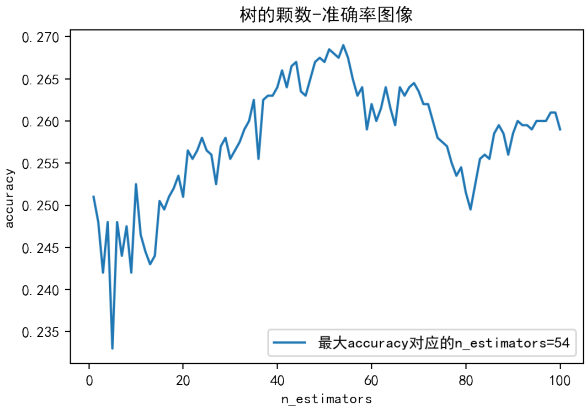

图 10: 树的颗数-准确率变化曲线

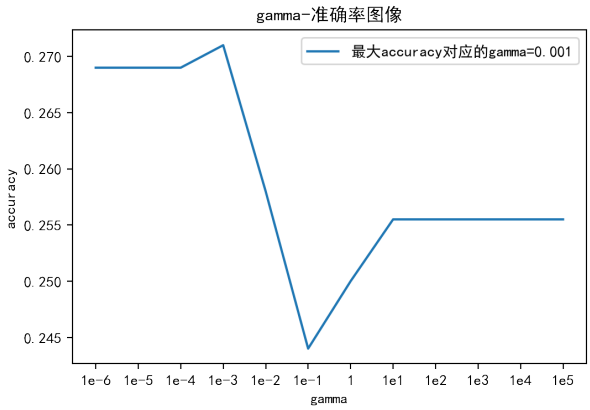

图 11: gamma-准确率变化曲线

获得了最佳树颗数 54。下面在 n_estimators=54 的情况下对 gamma 进行调整，得到如**图** 10 结果。

选出最佳的 gamma 值为 0.001 进行模型训练，得到五折交叉验证结果如下：

表 9: $gamma = 0.001, n\_estimators = 54$ 时的 XGBoost+ 分类树算法五折交叉验证结果

折数	1	2	3	4	5	average
accuracy	0.26	0.2555	0.243	0.2615	0.253	0.2546
micro-F1	0.26	0.2555	0.243	0.2615	0.253	0.2546

4.4 选择最佳模型 (假设检验)

现在对所有模型进行假设检验来选取效果最好的模型。这里采用的是 Friedman 检验与 Nemenyi 后续检验。

将上面所有模型五折交叉验证的序值表列表如下：

表 10: 所有算法五折交叉验证结果序值表

数据集 | 模型	逻辑回归	支持向量机 DCD 算法	支持向量机 Pegasos 算法	决策树	神经网络	基于分类树的 XGBoost
第一折	4	5	1	6	3	2
第二折	1	4	3	5	6	2
第三折	4	2	6	1	5	3
第四折	6	5	3	1	4	2
第五折	2	3	4	6	5	1
平均序值	3.4	3.8	3.4	3.8	4.6	2.0

计算出 τ_F = 根据教材表 2.6 查到算法个数 k=6，数据集个数 N = 5，时的临界值为 1.0578 < 2.158，可以接受“所有算法性能相同”这个假设，然后根据教材表 2.7 计算

出平均序值差别的临界值域 $CD = 3.0633$ ，并且根据平均序值的差来判断任意两个算法性能是否相同并且找出最佳模型。可以得出结果任意两个模型的性能均大体相同，所以我们取平均序值最高的 XGBoost+ 决策树模型作为最佳模型。

Part 5: 模型验证

利用选出的最佳模型 XGBoost+ 决策树预测 test_feature.csv 中的数据类别，输出到 test_label.csv 中，实验完成。

```
1 test_label.to_csv('test_label.csv',index=False)
```

Part 6: 实验总结

本次实验让我们从头开始独立完成一个数据分析机器学习项目，从数据预处理到模型选择，再到参数调整，体验了一次完整的实验过程，提升了我做实验的严谨性和思考、检索能力，受益颇深，但是由于对数据的处理不足，能力不够，无法找出和预测标签十分相关的数据。