[redacted]学生实验报告

开课学院及实验室：计算机科学与网络工程学院 513 实验室　　　　　2023 年 10 月 30 日

学院	计算机科学与网络工程学院	年级/专业/班	[redacted]	姓名	[redacted]	学号	[redacted]
实验课程名称	数据挖掘与机器学习实验					成绩	
实验项目名称	线性回归					指导老师	

实验一　线性回归

一、　实验目的

本实验课程是计算机、人工智能、软件工程等专业学生的一门专业课程，通过实验，帮助学生更好地掌握数据挖掘与机器学习相关概念、技术、原理、应用等；通过实验提高学生编写实验报告、总结实验结果的能力；使学生对机器学习模型、算法等有比较深入的认识。要掌握的知识点如下：

1. 掌握机器学习中涉及的相关概念、模型、算法；
2. 熟悉机器学习模型训练、验证、测试的流程；
3. 熟悉常用的数据预处理方法；
4. 掌握线性回归优化问题的表示、求解及编程。

二、基本要求

1. 实验前，复习《数据挖掘与机器学习》课程中的有关内容。
2. 准备好实验数据，编程完成实验内容，收集实验结果。
3. 独立完成实验报告。

三、实验软件

推荐使用 Python 编程语言（允许使用 numpy 库，需实现详细实验步骤，不允许直接调用 scikit-learn 中关于回归、分类等高层 API）。

四、实验内容：

基于 California Housing Prices 数据集，完成关于房价预测的线性回归模型训练、测试与评估。

1 准备数据集并认识数据

下载 California Housing Prices 数据集

https://www.kaggle.com/camnugent/california-housing-prices

了解数据集各个维度特征及预测值的含义

2 探索数据并预处理数据

观察数据集各个维度特征及预测值的数值类型与分布

预处理各维度特征（如将类别型维度 ocean_proximity 转换为 one-hot 形式的数值数据），参考：

https://blog.csdn.net/SanyHo/article/details/105304292

划分 70%的样本作为训练数据集，30%的样本作为测试数据集

3 求解模型参数

编程实现线性回归模型的闭合形式参数求解

编程实现线性回归模型的梯度下降参数优化

4 测试和评估模型

在测试数据集上计算所训练模型的 R^2 指标

五、实验报告

（1）线性回归闭合形式参数求解的原理

模型： $h_\theta(x)=\theta^T X$

损失函数： $J(\theta)=\|X\theta-Y\|_2^{\,2}$

目标： $$\begin{aligned}&\min J(\theta)\\&\theta=\arg\min J(\theta)\end{aligned}$$

说明: $$\begin{cases}\theta\in\mathbb{R}^{d\times 1}\\ X\in\mathbb{R}^{m\times d}\\ Y\in\mathbb{R}^{m\times 1}\end{cases}$$

正规方程形式求解，即为直接求 $J(\theta)$ 的最小值：

先展开 $J(\theta)$：

$$\begin{aligned}J(\theta)&=\|X\theta-Y\|_2^{\,2}\\&=(X\theta-Y)^T(X\theta-Y)\\&=(X^T\theta^T-Y^T)(X\theta-Y)\\&=X^T\theta^T X\theta-Y^T X\theta-Y^T X\theta+Y^T Y\\&=X^T\theta^T X\theta-2Y^T X\theta+Y^T Y\end{aligned}$$

对 $J(\theta)$ 进行求导：

$$\begin{aligned}\frac{\partial J(\theta)}{\partial\theta}&=\frac{\partial X^T\theta^T X\theta-2Y^T X\theta+Y^T Y}{\partial\theta}\\&=2X^T X\theta-2Y^T X\end{aligned}$$

令 $J(\theta)=0$ 得：

$$\begin{aligned}2X^T X\theta-2Y^T X&=0\\X^T X\theta&=Y^T X\\\theta&=(X^T X)^{-1}Y^T X\end{aligned}$$

上述结果即为求解结果，需要说明的是：特征矩阵 ***X*** 不满秩（即存在特征间的线性相关性），则正规方程求解过程中的矩阵求逆操作

可能会导致数值不稳定性。

（2）线性回归梯度下降参数求解的原理

模型：$h_\theta(x)=\sum_{i=1}^{d}\theta_i x_i$　　注：x_i 表示 x 的第 i 维

损失函数：$J(\theta)=\frac{1}{2m}\sum_{j=0}^{m}\left(y^j-h_\theta(x^j)\right)^2$　　注：x^j 表示第 j 个样本

目标：$\begin{array}{l}\min J(\theta)\\ \theta=\arg\min J(\theta)\end{array}$

说明：$\begin{cases}\theta\in\mathbb{R}^d\\ x\in\mathbb{R}^d\\ y\in\mathbb{R}^m\end{cases}$

损失函数 $J(\theta)$ 是一个关于参数 θ 的二次型，对 $J(\theta)$ 进行展开：

$$\begin{aligned}J(\theta)&=\frac{1}{2m}\sum_{j=0}^{m}\left(y^j-h_\theta(x^j)\right)^2\\&=\frac{1}{2m}\sum_{j=0}^{m}\left(y^j-\sum_{i=1}^{d}\theta_i x_i{}^j\right)^2\end{aligned}$$

对 $J(\theta)$ 进行偏微分求导运算得到：

$$\begin{aligned}\partial\frac{J(\theta)}{\partial\theta_i}&=\frac{\partial}{\partial\theta_i}\frac{1}{2m}\sum_{j=0}^{m}\left(y^j-\sum_{i=1}^{d}\theta_i x_i{}^j\right)^2\\&=\frac{1}{m}\sum_{j=0}^{m}\left(y^j-\sum_{i=1}^{d}\theta_i x_i{}^j\right)(-x_i{}^j)\\&=\frac{1}{m}\sum_{j=0}^{m}\left(\sum_{i=1}^{d}\theta_i x_i{}^j-y^j\right)x_i{}^j\end{aligned}$$

每次根据梯度更新参数：

$$\begin{aligned}\theta_i&=\theta_i-\alpha\partial\frac{J(\theta)}{\partial\theta_i}\\&=\theta_i-\alpha\left(\frac{1}{m}\sum_{j=0}^{m}\left(\sum_{i=1}^{d}\theta_i x_i{}^j-y^j\right)x_i{}^j\right)\\&=\theta_i+\alpha\frac{1}{m}\sum_{j=0}^{m}\left(y^j-\sum_{i=1}^{d}\theta_i x_i{}^j\right)x_i{}^j\end{aligned}$$

梯度下降法：

Repeat until convergence {

$$\theta_i := \theta_i + \alpha \frac{1}{m} \sum_{j=0}^{m} \left(y^j - \sum_{i=1}^{d} \theta_i x_i^j \right) x_i^j$$

}

（3）程序清单

0.导包

```
# 导入所需的包
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
import time

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

1.读取数据集

```
# 读取数据
df = pd.read_csv("./housing.csv")
# 预览数据
print(df.head())

print(df.info())
```

2.数据探索

1）查看数据的分布情况

```
i = 2
fig, ax = plt.subplots(3, 3, figsize=(18, 18))

# 散点图看看价格/房子分布
sns.scatterplot(data=df, x="longitude", y="latitude",
size="median_house_value", hue="median_house_value", ax=ax[0][0])
```

```python
# 直方图，正态否？
for col in df.columns:
    if col == "longitude" or col == "latitude" or col == "ocean_proximity":
        continue
    sns.histplot(data=df[col], bins=60, kde=True, ax=ax[(i - 1) // 3][(i - 1) % 3])
    i = i + 1

# 分类变量
sns.countplot(data=df,x="ocean_proximity",ax=ax[2][2])
```

2）研究数据之间的关系

距海远近 VS 房龄

```python
sns.displot(data=df,x="housing_median_age",hue="ocean_proximity",kind="kde")

# housing_median_age 描述性统计
df.groupby('ocean_proximity')['housing_median_age'].describe()
```

距海远近 VS 价格

```python
sns.displot(data=df,x="housing_median_age",hue="ocean_proximity",kind="kde")

# median_house_value 描述性统计
df.groupby('ocean_proximity')['median_house_value'].describe()
```

连续型 VS 连续型

```python
sns.pairplot(data=df.select_dtypes(include='float64'), kind='reg',
diag_kind='hist')
plt.savefig("1.png")

# 计算变量之间的相关系数,皮尔逊相关系数展示线性相关关系
df_corr = df.select_dtypes(include='float64').corr()
df_corr

# 绘制热力图
sns.heatmap(df_corr, cmap="Blues")
```

3.数据预处理

1）处理缺失值

```
# 取出有缺失值的列
# reshape 是为了适应 sklearn 要求
total_bedrooms = df.loc[:, "total_bedrooms"].values.reshape(-1, 1)

# 复制一份不破坏原数据
filled_df = df.copy()

# 中位数填补
filled_df.loc[:, "total_bedrooms"] =
SimpleImputer(strategy="median").fit_transform(total_bedrooms)

# 看一下效果
filled_df.info()
```

2）编码

```
# 编码
code = OneHotEncoder().fit_transform(filled_df.loc[:,
"ocean_proximity"].values.reshape(-1, 1))

# 合并
coded_df = pd.concat([filled_df, pd.DataFrame(code.toarray())], axis=1)

# 删除原列
coded_df.drop(["ocean_proximity"], axis=1, inplace=True)

# 改下表头
coded_df.columns = list(coded_df.columns[:-5]) + ["ocean_0", "ocean_1",
"ocean_2", "ocean_3", "ocean_4"]
# coded_df.columns = coded_df.columns.astype(str)

# 看看效果
coded_df.head(10)
```

3）划分训练集、测试集 7：3

```
feature = coded_df.iloc[:, :8].join(coded_df.iloc[:, -5:])
label = coded_df["median_house_value"]
```

```python
Xtrain,Xtest,Ytrain,Ytest =
train_test_split(feature,label,test_size=0.3)

Xtrain.head()
```

4.求解模型参数

0）数据标准化

```python
# 数据标准化
def normalize(X):
    sigma = np.std(X, axis=0)
    mu = np.mean(X, axis=0)
    X = (X - mu) / sigma
    return np.array(X)


X = np.array(Xtrain).reshape(np.size(Xtrain, 0), -1)
y = np.array(Ytrain).T.reshape(-1, 1)

# 归一化（闭式求解其实不需要，但梯度下降需要，为了对比统一都采用归一化）
X = normalize(X)
y = normalize(y)

# 计算 R^2
def R2(y, y_pred):
    return 1 - (np.sum((y - y_pred) ** 2) / np.sum((y - np.mean(y)) ** 2))
```

1）线性回归模型的闭合形式参数求解

```python
# 正规方程求解
def Normal_Equation(X, y):
    return np.linalg.inv(X.T @ X) @ X.T @ y

start_time = time.time()
theta_ne = Normal_Equation(X, y)

print(f"花费时间: {time.time() - start_time}")
print(f"R^2: {R2(y, X @ theta_ne)}")
```

```
# 创建 DataFrame
result_cf = pd.DataFrame({"ColumnName": list(Xtrain.columns), "Theta":
theta_ne.flatten()})
result_cf
```

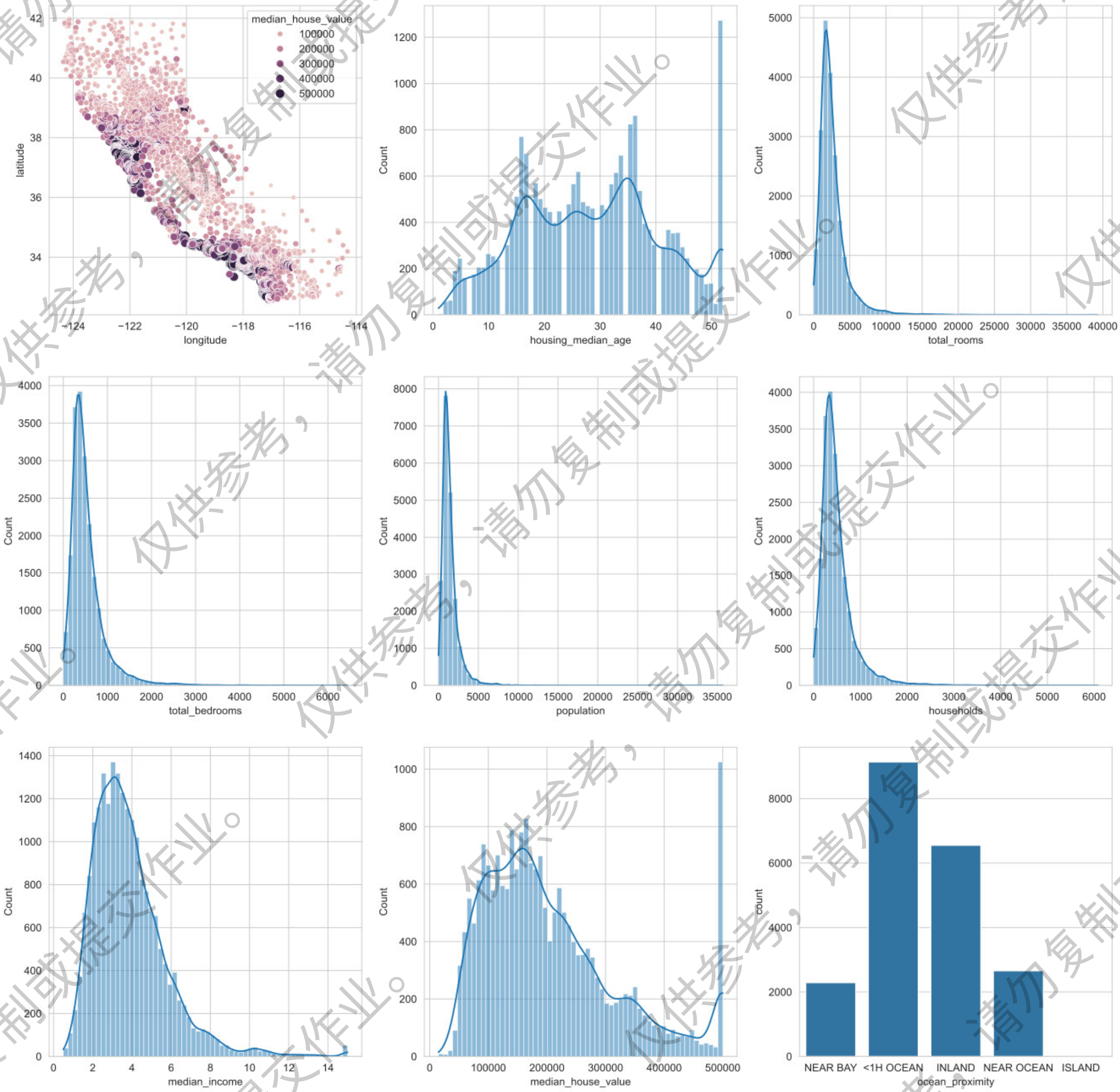
2）线性回归梯度下降参数求解

```
# 损失函数
def MSE_Loss(y, y_pred):
    return np.sum((y_pred - y) ** 2) / (2 * np.size(y))


# 梯度下降
def GD(X, y, lr=0.01, epochs=5000):
    m, n = X.shape

    # 初始化参数为标准正态分布
    theta = np.random.randn(n, 1)
    # 记录每代损失
    loss = np.zeros(epochs)

    for epoch in range(epochs):
        # 计算梯度
        gradient = (1 / m) * (X.T @ (X @ theta - y))
        # 更新参数
        theta -= lr * gradient
        # 记录损失
        loss[epoch] = MSE_Loss(y, X @ theta)

    return theta, loss


start_time = time.time()
[theta_gd, loss] = GD(X, y)

print(f"花费时间: {time.time() - start_time}")
print(f"R^2: {R2(y, X @ theta_gd)}")

# 创建 DataFrame
result_gd = pd.DataFrame({"ColumnName": list(Xtrain.columns), "Theta":
theta_gd.flatten()})
result_gd
```

```python
# 绘制损失函数梯度下降曲线
sns.lineplot(x=np.arange(5000), y=loss.flatten(), label='Loss Curve')

plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Gradient Descent Loss Curve')
```

（4）展示实验结果，比较两种求解方式的优劣

1. 探索数据特征实验结果：



从上图可以清晰看出各个特征的数据分布。

2. 求解结果展示：

正规方程求解

花费时间：0.0021278858184814453
R^2：0.6421055244197833

	ColumnName	Theta
0	longitude	-0.450905
1	latitude	-0.459258
2	housing_median_age	0.115849
3	total_rooms	-0.080094
4	total_bedrooms	0.259762
5	population	-0.365912
6	households	0.230004
7	median_income	0.638939
8	ocean_0	0.105065
9	ocean_1	-0.061217
10	ocean_2	0.023299
11	ocean_3	0.055310
12	ocean_4	0.079888

13 行 × 2 列 pd.DataFrame

梯度下降求解

花费时间：0.856015682220459
R^2：0.6450859546376048

	ColumnName	Theta
0	longitude	-0.505338
1	latitude	-0.509818
2	housing_median_age	0.117400
3	total_rooms	-0.085778
4	total_bedrooms	0.335908
5	population	-0.385874
6	households	0.183076
7	median_income	0.633818
8	ocean_0	0.793215
9	ocean_1	0.591630
10	ocean_2	0.050584
11	ocean_3	0.485062
12	ocean_4	0.552887

13 行 × 2 列 pd.DataFrame

（5）讨论实验结果，分析各个特征与目标预测值的正负相关性

1. 实验结果讨论

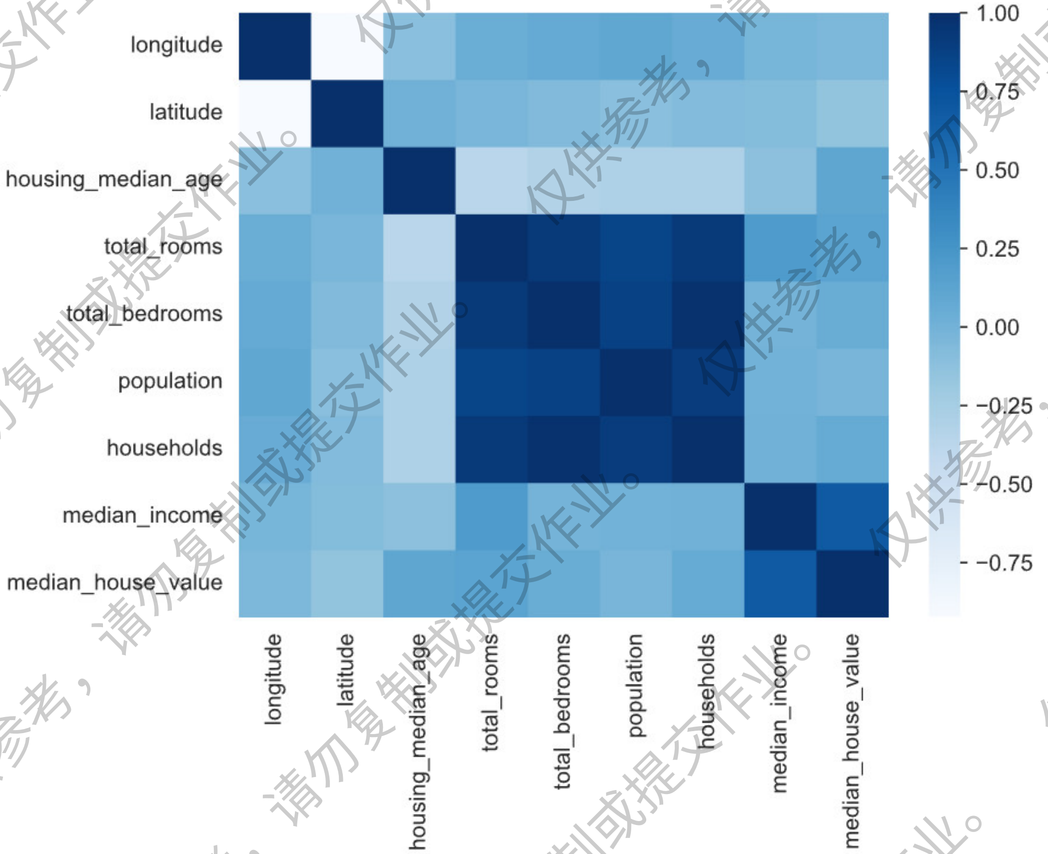
在本数据集中，使用正规方程方法求解耗时 0.0021s，梯度下降法耗时 0.85s，可以看到正规方程法比梯度下降求解更加快，正规方程是直接运算求解显式解，而梯度下降法则需要不断迭代更新，这是正规方程法远远快于梯度下降法的原因。但是，正规方程法求解时需要计算 $(X^T X)^{-1}$，在计算机中求解矩阵的逆的时间复杂度达到 n^3，当特征维度非常大（查询资料对“大”的定义：维度 d 超过 10000）时将非常非常耗费时间。

在维度特征过大时，最好使用梯度下降法进行求解，梯度下降法是一个迭代算法，这意味着可能会更加慢。在本实验中梯度下降法的确远远慢于正规方程法，并且梯度下降法需要预先设定好学习率，当学习率过大时将错过最优值点，过小又会导致收敛过慢。

两种算法各有优劣，需要具体根据数据集来定使用哪个。

2. 分析各个特征与目标预测值的正负相关性

采用相关系数来分析正负相关性，绘制相关系数热力图如下：



	median_house_value
longitude	-0.045967
latitude	-0.144160
housing_median_age	0.105623
total_rooms	0.134153
total_bedrooms	0.049686
population	-0.024650
households	0.065843
median_income	0.688075
median_house_value	1.000000

通过热力图和相关系数的分析，发现 median_income 与目标预测值呈现最强烈的正相关性，即随着 median_income 的增加，房价也随之上升。进一步计算相关系数显示，还有 housing_median_age、total_rooms、total_bedrooms、households 对预测值具有正相关作用；longitude、latitude 和 population 呈现负相关作用，他们越大，预测值反而会越小。