9. Experiment I

线性分类与回归

Linear Classification and Regression

第一节实验课（**第 11 周，3 课时，学院楼 B406**）将进行对线性分类与回归的实践能力考察。本小节将具体从 2 个子实验对线性分类与回归的实践要求进行说明。

实验内容 1：Setosa 鸢尾花和 Versicolour 鸢尾花二分类问题

实验内容 2：Diamonds 回归预测问题

本次实验所涉及编程内容不可使用现成函数进行，即分类器（回归器）、损失函数、优化求解过程均需编码实现。

实验报告提交（**请仔细对照**）：

- 格式：四号字体、单倍行距、中文宋体、英文新罗马字体，封面使用“实验报告封面.docx”，电子版编辑完后转为 pdf 格式发送，文件名 20AI+a+ 学号.pdf（例如：20AIa1033200101.pdf）；
- 时间节点：12 周周末（2023/05/07 18:00 前）
- 电子版发送至 raojiyong@stu.jiangnan.edu.cn，同步抄送 tianyang.xu@jiangnan.edu.cn，邮件标题:20AI+a+ 学号（例如：20AIa1033200101）

9.1 Setosa 鸢尾花和 Versicolour 鸢尾花二分类问题

9.1.1 数据：

结合第一章末尾处的内容，利用 ‘conda activate #name’ 进行实验环境激活，再安装 matplotlib、sklearn 等常用工具包。结合下面的代码熟悉鸢尾花 iris 数据集，实现对该数据

集的样本数、维度、类别等属性的分析与可视化。

```
# Code source: Gaël Varoquaux
# Modified for documentation by Jaques Grobler
# License: BSD 3 clause

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
y = iris.target

x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

plt.figure(2, figsize=(8, 6))
plt.clf()

# Plot the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

# To getter a better understanding of interaction of the dimensions
# plot the first three PCA dimensions
fig = plt.figure(1, figsize=(8, 6))
ax = Axes3D(fig, elev=-150, azim=110)
X_reduced = PCA(n_components=3).fit_transform(iris.data)
ax.scatter(
    X_reduced[:, 0],
    X_reduced[:, 1],
    X_reduced[:, 2],
    c=y,
    cmap=plt.cm.Set1,
    edgecolor="k",
```

```
    s=40,
)
ax.set_title("First three PCA directions")
ax.set_xlabel("1st eigenvector")
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("2nd eigenvector")
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("3rd eigenvector")
ax.w_zaxis.set_ticklabels([])

plt.show()
```

9.1.2 实验课具体要求：

编写线性分类器模型，编写损失函数、编写梯度反传函数；实现 iris 中 Setosa 鸢尾花和 Versicolour 鸢尾花的二分类学习；训练数据为 Setosa 鸢尾花和 Versicolour 鸢尾花各自前 30 条数据（共 60 条数据）；测试数据为 Setosa 鸢尾花和 Versicolour 鸢尾花各自后 20 条数据（共 40 条数据）。

9.1.3 评分依据：

- ★★★ 完成分类器、损失函数、梯度的编写，调整训练参数实现测试集上的高精度。
- ★★☆ 完成分类器、损失函数、梯度的编写，调通训练和测试过程。
- ★☆☆ 利用已有函数的调用，调通训练和测试过程。

9.1.4 实验报告：

- 核心分类器、损失函数、梯度反传的代码及注释
- 训练测试的过程分析（可选：不同学习率、不同损失函数对结果的影响）
- 实验分析（如：哪种特征更加有利于中 Setosa 鸢尾花和 Versicolour 鸢尾花的二分类）

9.1.5 深入学习：

Kaggle 上有大量关于此数据集的代码（https://www.kaggle.com/datasets/uciml/iris），可以通过阅读他人代码掌握看问题的不同角度。

9.2 Diamonds 回归预测问题

9.2.1 数据：

利用 seaborn 获取 diamonds 数据集，通过可视化初步了解数据内容，熟悉如下各种数据分析可视化方式。https://www.kaggle.com/shivam2503/diamonds

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.get_dataset_names()
diamond=sns.load_dataset('diamonds', cache=True, data_home=None)
# 网络原因或导致无法直接下载; 则可自行下载后导入

# Scatter Plot
sns.scatterplot(x="carat", y="price", hue="cut", size="depth", data=diamond)

plt.xlabel('Carat Weight')
plt.ylabel('Price')
plt.title('Scatter Plot: Carat Vs Price (Based on Cut)')

# Rug Plot
sns.rugplot(x="carat", y="price", hue="cut", data=diamond)

# Scatter Plot with Rug Plot: Carat Vs Price (Based on Cut)
sns.scatterplot(x="carat", y="price", hue="cut", size="depth", data=diamond)

sns.rugplot(x="carat", y="price", hue="cut", data=diamond)

# Scatter Plot with Rug Plot: Carat Vs Price (Based on Color)
sns.scatterplot(x="carat", y="price", hue="color", size="depth", data=
    diamond)

sns.rugplot(x="carat", y="price", hue="color", data=diamond)

# Scatter Plot with Rug Plot: Carat Vs Price (Based on Clarity)
sns.scatterplot(x="carat", y="price", hue="clarity", size="depth", data=
    diamond)

sns.rugplot(x="carat", y="price", hue="clarity", data=diamond)

# Strip plot
fig, st_axis = plt.subplots(2, 3, figsize=(16,8))
sns.stripplot(x="cut", y="carat", data=diamond,ax = st_axis[0,0])
sns.stripplot(x="color", y="carat", data=diamond,ax = st_axis[0,1])
sns.stripplot(x="clarity", y="carat", data=diamond,ax = st_axis[0,2])
sns.stripplot(x="cut", y="price", data=diamond,ax = st_axis[1,0])
sns.stripplot(x="color", y="price", data=diamond,ax = st_axis[1,1])
sns.stripplot(x="clarity", y="price", data=diamond,ax = st_axis[1,2])

```

```
# Bar Plot
fig, ba_axis = plt.subplots(1, 3, figsize=(16,12))
sns.barplot(x="cut", y="price", data=diamond, ax = ba_axis[0], color='Red')
sns.barplot(x="color", y="price", data=diamond, ax = ba_axis[1], color='salmon')
sns.barplot(x="clarity", y="price", data=diamond, ax = ba_axis[2], color='purple')

# Pair Plot
sns.pairplot(diamond, plot_kws=dict(marker="+", linewidth=1), diag_kws=dict(fill=False))

# Line Plot
sns.lineplot(x="carat", y="x", data=diamond, color='salmon')
sns.lineplot(x="carat", y="y", data=diamond, color='purple')
sns.lineplot(x="carat", y="z", data=diamond, color='Red')
sns.lineplot(x="carat", y="depth", data=diamond, sort=False, lw=1, color='blue')

sns.lineplot(x="carat", y="x", data=diamond, hue='cut')
sns.lineplot(x="carat", y="x", data=diamond, hue='color')
sns.lineplot(x="carat", y="x", data=diamond, hue='clarity')
sns.lineplot(x="carat", y="y", data=diamond, hue='cut')
sns.lineplot(x="carat", y="y", data=diamond, hue='color')
sns.lineplot(x="carat", y="y", data=diamond, hue='clarity')
sns.lineplot(x="carat", y="z", data=diamond, hue='cut')
sns.lineplot(x="carat", y="z", data=diamond, hue='color')
sns.lineplot(x="carat", y="z", data=diamond, hue='clarity')

# Count Plot
sns.histplot(data=diamond, x='carat', color='black')
sns.histplot(data=diamond, y='price', color='salmon')
sns.histplot(data=diamond, x='depth', hue='cut', color='cyan')
sns.histplot(data=diamond, x='table', hue='color', multiple='stack', color='red')
sns.histplot(data=diamond, x='x', hue='clarity', element='step', color='blue')
sns.histplot(data=diamond, x='y', hue='clarity', element='poly', color='pink')
sns.histplot(data=diamond, x='z', hue='cut', element="step", stat="density", common_norm=False, color='orange')

# Heat Map
```

```
77 sns.heatmap(diamond.corr(), linecolor='white', linewidths=2,annot=True)
78
79 # ECDF Plot
80 sns.ecdfplot(data=diamond, x="carat", stat="count")
81 sns.ecdfplot(data=diamond, x="price", stat="count")
82 sns.ecdfplot(data=diamond, x="cut", stat="count")
83 sns.ecdfplot(data=diamond, x="color", stat="count")
84 sns.ecdfplot(data=diamond, x="clarity", stat="count")
85 sns.ecdfplot(data=diamond, x="depth", stat="count", complementary=True)
86 sns.ecdfplot(data=diamond, x="table", stat="count", complementary=True)
87
88 # Box Plot
89 sns.boxplot(x="carat", y="cut", data=diamond, color="salmon", width=0.5,
       linewidth=2.5)
90 sns.boxplot(x="carat", y="color", data=diamond, color="royalblue", width
       =0.5, linewidth=2.5)
91 sns.boxplot(x="carat", y="clarity" , data=diamond, color="cyan", width=0.5,
       linewidth=2.5)
92
93 # Violin plot
94 sns.violinplot(x="cut", y="carat", data=diamond, color="cyan", width=1)
95 sns.violinplot(x="color", y="carat", data=diamond, palette="muted" , color="
       royalblue" , width=1)
96 sns.violinplot(x="clarity", y="carat", data=diamond, palette="muted", split=
       True, inner="quartile", color="red", width=1)
97
98 # Point plot
99 sns.pointplot(x="cut", y="carat", data=diamond)
100 sns.pointplot(x="cut", y="carat", hue="color", data=diamond, dodge=True)
101 sns.pointplot(x="cut", y="carat", hue="clarity", data=diamond, dodge=True)
102 sns.pointplot(x="color", y="carat", data=diamond)
103 sns.pointplot(x="color", y="carat", hue="cut", data=diamond, join=False)
104 sns.pointplot(x="color", y="carat", hue="clarity", data=diamond, join=False)
105 sns.pointplot(x="carat", y="clarity", data=diamond)
106 sns.pointplot(x="carat", y="clarity", hue="cut", data=diamond)
107 sns.pointplot(x="carat", y="clarity", hue="color", data=diamond)
108
109 # Reg Plot
110 fig, re_axis = plt.subplots(2, 2,figsize=(12,12))
111 sns.regplot(x = 'carat', y = 'price', data=diamond, ax = re_axis[0,0], color
       ='royalblue')
112 sns.regplot(x = 'x', y = 'price', data=diamond, ax = re_axis[0,1], color='
       salmon')
```

```
sns.regplot(x = 'y', y = 'price', data=diamond, ax = re_axis[1,0], color='green')
sns.regplot(x = 'z', y = 'price', data=diamond, ax = re_axis[1,1], color='cyan')

# Kernel Density Plot
sns.kdeplot(data=diamond, x="carat", hue="cut")
sns.kdeplot(data=diamond, x="carat", hue="color", multiple="stack")
sns.kdeplot(data=diamond, x="carat", hue="clarity", multiple="fill")

# Jointplot
sns.jointplot(data=diamond, x='carat', y='depth', marker='^', hue='cut')

sns.jointplot(data=diamond, x='carat' ,y='depth', marker='+', hue='color')

sns.jointplot(data=diamond, x='carat', y='depth', marker='>', hue='clarity')

# JointGrid
jg_cut = sns.JointGrid(data=diamond, x="carat", y="price", hue="cut")
jg_cut.plot(sns.scatterplot, sns.histplot, alpha=.7, edgecolor=".2", linewidth=.5)

jg_color = sns.JointGrid(data=diamond, x="carat", y="price", hue="color")
jg_color.plot(sns.scatterplot, sns.boxplot)

jg_clarity = sns.JointGrid(data=diamond, x="carat", y="price", hue="clarity")
jg_clarity.plot(sns.scatterplot, sns.kdeplot)
```

9.2.2 实验课具体要求：

编写线性回归器模型，编写损失函数、编写梯度反传函数；实现 diamonds 特征（carat，cut，color，clarity，depth，table，x，y，z）对价格（price）的预测；训练数据为第 1-40000 条数据中所有合数索引对应的数据；测试数据为第 1-40000 条数据中所有质数索引对应的数据（4203 个）。

[TIPS：求解 n 以内所有质数的函数 ⇓]

```
import math
def func_get_prime(n):
  return filter(lambda x: not [x%i for i in range(2, int(math.sqrt(x))+1) if x%i ==0], range(2,n+1))
```

9.2.3 评分依据：

- ★★★ 完成回归器、损失函数、梯度的编写，调整训练参数实现测试集上的高精度。
- ★★☆ 完成回归器、损失函数、梯度的编写，调通训练和测试过程。
- ★☆☆ 利用已有函数的调用，调通训练和测试过程。

9.2.4 实验报告：

- 核心回归器、损失函数、梯度反传的代码及注释
- 训练测试的过程分析（可选：不同学习率、不同损失函数对结果的影响）
- 实验分析（如：哪种特征对钻石价格的影响是正相关/负相关的）