机器学习导论
习题一

171830635, 俞星凯, yuxk@smail.nju.edu.cn

2020 年 3 月 11 日

学术诚信

本课程非常重视学术诚信规范，助教老师和助教同学将不遗余力地维护作业中的学术诚信规范的建立。希望所有选课学生能够对此予以重视。[1]

(1) 允许同学之间的相互讨论，但是署你名字的工作必须由你完成，不允许直接照搬任何已有的材料，必须独立完成作业的书写过程;

(2) 在完成作业过程中，对他人工作（出版物、互联网资料）中文本的直接照搬（包括原文的直接复制粘贴及语句的简单修改等）都将视为剽窃，剽窃者成绩将被取消。对于完成作业中有关键作用的公开资料，应予以明显引用；

(3) 如果发现作业之间高度相似将被判定为互相抄袭行为，抄袭和被抄袭双方的成绩都将被取消。因此请主动防止自己的作业被他人抄袭。

作业提交注意事项

(1) 请在 LaTeX 模板中第一页填写个人的姓名、学号、邮箱信息；

(2) 本次作业需提交该 pdf 文件、问题 2 问题 4 可直接运行的源码 (两个.py 文件)、作业 2 用到的数据文件 (为了保证问题 2 代码可以运行)，将以上四个文件压缩成 zip 文件后上传，例如 181221001.zip;

(3) 未按照要求提交作业，或提交作业格式不正确，将会被扣除部分作业分数；

(4) 本次作业提交截止时间为 3 月 15 日 23:59:59。除非有特殊情况（如因病缓交），否则截止时间后不接收作业，本次作业记零分。

[1] 参考尹一通老师高级算法课程中对学术诚信的说明。

Problem 1

若数据包含噪声，则假设空间中有可能不存在与所有训练样本都一致的假设，此时的版本空间是什么？在此情形下，试设计一种归纳偏好用于假设选择。

Solution. 此处用于写解答 (中英文均可)

从数据中选取最大子集，使得存在与其一致的假设，并且遵循归纳偏好选择假设：

- 如果存在多个基数相同的最大子集，那么选择与最早发现的最大子集一致的假设。
- 如果存在多个与同一最大子集一致的假设，那么根据“奥卡姆剃刀”选择更“简单”的假设。
- 如果存在多个同样“简单”的假设，那么任意选择一个假设。

Problem 2 [编程]

现有 500 个测试样例,其对应的真实标记和学习器的输出值如表1所示 (完整数据见 data.csv 文件)。该任务是一个二分类任务，1 表示正例，0 表示负例。学习器的输出越接近 1 表明学习器认为该样例越可能是正例，越接近 0 表明学习器认为该样例越可能是负例。

表 1: 测试样例表

样本	x_1	x_2	x_3	x_4	x_5	...	x_{496}	x_{497}	x_{498}	x_{499}	x_{500}
标记	1	1	0	0	0	...	0	1	0	1	1
输出值	0.206	0.662	0.219	0.126	0.450	...	0.184	0.505	0.445	0.994	0.602

(1) 请编程绘制 P-R 曲线

(2) 请编程绘制 ROC 曲线，并计算 AUC

本题需结合关键代码说明思路，并贴上最终绘制的曲线。建议使用 Python 语言编程实现。(预计代码行数小于 100 行)

提示:

- 需要注意数据中存在输出值相同的样例。
- 在 Python 中，数值计算通常使用 Numpy, 表格数据操作通常使用 Pandas, 画图可以使用 Matplotlib (Seaborn), 同学们可以通过上网查找相关资料学习使用这些工具。未来同学们会接触到更多的 Python 扩展库，如集成了众多机器学习方法的 Sklearn, 深度学习工具包 Tensorflow, Pytorch 等。

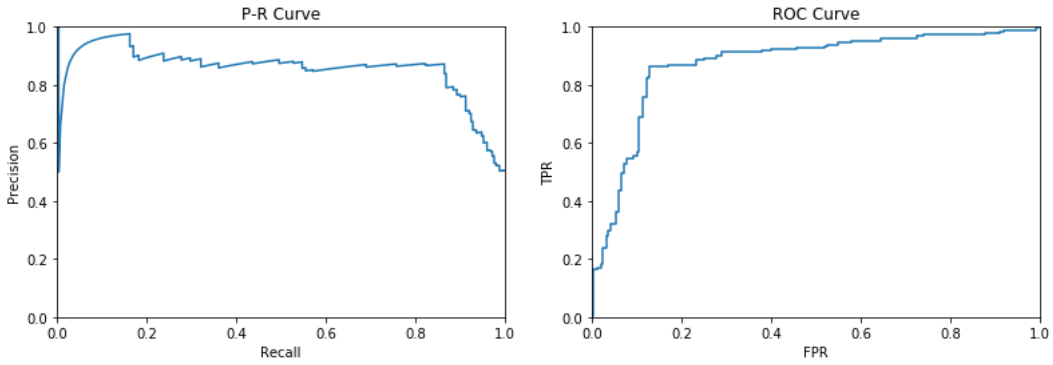
Solution. 此处用于写解答 (中英文均可)

```
data = data.sort_values('output', ascending=False)
thresholds = data['output'].unique()
n = len(thresholds)
for i in range(n):
    data['pred'] = data['output'] >= thresholds[i]
    TP = np.sum((data['label'] == 1) & (data['pred'] == 1))
    FP = np.sum((data['label'] == 0) & (data['pred'] == 1))
    TN = np.sum((data['label'] == 0) & (data['pred'] == 0))
    FN = np.sum((data['label'] == 1) & (data['pred'] == 0))
    P[i] = TP / (TP + FP)
    R[i] = TP / (TP + FN)
    TPR[i] = TP / (TP + FN)
    FPR[i] = FP / (FP + TN)
```

先将数据按照output从大到小排序，再依次取output集合中的元素作为阈值，分别计算真正例、假正例、真反例、假反例，从而计算查准率、查全率、真正例率、假正例率。

```
AUC = np.sum((FPR[1:] - FPR[:-1]) * (TPR[:-1] + TPR[1:])) / 2
```

AUC 根据上面代码计算得到 0.8737。



(a) P-R 曲线 (b) ROC 曲线

图 1: 代码绘图

Problem 3

对于有限样例，请证明

$$\mathrm{AUC} = \frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(\mathbb{I}(f(x^+) > f(x^-)) + \frac{1}{2}\mathbb{I}(f(x^+) = f(x^-))\right)$$

Proof. 此处用于写证明 (中英文均可)

将 m^+ 个正例和 m^- 个反例分别按照预测值从大到小进行排序，分别记为 $x_1^+, x_2^+, \cdots, x_{m^+}^+$

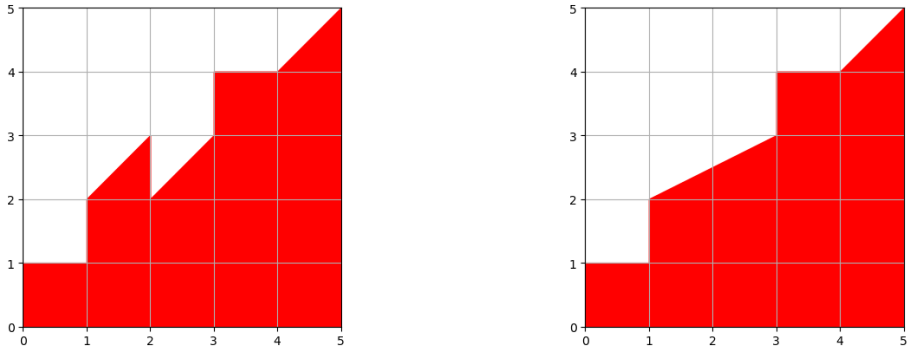
和 $x_1^-, x_2^-, \cdots, x_{m^-}^-$。考虑一个网格平面，纵坐标范围 $[0, m^+]$，横坐标范围 $[0, m^-]$。对于任意 1×1 网格，根据其右上角顶点 (i, j)，按照如下规则染色：

- $f(x_j^+) > f(x_i^-)$，将整个网格染色。
- $f(x_j^+) < f(x_i^-)$，不染色。
- $f(x_j^+) = f(x_i^-)$，将网格左下、右上、右下三点组成的三角形染色。

因为小正方形面积为 1，小三角形面积为 $\frac{1}{2}$，所以染色区域面积为：

$$\sum_{x^+ \in D^+} \sum_{x^- \in D^-} \left(\mathbb{I}(f(x^+) > f(x^-)) + \frac{1}{2} \mathbb{I}(f(x^+) = f(x^-)) \right)$$

修正染色区域，若 a 个正例和 b 个反例预测值相同，即存在宽为 a，长为 b 的矩形，其中所有网格都是三角形染色，则对该矩形重新染色，将矩形左下、右上、右下三点组成的三角形染色。注意修正前后，该矩形染色面积都为 $\frac{1}{2}ab$，因此修正之后的染色区域面积不变。



(a) 染色图示 (b) 修正图示

图 2: 染色与修正，样例满足 $x_1^+ > x_1^- > x_2^+ > x_2^- = x_3^+ = x_3^- > x_4^+ > x_4^- > x_5^+ = x_5^-$

将坐标系的横纵坐标范围都缩小到 $[0, 1]$，这时染色区域面积缩小 m^+m^- 倍。考虑染色区域左边界上的点 (x, y)，对应原来网格平面上的点 (i, j)。若将满足 $f(x) \geq max\{f(x_i^-), f(x_j^+)\}$ 的样例预测为正例，其余预测为反例，则 x 是假正例率，y 是真正例率，于是所有这些点构成 ROC 曲线。又根据小于关系传递性，染色区域内部不存在空白，故 AUC 即为染色区域面积：

$$\text{AUC} = \frac{1}{m^+m^-} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \left(\mathbb{I}(f(x^+) > f(x^-)) + \frac{1}{2} \mathbb{I}(f(x^+) = f(x^-)) \right)$$

□

Problem 4 [编程]

在数据集 D_1, D_2, D_3, D_4, D_5 运行了 A, B, C, D, E 五种算法，算法比较序值表如表2所示:

使用 Friedman 检验 ($\alpha = 0.05$) 判断这些算法是否性能都相同。若不相同，进行 Nemenyi 后续检验 ($\alpha = 0.05$)，并说明性能最好的算法与哪些算法有显著差别。本题需编程实现 Friedman 检验和 Nemenyi 后续检验。(预计代码行数小于 50 行)

表 2: 算法比较序值表

数据集	算法 A	算法 B	算法 C	算法 D	算法 E
D_1	2	3	1	5	4
D_2	5	4	2	3	1
D_3	4	5	1	2	3
D_4	2	3	1	5	4
D_5	3	4	1	5	2
平均序值	3.2	3.8	1.2	4	2.8

Solution. 此处用于写解答 (中英文均可)

```
# Friedman检验
r = data.mean()
n, k = data.shape
chi2 = 12 * n / k / (k + 1) * (np.sum(r ** 2) - (k * (k + 1) ** 2) / 4)
fvalue = (n - 1) * chi2 / (n * (k - 1) - chi2)
if fvalue < f.ppf(0.95, 4, 16):
    print("所有算法性能相同")
# Nemenyi检验
else:
    cd = 2.728 * np.sqrt(k * (k + 1) / 6 / n)
    for i in range(k):
        for j in range(i + 1, k):
            if abs(r[i] - r[j]) > cd:
                print(f'算法{r.index[i]}与{r.index[j]}性能显著不同')
```

代码根据西瓜书公式实现，性能最好的是算法 C，与算法 D 有显著区别。