机器学习导论
习题一

171840708, 张逸凯, zykhelloha@gmail.com

2020 年 3 月 10 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. 由版本空间定义: 与训练集一致的假设集合. 我们可以知道此时版本空间为**空集.**

此时归纳偏好从假设空间中选取假设, 在实际问题中算法的归纳偏好是否与问题本身匹配, 大多数时候直接决定了算法能否取得好的性能. 但是本题没有给实际问题, 那么可以**从以下角度**考虑设计一种归纳偏好用于假设选择:

- 如果可以度量某个假设对训练样本的拟合程度: 选择对训练样本拟合程度最高的模型, 比如选择训练中选择满足最多样本的假设.
- 反之如果不可以比较不同假设之间对训练样本的拟合程度(偏序关系): 则可以考虑奥卡姆剃刀原则: 选择简单的, 比如曲线更平滑就是更简单.

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. PR曲线, ROC曲线绘制原理:

按学习器对样本的预测值进行排序, **逐个把样本作为正例**(如果是ROC, 则是**逐个样本的预测值作为threshold**)进行计算P, R, TPR, FPR(分别按照书上公式2.8, 2.9, 2.18, 2.19). 在这个计算过程中存储每一步的值并绘制图像.

计算AUC: 就是计算一个个小梯形面积, 书本式2.20.

```python
# 最先调用时befIsTrue == 1, 即初始时第一个为True, 其余为False.
# data is sorted
def countPRConfusionMatrix(data, befIsTrue):
    TP, FP, FN, TN = 0, 0, 0, 0
    # 认为是正例的:
    for i in range(0, befIsTrue):
        if data[i][1] > 0.5:
            TP += 1
        else:
            FP += 1
    # 认为是负例的:
    for i in range(befIsTrue, data.shape[0]):
        if data[i][1] > 0.5:
            FN += 1
        else:
            TN += 1
    return TP, FP, FN, TN
```

图 1: Problem 2:计算PR曲线的for-loop代码(按照排序后一个个样例的顺序)

```python
    testy = data[:, 1]
    mPlus, mSub = len(testy[testy == 1]), len(testy[testy == 0])
    TPR, FPR = [0], [0]
    bef = 0
    AUC = 0
    for threshold in data[:, 2]:
        TP, FP = 0, 0
        for j in range(bef, data.shape[0]):
            if data[j][2] < threshold:
                bef = j
                break
            if data[j][1] < 0.5:
                FP += 1
            else:
                TP += 1
        if TP * FP != 0:
            print("A slash!")
        TPR.append(TPR[-1] + TP / mPlus)
        FPR.append(FPR[-1] + FP / mSub)
        AUC += (TPR[-2] + TPR[-1]) * (FPR[-1] - FPR[-2]) / 2
    TPR, FPR = np.array(TPR), np.array(FPR)
    print("AUC = {}".format(AUC))
```

图 2: Problem 2:计算ROC曲线的for-loop代码(按照排序后一个个阈值的顺序)

注意到提示信息: 需要注意数据中存在输出值相同的样例, 输出(预测)值相同但groundtruth可能不同, 这样就**会影响**每一步计算P, R, TPR, FPR. 所以可能会产生多种不同PR曲线(或ROC出现斜线, 但是这里只要ROC算法写对了就没关系), 在此必须做一个检查:

```
87      pred = data[:, 2]
88      import collections
89      samePred = [item for item, count in collections.Counter(pred).items() if count > 1]
90      errPred = []
91      samePredDict = {}
92      for i in data:
93          if i[2] in samePred:
94              if i[2] not in samePredDict:
95                  # 第一次:
96                  samePredDict[i[2]] = [i]
97              else:
98                  samePredDict[i[2]].append(i)
99
100     for i in samePredDict:
101         tag = samePredDict[i][1][1]
102         for j in samePredDict[i]:
103             if j[1] != tag:
104                 errPred.append(j[2])
105
106     assert len(errPred) == 0
```

图 3: Problem 2:检查是否有输出值相同但真实标签不同的数据的代码

最终绘制曲线:

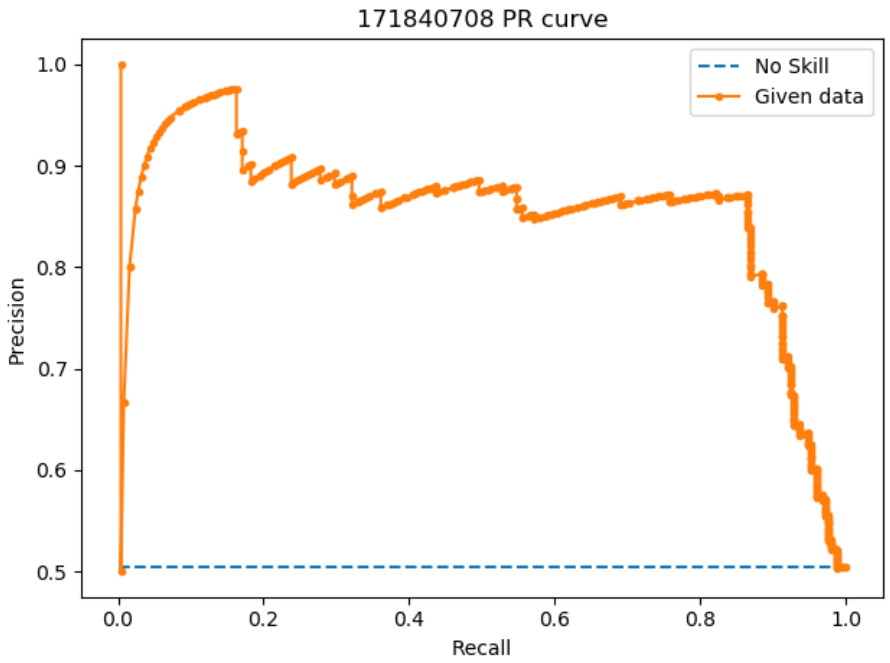

图 4: PR 曲线

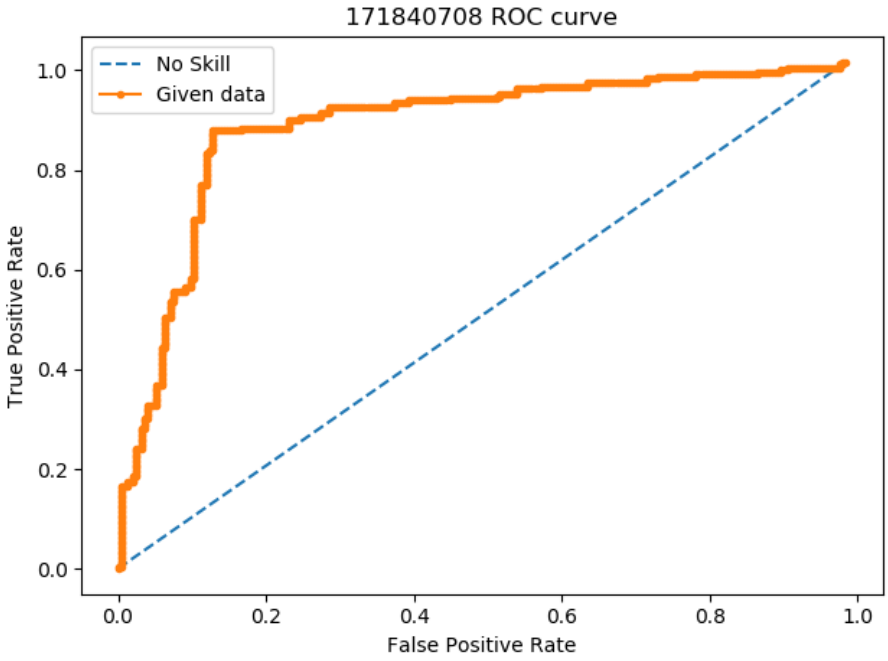

图 5: ROC 曲线

计算AUC: AUC **= 0.8737**

Problem 3

对于有限样例，请证明

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

Proof. 由课本式2.21以及2.22得到:

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

其中D^+为所有正例组成的集合, x^+是其中的正例, D^-为所有反例组成的集合. **注意到:**

$$\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(\mathbb{I}(f(x^+) < f(x^-)) + \mathbb{I}(f(x^+) = f(x^-)) + \mathbb{I}(f(x^+) > f(x^-))\right) = \sum_{x^+\in D^+}\sum_{x^-\in D^-} 1 = m^+m^- \quad (\#\#)$$

$(\#\#)$ **式代入** $(\#)$ **式(**$\div(m^+m^-)$ 并代替 1), 自然地有:

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

□

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:

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

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)

Solution. **最后结果:**

使用Friedman检验($\alpha = 0.05$)判断得这些算法性能**并不相同**, 进行Nemenyi后续检验($\alpha = 0.05$), 性能最好的算法D与算法A, B, E有显著差别, 与C没有显著差别.

计算过程:

使用课本式2.34, 2.35进行Friedman检验, 使用2.36进行Nemenyi后续检验. 代码如下:

```python
import math
ri, k, N = [3.2, 3.8, 1.2, 4, 2.8], 5, 5
alpha, F_test_threshold = 0.05, 3.007
q_alpha = 2.728 # for Nemenyi Test

tau_chi_square = 0
for i in ri:
    tau_chi_square += i * i
tau_chi_square = 12 * N / k / (k + 1) * (tau_chi_square - k * (k + 1) * (k + 1) / 4)

tau_F = (N - 1) * tau_chi_square / (N * (k - 1) - tau_chi_square)
print("\\tau_F = {}".format(tau_F))

assert k == 5 and N == 5 and alpha == 0.05
```

```
16    if tau_F <= F_test_threshold:
17        print("Above algorithms have the same performance (fail to reject H0)")
18    else:
19        print("Above algorithms have different performance (reject H0)")
20        print("Continue to Nemenyi Test..")
21        CD = q_alpha * math.sqrt(k * (k + 1) / 6 / N)
22
23        diffTrue, diffFalse = [], []
24        maxRiIndex = ri.index(max(ri))
25        print("The best performing algorithms is {}, whose r_i = {}".format(maxRiIndex + 1, ri[maxRiIndex]))
26        for i in range(len(ri)):
27            if i != maxRiIndex:
28                if abs(ri[i] - ri[maxRiIndex]) <= CD:
29                    diffTrue.append(i + 1)
30                else:
31                    diffFalse.append(i + 1)
32
33        print("The best performing algorithms differ significantly from the following: ")
34        print(diffTrue)
35        print("The best performing algorithms are not significantly different from the following: ")
36        print(diffFalse)
37        print("The index starts at one")
```

Problem 3额外补充: 课本式2.22的证明:

本证明参考了https://datawhalechina.github.io/pumpkin-book/#/chapter2/chapter2 南瓜书的思想.

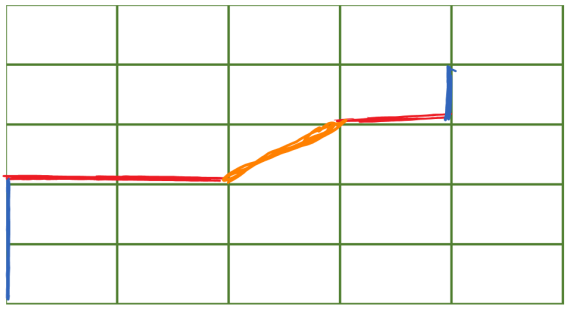

图 6: ROC曲线放大

Proof.

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

注意到在画ROC curve的时候x轴$step_x = \frac{1}{m^-}$, y轴$step_y = \frac{1}{m^+}$.

所以xy平面可以被划分成面积是$step_x \times step_y$的小方块构成的, 由ROC curve绘制过程可以发现从一维的角度, 曲线上每一个平行于y轴的线段代表一个正例(即$x^+ \in D^+$), 同理平行于x轴的线段代表$x^- \in D^-$.

注意如果有多个真实分别是正例(s个)和假例(t个)的预测值一样, 这样降低分类阈值的时候会多个出现, 取下一个点就是:

$$\left(x + \frac{t}{m^-}, y + \frac{s}{m^+}\right)$$

所以其实ROC curve还会出现斜线, 不过问题不大, 因为我们可以发现这对书上AUC计算没有影响, 就变成计算梯形面积了.

从二维平面的角度$\sum_{x^+ \in D^+} \sum_{x^- \in D^-}$ 代表对xy平面的$m^+ \times m^-$进行遍历.

$f(x^+) < f(x^-)$ 代表通过遍历所有反样例来统计预测值大于x_i^+的预测值的反样例个数，也即该线段左边和下边的**平行于x轴线段**个数, 加上**倾斜线段**对应的反样例个数(即在x轴上投影有多少个$step_x$).

综上所述, 可以发现

$$l_{\text{rank}} + AUC = 1$$

□