Introduction to Machine Learning
Homework 5

181860066 牛铭杨

2020 年 12 月 29 日

1 [30pts] Naive Bayes Classifier

We learned about the naive Bayes classifier using the "property conditional independence hypothesis". Now we have a data set as shown in the following table:

表 1: Dataset

	x_1	x_2	x_3	x_4	y
Instance1	1	1	1	0	1
Instance2	1	1	0	0	0
Instance3	0	0	1	1	0
Instance4	1	0	1	1	1
Instance5	0	0	1	1	1

(1) [15pts] Calculate: $\Pr\{y=1|\mathbf{x}=(1,1,0,1)\}$ and $\Pr\{y=0|\mathbf{x}=(1,1,0,1)\}$.
基于属性条件独立性假设，想要求得的表达式为

$$P(y\,|\boldsymbol{x}\,)=\frac{P(y)}{P(\boldsymbol{x})}P(\boldsymbol{x}\,|y\,) \tag{1.1}$$

其中

$$P(\boldsymbol{x}\,|y\,)=\prod_{i=1}^{4}P(x_i\,|y\,) \tag{1.2}$$

根据表格数据可以计算得到

$$P(y=1)=\frac{3}{5}, \tag{1.3}$$

$$P(y=0)=\frac{2}{5}, \tag{1.4}$$

$$P(x_1=1\,|y=1\,)=\frac{2}{3}, \tag{1.5}$$

$$P(x_2=1\,|y=1\,)=\frac{1}{3}, \tag{1.6}$$

$$P(x_3=0\,|y=1\,)=\frac{0}{3}=0, \tag{1.7}$$

$$P(x_4=1\,|y=1\,)=\frac{2}{3}, \tag{1.8}$$

$$P(\boldsymbol{x}=(1,1,0,1)\,|y=1\,)=\prod_{i=1}^{4}P(x_i\,|y=1\,)=0, \tag{1.9}$$

$$P(x_1=1\,|y=0\,)=\frac{1}{2}, \tag{1.10}$$

$$P(x_2=1\,|y=0\,)=\frac{1}{2}, \tag{1.11}$$

$$P(x_3=0\,|y=0\,)=\frac{1}{2}, \tag{1.12}$$

$$P(x_4=1\,|y=0\,)=\frac{1}{2}, \tag{1.13}$$

$$P(\boldsymbol{x}=(1,1,0,1)\,|y=0\,)=\prod_{i=1}^{4}P(x_i\,|y=0\,)=\frac{1}{16}. \tag{1.14}$$

下面可以用全概率公式计算 $P(\boldsymbol{x})$,

$$\begin{aligned}P(\boldsymbol{x}=(1,1,0,1))&=\sum_{y=0}^{1}P(y)P(\boldsymbol{x}=(1,1,0,1)\,|y\,)\\&=\frac{2}{5}\times\frac{1}{16}\\&=\frac{1}{40}.\end{aligned} \tag{1.15}$$

所以有

$$\Pr\{y=1|\mathbf{x}=(1,1,0,1)\} = \frac{P(y=1)}{P(\boldsymbol{x}=(1,1,0,1))} P(\boldsymbol{x}=(1,1,0,1)\,|y=1\,) \tag{1.16}$$

$$= \frac{\frac{3}{5}}{\frac{1}{40}} \times 0 \tag{1.17}$$

$$= 0, \tag{1.18}$$

$$\Pr\{y=0|\mathbf{x}=(1,1,0,1)\} = \frac{P(y=0)}{P(\boldsymbol{x}=(1,1,0,1))} P(\boldsymbol{x}=(1,1,0,1)\,|y=0\,) \tag{1.19}$$

$$= \frac{\frac{2}{5}}{\frac{1}{40}} \times \frac{1}{16} \tag{1.20}$$

$$= 1. \tag{1.21}$$

(2) [15pts] After using Laplacian Correction, recalculate the value in the previous question.
基于拉普拉斯修正，计算值改为

$$P(y=1) = \frac{3+1}{5+2} = \frac{4}{7}, \tag{1.22}$$

$$P(y=0) = \frac{2+1}{5+2} = \frac{3}{7}, \tag{1.23}$$

$$P(x_1=1\,|y=1\,) = \frac{2+1}{3+2} = \frac{3}{5}, \tag{1.24}$$

$$P(x_2=1\,|y=1\,) = \frac{1+1}{3+2} = \frac{2}{5}, \tag{1.25}$$

$$P(x_3=0\,|y=1\,) = \frac{0+1}{3+2} = \frac{1}{5}, \tag{1.26}$$

$$P(x_4=1\,|y=1\,) = \frac{2+1}{3+2} = \frac{3}{5}, \tag{1.27}$$

$$P(\boldsymbol{x}=(1,1,0,1)\,|y=1\,) = \prod_{i=1}^{4} P(x_i\,|y=1\,) = \frac{18}{625}, \tag{1.28}$$

$$P(x_1 = 1\,|y = 0\,) = \frac{1+1}{2+2} = \frac{1}{2}, \tag{1.29}$$

$$P(x_2 = 1\,|y = 0\,) = \frac{1+1}{2+2} = \frac{1}{2}, \tag{1.30}$$

$$P(x_3 = 0\,|y = 0\,) = \frac{1+1}{2+2} = \frac{1}{2}, \tag{1.31}$$

$$P(x_4 = 1\,|y = 0\,) = \frac{1+1}{2+2} = \frac{1}{2}, \tag{1.32}$$

$$P(\boldsymbol{x} = (1,1,0,1)\,|y = 0\,) = \prod_{i=1}^{4} P(x_i\,|y = 0\,) = \frac{1}{16}. \tag{1.33}$$

所以得到的结果修正为

$$\begin{aligned} P(\boldsymbol{x} = (1,1,0,1)) &= \sum_{y=0}^{1} P(y)P(\boldsymbol{x} = (1,1,0,1)\,|y\,) \\ &= \frac{4}{7} \times \frac{18}{625} + \frac{3}{7} \times \frac{1}{16}. \end{aligned} \tag{1.34}$$

$$\Pr\{y = 1|\mathbf{x} = (1,1,0,1)\} = \frac{P(y = 1)}{P(\boldsymbol{x} = (1,1,0,1))} P(\boldsymbol{x} = (1,1,0,1)\,|y = 1\,) \tag{1.35}$$

$$= \frac{\frac{4}{7}}{\frac{4}{7} \times \frac{18}{625} + \frac{3}{7} \times \frac{1}{16}} \times \frac{18}{625} \tag{1.36}$$

$$= \frac{384}{1009}, \tag{1.37}$$

$$= 0.381, \tag{1.38}$$

$$\Pr\{y = 0|\mathbf{x} = (1,1,0,1)\} = \frac{P(y = 0)}{P(\boldsymbol{x} = (1,1,0,1))} P(\boldsymbol{x} = (1,1,0,1)\,|y = 0\,) \tag{1.39}$$

$$= \frac{\frac{3}{7}}{\frac{4}{7} \times \frac{18}{625} + \frac{3}{7} \times \frac{1}{16}} \times \frac{1}{16} \tag{1.40}$$

$$= \frac{625}{1009}, \tag{1.41}$$

$$= 0.619. \tag{1.42}$$

2 实验报告

2.1 实验目的

本次实验的主要目的在于实现 AdaBoost 算法和 RandomForest 算法，并通过 5 折交叉验证的方法测试集成之后的性能，探究基分类器数目和算法性能的关系。通过编程实现的方法，加深对于这两种机器学习经典算法的理解。

2.2 实验说明

实验环境如下
操作系统：Windows10
编程语言：python 3
运行方法（数据文件 adult.data、adult.test 与代码在同一目录下）：
python AdaBoost.py
python RandomForestMain.py
程序会输出运行后的 AUC 和准确率，若要使用 5-Fold 交叉验证，则将注释代码块取消注释即可

2.3 实验过程

2.3.1 AdaBoost

首先借助 sklearn 包实现 AdaBoost 算法的主体，即一个类 class AdaBoost。我设置了三个成员变量 T、base 和 alpha，分别代表训练轮数、基学习器列表和学习器权重列表。并且有三个成员函数 train、predict 和 predict_with_prob

在 train 中，我们对给定数据训练 AdaBoost。
1. 先初始化样本分布 D 即每个样本的权重为 $\frac{1}{m}$
2. 之后循环 T 次，在每次循环中，
(1) 先基于当前分布 D 训练出单层决策树，并存储在 self.base 中
(2) 计算训练集上预测结果与真实结果的差异 diff
(3) 使用 diff 和分布 D 点乘计算出样本误差 error，若大于 0.5 则停止训练，若为 0 则直接返回
(4) 使用公式 $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$ 计算出当前学习器的权重

(5) 然后更新分布 D，$D = D \times e^{-diff \times \alpha_t}$，并进行归一化，除以 D 所有值的总和

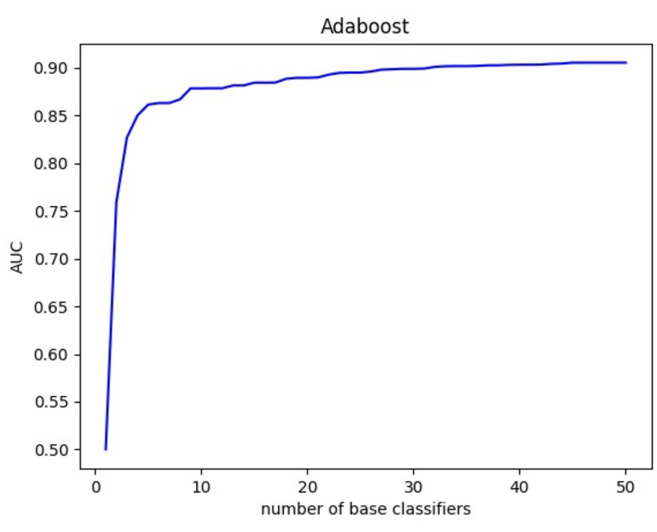
在 predict 中，我们给出测试数据的预测结果（0 或 1）。

1. 记录 T 个学习器的每一个对测试样本总体的预测 $h_t(\boldsymbol{x})$
2. 根据权重求平均预测结果,并根据符号判断类别 $H(\boldsymbol{x}) = \mathrm{sign}(\sum_{i=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

在 predict_with_prob 中，流程与 predict 基本类似，不同点在于这个函数输出的是概率而不是类别。这主要是为了计算 AUC。为此，我们需要做的是将 α_t 归一化，并直接输出一个 [0, 1] 的结果

实现了 AdaBoost 类之后，由于读入数据不能直接使用 sklearn 包处理，我们需要对数据进行预处理。我这里使用了 pandas 包进行处理。我使用 read_csv 读入所有数据，并使用 get_dummies 函数对非数值变量进行独热编码，方便决策树操作。这里我将训练数据和测试数据组合起来进行编码，处理之后再分开，防止编码出的结果维度不同。可以看到进行独热编码后数据维度大大增加。

在实现完基本功能之后需要对 AdaBoost 进行 5-Fold 交叉验证。这里我主要使用了 sklearn.model_selection 的 KFold 函数来分割数据集。之后的实现大致调用 AdaBoost 类中的函数即可。我使用 matplotlib.pyplot 将实验结果画在折线图上，其中对基学习器的数量从 1 到 50，分别记录了集成后的 AUC。



可以看出，当基学习器数量为 25 左右时，算法基本已经收敛不再有大的波动或增长。所以我取基学习器的数量为 25，得到 $AUC = 0.897, accuracy =$

0.851

2.3.2 RandomForest

随机森林算法的伪代码如下

数据预处理和 5-Fold 交叉验证的过程和 AdaBoost 完全相同，我这里就不再赘述。主要区别在于类 RandomForest 的实现。我设置了两个成员变量 T、base，分别代表训练轮数、基学习器列表。除了三个成员函数 train、predict 和 predict_with_prob 之外，还有一个 bootstrap_sample 函数用于自助采样每轮的训练数据。

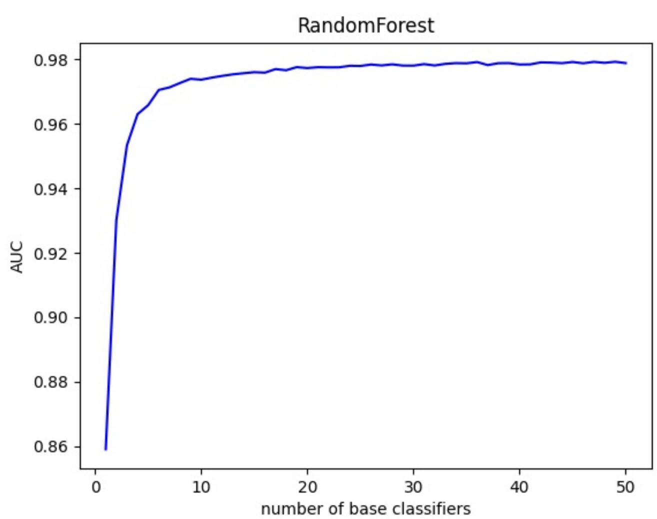
在 bootstrap_sample 中，我自助采样每轮的训练数据，使用 np.random.choice 函数来从训练集中有放回地抽样 m 个数据，来训练决策树。这里我一次性将所有 T 组训练数据全部采样好，这个函数调用一次调用即可。

在 train 中，我们先调用 bootstrap_sample 获取 T 个学习器的训练数据，然后用这些训练数据训练 T 个决策树，都存入 self.base 中。

在 predict 中，使用简单平均法，将 T 个基学习器的结果做平均，与 0.5 比较来得出是正类还是反类。

在 predict_with_prob 中，流程与 predict 基本类似，区别在于不需要和 0.5 比较，直接返回概率即可。

进行和 AdaBoost 一样的 5-Fold 交叉验证，得到的结果如图。



可以看出，当基学习器数量为 15 左右时，算法基本已经收敛不再有大的

Algorithm 1 Random Forest 算法

Require: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$; 属性集 $A = \{a_1, a_2, \cdots, a_d\}$; 训练轮数 T

Ensure: 集成学习器 $H(\boldsymbol{x})$

function RandomDecisionTree(D, A)
    node $\leftarrow$ create a tree node
    if all samples in D belongs to the same class C **then**
        classify *node* as class C
    end if
    if $A = \emptyset$ **or** all samples in D are equal on A **then**
        classify *node* as major class in D
        return a tree with root *node*
    end if
    $K \leftarrow log_2 d$ features randomly chosen from A
    $a_* \leftarrow$ best feature chosen from K
    for each value a_*^v of a_* **do**
        create a branch $node_l$ for *node*
        $D_v \leftarrow \{(\boldsymbol{x}, y) \in D | x_{a_*} = a_*^v\}$
        if $D_v = \emptyset$ **then**
            classify $node_l$ as major class in D
            return a tree with root *node*
        else
            $node_l \leftarrow$ RandomDecisionTree($D_v, K - \{a_*\}$)
        end if
    end for
    return a tree with root *node*
end function

function Train(D, A, T)
    for $i = 1, 2, \cdots, T$ **do**
        $S \leftarrow$ samples randomly chosen from D for m times
        $h_i \leftarrow$ RandomDecisionTree(S, A)
    end for
    return $H(\boldsymbol{x}) \leftarrow \frac{1}{T} \sum_{i=1}^{T} h_i(\boldsymbol{x})$
end function

波动或增长。所以我取基学习器的数量为 15，得到 $AUC = 0.880, accuracy = 0.841$

2.4 实验感想

- 为了体现集成学习对弱学习器的提升效果，我一开始将基学习器决策树的最大深度设为 1，但是这样设置在 RandomForest 中效果很差，AdaBoost 则相反。我经过查阅资料和思考，觉得这跟算法的目的有关。从偏差-方差的角度来看，AdaBoost 主要是减小偏差，本身算法的设计就是为了减小与真实值的差距，所以基学习器弱一点是可以通过改变下一个学习器的分布来弥补的，而 RandomForest 是减小方差，是为了减小样本的不平衡造成的影响设计的，需要基学习器足够强，所以我对 RandomForest 没有设置深度限制，获得了较好的结果
- 基学习器数量对 AUC 的影响。可以从两个算法输出的折线图发现，虽然可能存在波动，但大体上 AUC 都是随基学习器的增多而变大的，且基学习器数量较少时增长块，基学习器数量较多时增长慢。要选择最优的数量，只需取趋近于收敛的点即可。数量较少时效果差，而数量较多时训练时间长。
- 对于 AdaBoost 中差错率为 0 的处理。其实这种情况一般不会发生，因为每个基学习器都是一个弱学习器。所以我设置为了直接返回。
- AdaBoost 中对于 AUC 的计算。计算 AUC 需要输出预测概率，而书上的 AdaBoost 算法是无法输出概率的。所以我对 AdaBoost 进行了改造，将所有的权重 α_t 归一化到 $[0, 1]$，就可以得到一个预测概率。