机器学习导论
习题六

171830635, 俞星凯, yuxk@smail.nju.edu.cn

2020 年 6 月 6 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 3 可直接运行的源码 (BoostMain.py, RandomForestMain.py，不需要提交数据集)，将以上三个文件压缩成 zip 文件后上传。zip 文件格式为**学号.zip**，例如 170000001.zip；pdf 文件格式为**学号 _ 姓名.pdf**，例如 170000001_ 张三.pdf。

(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**6 月 11 日 23:59:59**。本次作业不允许缓交，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。

1 [25pts] Bayesian Network

贝叶斯网 (Bayesian Network) 是一种经典的概率图模型，请学习书本 7.5 节内容回答下面的问题：

(1) **[5pts]** 请画出下面的联合概率分布的分解式对应的贝叶斯网结构：

$$\Pr(A,B,C,D,E,F,G) = \Pr(A)\Pr(B)\Pr(C)\Pr(D|A)\Pr(E|A)\Pr(F|B,D)\Pr(G|D,E)$$

(2) **[5pts]** 请写出图1中贝叶斯网结构的联合概率分布的分解表达式。

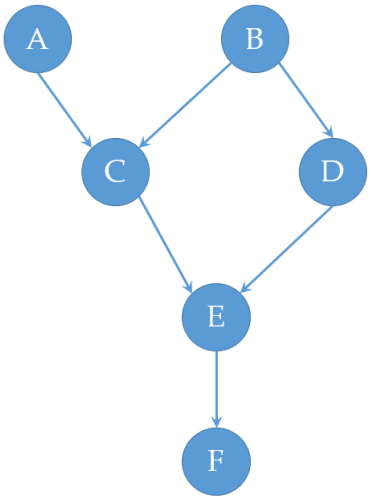


图 1: 题目 1-(2) 有向图

(3) **[15pts]** 基于第 (2) 问中的图1, 请判断表格1中的论断是否正确。首先需要作出对应的道德图，并将下面的表格填完整。

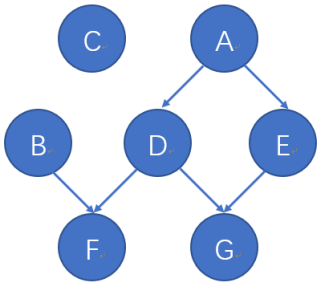
表 1: 判断表格中的论断是否正确

序号	关系	True/False	序号	关系	True/False
1	$A \perp\!\!\!\perp B$		7	$F \perp B \mid C$	
2	$A \perp B \mid C$		8	$F \perp B \mid C, D$	
3	$C \perp\!\!\!\perp D$		9	$F \perp B \mid E$	
4	$C \perp D \mid E$		10	$A \perp\!\!\!\perp F$	
5	$C \perp D \mid B, F$		11	$A \perp F \mid C$	
6	$F \perp\!\!\!\perp B$		12	$A \perp F \mid D$	

Solution.

(1) 根据联合概率表达式可以得到有向边：

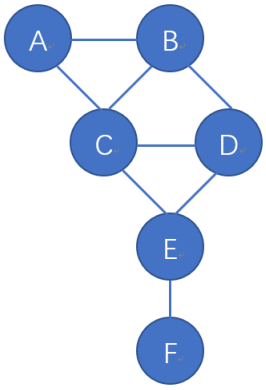
$$A \to D, A \to E, B \to F, D \to F, D \to G, E \to G$$



(2) 根据有向边可以写出联合概率表达式：

$$Pr(A,B,C,D,E,F) = Pr(A)Pr(B)Pr(C|A,B)Pr(D|B)Pr(E|C,D)Pr(F|E)$$

(3) 贝叶斯网对应道德图：



序号	关系	True/False	序号	关系	True/False
1	$A \perp\!\!\!\perp B$	True	7	$F \perp B \mid C$	False
2	$A \perp B \mid C$	False	8	$F \perp B \mid C, D$	True
3	$C \perp\!\!\!\perp D$	False	9	$F \perp B \mid E$	True
4	$C \perp D \mid E$	False	10	$A \perp\!\!\!\perp F$	False
5	$C \perp D \mid B, F$	False	11	$A \perp F \mid C$	False
6	$F \perp\!\!\!\perp B$	False	12	$A \perp F \mid D$	False

2 [35+10pts] Theoretical Analysis of k-means Algorithm

给定样本集 $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, k-means 聚类算法希望获得簇划分 $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ 使得最小化欧式距离

$$J(\gamma, \mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \|\mathbf{x}_i - \mu_j\|^2 \tag{1}$$

其中 $\mu_1, \ldots, \mu_k$ 为 k 个簇的中心 (means), $\gamma \in \mathbb{R}^{n\times k}$ 为指示矩阵 (indicator matrix) 定义如下：若 $\mathbf{x}_i$ 属于第 j 个簇, 则 $\gamma_{ij} = 1$, 否则为 0，则最经典的 k-means 聚类算法流程如算法1中所示

Algorithm 1 $k-$means Algorithm

1: Initialize $\mu_1, \ldots, \mu_k$;

2: **repeat**

3: **Step 1:** Decide the class memberships of $\{\mathbf{x}_i\}_{i=1}^n$ by assigning each of them to its nearest cluster center.

$$\gamma_{ij} = \begin{cases} 1, & \|\mathbf{x}_i - \mu_j\|^2 \le \|\mathbf{x}_i - \mu_{j'}\|^2, \forall j' \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

4: **Step 2:** For each $j \in \{1, \cdots, k\}$, recompute μ_j using the updated γ to be the center of mass of all points in C_j :

$$\mu_j = \frac{\sum_{i=1}^n \gamma_{ij}\mathbf{x}_i}{\sum_{i=1}^n \gamma_{ij}} \tag{3}$$

5: **until** the objective function J no longer changes;

(1) **[5pts]** 试证明, 在算法1中, Step 1 和 Step 2 都会使目标函数 J 的值降低.

(2) **[5pts]** 试证明, 算法1会在有限步内停止。

(3) **[10pts]** 试证明, 目标函数 J 的最小值是关于 k 的非增函数, 其中 k 是聚类簇的数目。

(4) **[15pts]** 记 $\hat{\mathbf{x}}$ 为 n 个样本的中心点, 定义如下变量,

$$\begin{aligned} T(X) &= \textstyle\sum_{i=1}^n \|\mathbf{x}_i - \hat{\mathbf{x}}\|^2 / n \\ W_j(X) &= \textstyle\sum_{i=1}^n \gamma_{ij} \|\mathbf{x}_i - \mu_j\|^2 / \sum_{i=1}^n \gamma_{ij} \\ B(X) &= \textstyle\sum_{j=1}^k \frac{\sum_{i=1}^n \gamma_{ij}}{n} \|\mu_j - \hat{\mathbf{x}}\|^2 \end{aligned}$$

试探究以上三个变量之间有什么样的等式关系？基于此请证明, k-means 聚类算法可以认为是在最小化 $W_j(X)$ 的加权平均, 同时最大化 $B(X)$.

(5) **[Bonus 10pts]** 在公式1中, 我们使用 ℓ_{2-} 范数来度量距离 (即欧式距离), 下面我们考虑使用 ℓ_{1-} 范数来度量距离

$$J'(\gamma, \mu_1, \ldots, \mu_k) = \sum_{i=1}^n \sum_{j=1}^k \gamma_{ij} \|\mathbf{x}_i - \mu_j\|_1 \tag{4}$$

- 请仿效算法1, 给出新的算法 (命名为 $k-$means-ℓ_1 算法) 以优化公式4中的目标函数 J'.
- 当样本集中存在少量异常点(outliers)时, 上述的 $k-$means-ℓ_2 和 $k-$means-ℓ_1 算法, 我们应该采用哪种算法？即哪个算法具有更好的鲁棒性？请说明理由。

Solution.

(1) 在 Step 1 中，记 $x_i(1 \le i \le n)$ 与第 g_i 个簇中心距离最近。

$$J(\gamma, \mu_1, \cdots, \mu_k) \ge \sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}\|x_i - \mu_{g_i}\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{k}\|x_i - \mu_{g_i}\|^2$$

当 $\mu_i(1 \le i \le k)$ 给定时，$\gamma_{ij} = \mathbb{I}(j = g_i)$ 可以最小化 J。
在 Step2 中，计算 J 对 μ_j 偏导

$$\begin{aligned}\frac{\partial J}{\partial \mu_j} &= \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}\|x_i - \mu_j\|^2}{\partial \mu_j} \\ &= \frac{\sum_{i=1}^{n}\gamma_{ij}\|x_i - \mu_j\|^2}{\partial \mu_j} \\ &= 2\sum_{i=1}^{n}\gamma_{ij}(\mu_j - x_i)\end{aligned}$$

令偏导为 0，得出

$$\mu_j = \frac{\sum_{i=1}^{n}\gamma_{ij}x_i}{\sum_{i=1}^{n}\gamma_{ij}}$$

当 $\gamma_{ij}(1 \le i \le n, 1 \le j \le k)$ 给定时，μ_j 按照上式取值可以最小化 J。
因此，Step1 和 Step2 都会使 J 的值降低。

(2) 显然 J 存在下界 0，并且算法使得 J 单调递减，所以算法一定收敛。

(3) 已知当簇的数量为 k 时的最优 γ 和 μ，当簇的数量为 $k+1$ 时，令第 $k+1$ 个簇中心 $\mu'_{k+1} = x_n$，并且 x_n 属于第 $k+1$ 个簇中心 $\gamma'_{n,k+1} = 1$，其余 $\gamma'_{ij}(1 \le i \le n-1, 1 \le j \le k)$ 和 $\mu'_j(1 \le j \le k)$ 不变，于是

$$\begin{aligned}J(\gamma', \mu'_1, \cdots, \mu'_{k+1}) &= \sum_{i=1}^{n}\sum_{j=1}^{k+1}\gamma'_{ij}\|x_i - \mu'_j\|^2 \\ &= \sum_{i=1}^{n-1}\sum_{j=1}^{k+1}\gamma'_{ij}\|x_i - \mu'_j\|^2 + \sum_{j=1}^{k+1}\gamma'_{ij}\|x_n - \mu'_j\|^2 \\ &= \sum_{i=1}^{n-1}\sum_{j=1}^{k}\gamma_{ij}\|x_i - \mu_j\|^2 + \|x_n - \mu'_{k+1}\|^2 \\ &= \sum_{i=1}^{n-1}\sum_{j=1}^{k}\gamma_{ij}\|x_i - \mu_j\|^2 \\ &\le J(\gamma, \mu_1, \cdots, \mu_k)\end{aligned}$$

所以 J 的最小值是关于 k 的非增函数。

(4)

$$\begin{aligned} nT(X) &= \sum_{i=1}^{n} \|x_i - \hat{x}\|^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}\|x_i - \hat{x}\|^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}\|x_i - \mu_j + \mu_j - \hat{x}\|^2 \\ &= \sum_{j=1}^{k}\sum_{i=1}^{n} \gamma_{ij}\|x_i - \mu_j\|^2 + \sum_{j=1}^{k}\sum_{i=1}^{n} \gamma_{ij}\|\mu_j - \hat{x}\|^2 + 2\sum_{j=1}^{k}(\mu_j - \hat{x})\sum_{i=1}^{n}\gamma_{ij}(x_i - \mu_j) \\ &= \sum_{j=1}^{k}\sum_{i=1}^{n} \gamma_{ij}W_j(X) + nB(X) + 0 \end{aligned}$$

因为 $J = \sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}W_j(X)$，所以 k-means 算法最小化 J 等价于最小化 $W_j(X)$ 的加权平均。同时 J 与 $nB(X)$ 之和为定值，最小化 J 等价于最大化 $B(X)$。

(5) 令 $S_j = \{x_i|\gamma_{ij} = 1\}$，即属于第 j 个簇的样本集。

Algorithm 2 $k-$means Algorithm

1: Initialize $\mu_1, \ldots, \mu_k$;

2: **repeat**

3: **Step 1:** Decide the class memberships of $\{\mathbf{x}_i\}_{i=1}^{n}$ by assigning each of them to its nearest cluster center.

$$\gamma_{ij} = \begin{cases} 1, & \|\mathbf{x}_i - \mu_j\|_1 \leq \|\mathbf{x}_i - \mu_{j'}\|_1, \forall j' \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

4: **Step 2:** For each $j \in \{1, \cdots, k\}$, recompute μ_j using the updated γ to be the center of mass of all points in C_j :

$$\mu_j = median_{\mathbf{x}\in S_j}(\mathbf{x}) \tag{6}$$

5: **until** the objective function J no longer changes;

k-means-ℓ_1 算法具有更好鲁棒性，因为当异常点数量不多时，中值操作可以屏蔽它们的影响，而均值操作则会被极大影响。

3 [40pts] Coding: Ensemble Methods

本次实验中我们将结合两种经典的集成学习思想：Boosting 和 Bagging，对集成学习方法进行实践。本次实验选取 UCI 数据集 Adult，此数据集为一个二分类数据集，具体信息可参照链

接，为了方便大家使用数据集，已经提前对数据集稍作处理，并划分为训练集和测试集，数据集文件夹为 adult_dataset。

由于 Adult 是一个类别不平衡数据集，本次实验选用 AUC 作为评价分类器性能的评价指标，可调用sklearn 算法包对 AUC 指标进行计算。

(1) 本次实验要求使用 Python3 编写，要求代码分布于两个文件中，BoostMain.py, RandomForestMain.py ，调用这两个文件就能完成一次所实现分类器的训练和测试；

(2) **[35pts]** 本次实验要求编程实现如下功能：

- **[10pts]** 结合教材 8.2 节中图 8.3 所示的算法伪代码实现 AdaBoost 算法，基分类器选用决策树，基分类器可调用 sklearn 中决策树的实现；
- **[10pts]** 结合教材 8.3.2 节所述，实现随机森林算法，基分类器仍可调用 sklearn 中决策树的实现，也可以手动实现，在实验报告中请给出随机森林的算法伪代码；
- **[10pts]** 结合 AdaBoost 和随机森林的实现，调查基学习器数量对分类器训练效果的影响，具体操作如下：分别对 AdaBoost 和随机森林，给定基分类器数目，在训练数据集上用 5 折交叉验证得到验证 AUC 评价。在实验报告中用折线图的形式报告实验结果，折线图横轴为基分类器数目，纵轴为 AUC 指标，图中有两条线分别对应 AdaBoost 和随机森林，基分类器数目选取范围请自行决定；
- **[5pts]** 根据参数调查结果，对 AdaBoost 和随机森林选取最好的基分类器数目，在训练数据集上进行训练，在实验报告中报告在测试集上的 AUC 指标；

(3) **[5pts]** 在实验报告中，除了报告上述要求报告的内容外还需要展现实验过程，实验报告需要有层次和条理性，能让读者仅通过实验报告便能了解实验的目的，过程和结果。

实验报告.

实现一个类 AdaBoost 封装算法，基学习器使用 sklearn 中的决策分类树。

首先需要保存基学习器个数 n 和决策树参数 $kwargs$。

```
def __init__(self, n, **kwargs):
  self.n = n
  self.kwargs = kwargs
```

然后实现 $fit()$ 方法，重复训练新的决策树，计算对应权重 $alpha$，并迭代样本权重 w。

```
def fit(self, X, y):
  self.clfs = []
  self.alphas = []
  w = np.ones_like(y) / y.shape[0]
  for i in range(self.n):
    clf = DecisionTreeClassifier(max_depth=1, **self.kwargs)
    clf.fit(X, y, sample_weight=w)
    z = clf.predict(X)
    err = 1 - accuracy_score(y, z, sample_weight=w)
    if err > 0.5:
      break
    alpha = 0.5 * np.log((1-err) / err)
    w *= np.exp(-alpha * y * z)
```

```
    w /= w.sum()
    self.clfs.append(clf)
    self.alphas.append(alpha)
  return self
```

最后实现 *predict*() 方法，将各决策树预测结果根据 *alpha* 线性组合，求出符号即为预测结果。

```
def predict(self, X):
  y = np.zeros(X.shape[0])
  for alpha, clf in zip(self.alphas, self.clfs):
    y += alpha * clf.predict(X)
  return np.sign(y)
```

实现一个类 RandomForest 封装算法，同样需要保存基学习器个数 n 和决策树参数 *kwargs*，不再赘述。直接看 *fit*() 方法，需要限制每次划分特征是从随机特征子集中选择的，并且训练集基于 Bootstrap 生成。

```
def fit(self, X, y):
  self.clfs = []
  for i in range(self.n):
    clf = DecisionTreeClassifier(max_features='log2', **self.kwargs)
    idx = np.random.choice(np.arange(y.shape[0]), y.shape[0])
    clf.fit(X[idx], y[idx])
    self.clfs.append(clf)
  return self
```

predict() 实现类似 AdaBoost，区别是不需要加权。

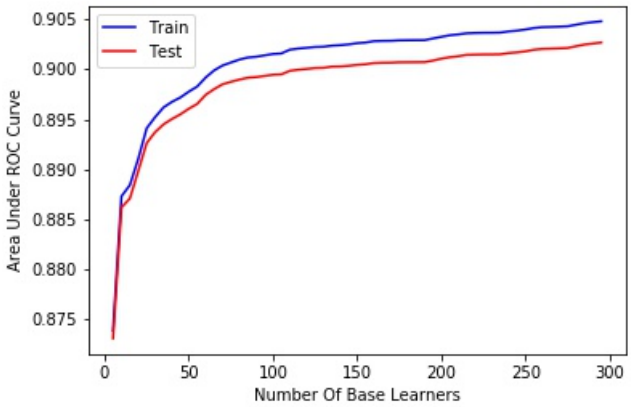
```
def predict(self, X):
  y = np.zeros(X.shape[0])
  for clf in self.clfs:
    y += clf.predict(X)
  return np.sign(y)
```

在训练集上的 5 折交叉验证 AUC 是通过 sklearn 中的 *cross_val_score*() 函数实现，特别注意的是，该函数需要类实现 *get_params*() 和 *predict_proba* 方法，前者返回模型参数，后者计算各类预测概率。这里我们以 AdaBoost 中的实现为例，RandomForest 的实现类似。

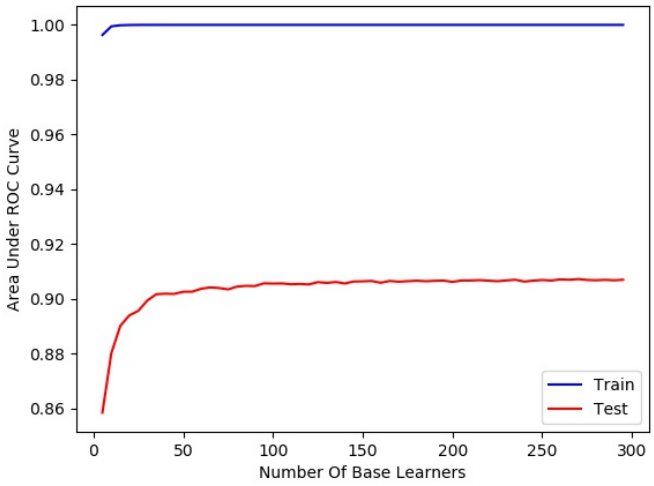
```
def get_params(self, deep=True):
  return {'n': self.n}

def predict_proba(self, X):
  y = np.zeros((X.shape[0], 2))
  for alpha, clf in zip(self.alphas, self.clfs):
    y += alpha * clf.predict_proba(X)
  return y / y.sum(1, keepdims=True)
```

下图是 AdaBoost 的 AUC-基学习器个数折线图，可以看到初始随着基学习器个数增加，AUC 提升显著，后来逐渐平稳上升，并且训练集和测试集 AUC 差距逐步拉大。



下图是 RandomForest 的 AUC-基学习器个数折线图，可以看到训练集很快就近乎完美，测试集性能上升不如 AdaBoost 平稳，并且大约 130 个分类器就达到收敛。



对于 AdaBoost 取定基学习器个数为 120，得到分类器在训练集上训练之后，测试集 AUC 为 0.8940。对于 RandomForest 取定基学习器个数为 130，得到结果为 0.9047。综合来看，RandomForest 略优于 AdaBoost。