机器学习

第 5 章决策树与随机森林

欧阳毅

浙江工商大学
管理工程与电子商务学院

2023 年 3 月 12 日

目录

决策树 I

决策树 (decision tree) 是一种基本的分类与回归方法. 其主要优点是模型具有可读性, 分类速度快.

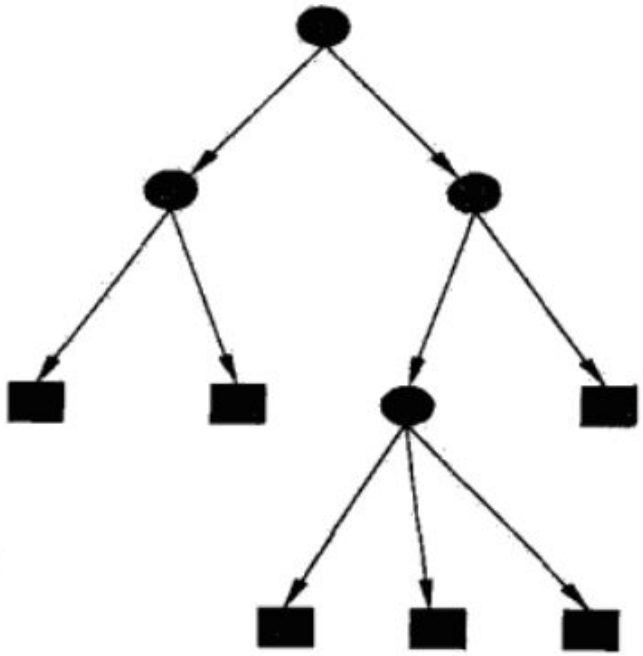
1. 利用训练数据, 根据损失函数最小化的原则建立决策树模型.
2. 预测时, 对新的数据, 利用决策树模型进行分类决策树学习
   - 特征选择
   - 决策树的生成
   - 决策树剪枝决策树容易形成过拟合，因而需要对树进行剪枝，通常剪枝有向前或向后剪枝

决策树模型与学习 I

定义 (决策树) 分类决策树模型是一种描述对实例进行分类的树形结构. 决策树由结点和有向边组成, 结点有两种类型: 内部结点 (internal node) 和叶结点 (leaf node). 内部结点表示一个特征或属性, 叶结点表示一个类

决策树模型与学习 II

图: 一个决策树的示意图. 图中圆和方框分别表示内部结点和叶结点



决策树与 if-then 规则 I

可以将决策树看成一个 if-then 规则的集合. 将决策树转换成 if-then 规则的过程:

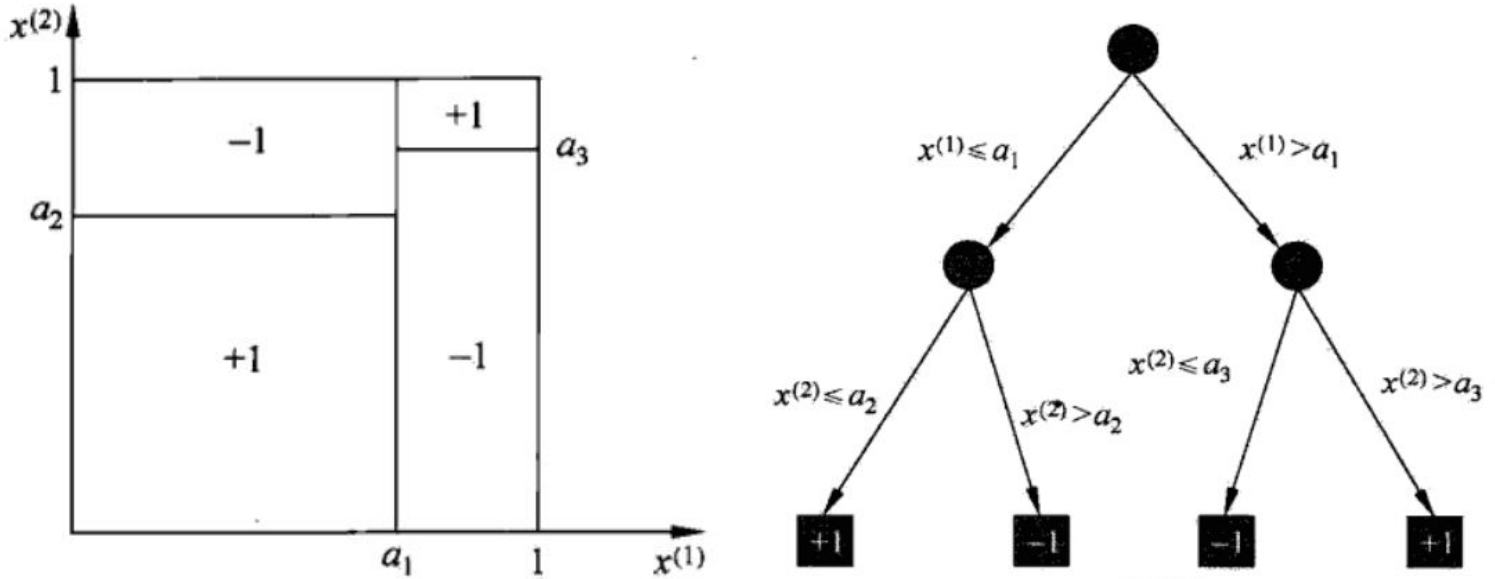
- 由决策树的根结点到叶结点的每一条路径构建一条规则;
- 路径上内部结点的特征对应着规则的条件, 而叶结点的类对应着规则的结论.
- 互斥并且完备

决策树的概率表现形式 I

条件概率分布由各个单元给定条件下类的条件概率分布组成.

- 假设 X 为表示特征的随机变量, Y 为表示类的随机变量, 那么这个条件概率分布可以表示为 $P(Y|X)$,X 取值于给定划分下单元的集合,Y 取值于类的集合.

图: 决策树的概率表现形式



特征选择 I

- 特征选择问题

例 1 表是一个由 15 个样本组成的贷款申请训练数据数据包括贷款申请人的 4 个特征 (属性):

- 第 1 个特征是年龄, 有 3 个可能值: 青年, 中年, 老年;
- 第 2 个特征是有工作, 有 2 个可能值: 是, 否;
- 第 3 个特征是有自己的房子, 有 2 个可能值: 是, 否;
- 第 4 个特征是信贷情况, 有 3 个可能值: 非常好, 好, 一般.
- 表的最后一列是类别, 是否同意贷款, 取 2 个值: 是, 否

当新的客户提出贷款申请时, 根据申请人的特征利用决策树决定是否批准贷款申请.

特征选择 II

图: 不同特征决定的不同决策数，选择哪个特征更好?

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

信息增益 (information gain) 就能够很好地表示这一直观的准则

信息增益-熵

熵 (entropy) 是表示随机变量不确定性的度量. 设 X 是一个取有限个值的离散随机变量, 其概率分布为

$$P(X = x_i) = p_i, i = 1, 2, ..., n$$

则随机变量 X 的熵定义为

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

在式中, 若 $p_i = 0$, 则定义 $0 \log 0 = 0$. 通常, 式中的对数以 2 为底或以 e 为底 (自然对数), 熵只依赖于 X 的分布, 而与 X 的取值无关, 所以也可将 X 的熵记作 $H(p)$, 即

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i$$

信息增益-熵

- 熵越大, 随机变量的不确定性就越大.

$$0 \leq H(p) \leq \log n$$

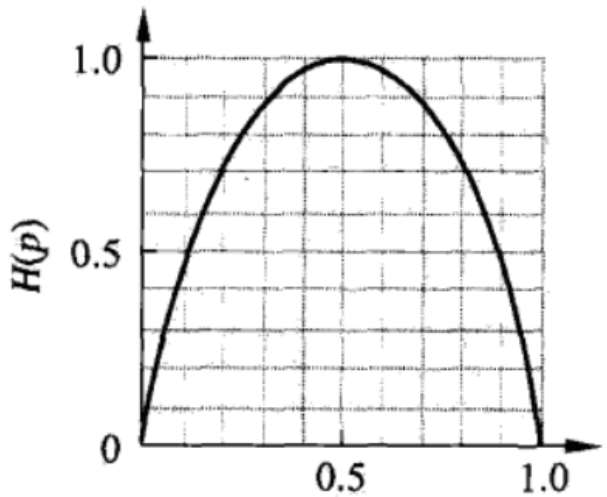
- 当随机变量只取两个值, 例如 1,0 时, 即 X 的分布为

$$P(X=1)=p, P(X=0)=1-p, 0 \leq p \leq 1$$

熵为

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p)$$

图: 分别为贝努利分布时熵与概率的关系



信息增益-条件熵

条件熵：随机变量 x 给定的条件下随机变量 Y 的条件熵 (conditional entropy)$H(Y|X)$, 定义为:
X 给定条件下 Y 的条件概率分布的熵对 X 的数学期望

$$H(Y|X) = \sum_{i=1}^{n} P(X = x_i) H(Y|X = x_i)$$

信息增益 (information gain)

定义 (信息增益) 特征 A 对训练数据集 D 的信息增益 g(D,A), 定义为集合 D 的经验熵 H(D) 与特征 A 给定条件下 D 的经验条件熵 $H(D|A)$ 之差, 即

$$g(D, A) = H(D) - H(D|A)$$

信息增益 已知特征的信息对类 Y 的信息的不确定性减少的程度

- ID3 算法: 应用信息增益准则选择特征

信息增益的算法 I

输入：训练数据集 D 和特征 A;$|D|$ 表示 D 样本个数. 设有 K 个类 $\sum_{k=1}^{K}|C_k| = |D|$

- 计算数据集 D 的经验熵 H(D)

$$H(D) = -\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

$$H(D) = -\frac{9}{15}\log_2\frac{9}{15} - \frac{6}{15}\log_2\frac{6}{15} = 0.971$$

信息增益的算法 II

- 计算特征 A 对数据集 D 的经验条件熵 $H(D|A)$，D_i 中属于类 C_k 的样本的集合为 D_{ik}

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

- 计算信息增益

$$g(D, A) = H(D) - H(D|A)$$

实例分析 I

例 2: 对表 1 所给数据集 D, 根据信息增益准则选择最优特征

解 首先计算经验熵 $H(D|A)$; 然后计算各特征对数据集 D 的信息增益. (A_1, A_2, A_3, A_4 表示年龄, 有工作、有自己的房子和信贷情况); 最后, 比较各特征的信息增益值

- (1) 计算 A_1 信息增益 D1,D2,D3 分别是 D 中 A(年龄) 取值为青年、中年和老年的样本子集,

$$
\begin{aligned}
g(D, A_1) &= H(D) - [\tfrac{5}{15}H(D_1) + \tfrac{5}{15}H(D_2) + \tfrac{5}{15}H(D_3)] \\
&= 0.971 - [\tfrac{5}{15}(-\tfrac{2}{5}\log_2\tfrac{2}{5} - \tfrac{3}{5}\log_2\tfrac{3}{5}) \\
&\quad + \tfrac{5}{15}(-\tfrac{3}{5}\log_2\tfrac{3}{5} - \tfrac{2}{5}\log_2\tfrac{2}{5}) \\
&\quad + \tfrac{5}{15}(-\tfrac{4}{5}\log_2\tfrac{4}{5} - \tfrac{1}{5}\log_2\tfrac{1}{5}) \\
&= 0.083
\end{aligned} \tag{1}
$$

实例分析 II

- (2) 计算 A_2 信息增益 D1,D2 分别表示是否有工作的样本子集，则

$$\begin{aligned} g(D, A_2) &= H(D) - [\tfrac{5}{15} H(D_1) + \tfrac{10}{15} H(D_2)] \\ &= 0.324 \end{aligned} \tag{2}$$

- (3) 计算 A_3 信息增益 D1,D2 分别表示是否有房子的样本子集，则

$$\begin{aligned} g(D, A_3) &= H(D) - [\tfrac{6}{15} H(D_1) + \tfrac{9}{15} H(D_2)] \\ &= 0.420 \end{aligned} \tag{3}$$

- (4) 计算 A_4 信息增益 D1,D2，D3 分别表示是否有借贷情况的样本子集，则

$$g(D, A_4) = 0.363 \tag{4}$$

. 由于特征 A_3(有自己的房子) 的信息增益值最大, 所以选择特征 A_3 作为最优特征

信息增益比 I

信息增益的问题：若有个特征（如：地址）为互不相同的情况？

定义

信息增益比 特征 A 对训练数据集 D 的信息增益比 $g_R(D, A)$ 定义为其信息增益 $g(D, A)$ 与训练数据集 D 的经验熵 $H_A(D)$ 之比:

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)} \tag{5}$$

- C4.5 算法：应用信息增益比准则选择特征

目录

ID3 算法 I

ID3 算法的核心是在决策树各个结点上应用信息增益准则选择特征, 递归地构建决策树. 具体方法是:

- 从根结点开始, 选择信息增益最大的特征作为结点的特征, 由该特征的不同取值建立子结点;
- 再对子结点递归地调用以上方法, 构建决策树; 直到所有特征的信息增益均很小或没有特征可以选择为止.
- 最后得到一个决策树.

ID3 算法 I

输入：训练数据集 D, 特征集 A, 阈值 E

输出：决策树 T

- (1) 若 D 中所有实例属于同一类 C_k, 则将类 $T = C_k$ 作为该结点的类标记, 若 A=∅, 则 T 为单结点树, 并将 D 中实例数最大的类 C_k 作为该结点的类标记, 返回 T
- (2) 否则, 计算 A 中各特征对 D 的信息增益, 选择信息增益最大的特征 A_g
- (3) 如果 A_g 的信息增益小于阈值 E, 则置 T 为单结点树, 并将 D 中实例数最大的类 C_k 作为该结点的类标记, 返回 T;
- (4) 否则, 对 A_g 的每一可能值 a_i, 依 $A_g = a_i$ 将 D 分割为若干非空子集 D_i, 将 D_i 中实例数最大的类作为标记, 构建子结点, 由结点及其子结点构成树 T, 返回 T;
- (5) 对第 i 个子结点, 以 D 为训练集, 以 $A - \{A_g\}$ 为特征集, 递归地调用步 (1) 步 (4), 得到子树 T, 返回 T

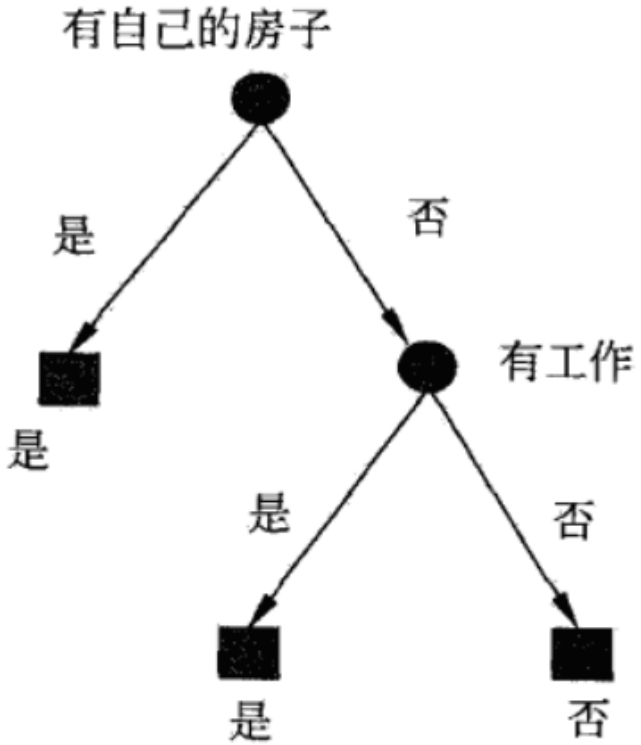
ID3 算法-例子 I

例 对表 1 的训练数据集, 利用 ID3 算法建立决策树

解 利用例 2 的结果, 由于特征 A3(有自己的房子) 的信息增益值最大, 所以选择特征 A3 作为根结点的特征. 它将训练数据集 D 划分为两个子集 D1(A_3 取值为“是”) 和 D2(A_3 取值为“否”). 由于 D1 只有同一类的样本点, 所以它成为一个叶结点, 结点的类标记为“是”

- 对 D2 则需从特征 A_1(年龄),A_2(有工作) 和 A_4(信贷情况) 中选择新的特征. 计算各个特征的信息增益:
  - $g(D_2, A_1) = H(D_2) - H(D_2|A_1) = 0.918 - 0667 = 0.251$
  - $g(D_2, A_2) = H(D_2) - H(D_2|A_2) = 0918$
  - $g(D_2, A_4) = H(D_2) - H(D_2|A_4) = 0.474$

ID3 算法-例子 II



ID3 算法只有树的生成, 所以该算法生成的树容易产生过拟合

目录

C4.5 的生成算法 I

C4.5 在生成的过程中, 用信息增益比来选择特征.(C4.5 的生成算法)

输入 : 训练数据集 D, 特征集 A, 阈值 E

输出 : 决策树 T.

- (1) 如果 D 中所有实例属于同一类 C_k, 则置 T 为单结点树, 并将 C_k 作为该结点的类, 返回 T;
- (2) 如果 A=∅, 则置 T 为单结点树, 并将 D 中实例数最大的类 C_k 作为该结点的类, 返回 T;
- (3) 否则, 按式 $(g_R(D,A) = \frac{g(D,A)}{H_A(D)})$ 计算 A 中各特征对 D 的信息增益比, 选择信息增益比最大的特征 A_g.
- (4) 如果 A_g 的信息增益比小于阈值 E, 则置 T 为单结点树, 并将 D 中实例数最大的类 C_k 作为该结点的类, 返回 T;

C4.5 的生成算法 II

- (5) 否则, 对 A_g 的每一可能值 a_i, 依 $A_g = a_i$ 将 D 分割为子集若干非空 D_i, 将 D_i 中实例数最大的类作为标记, 构建子结点, 由结点及其子结点构成树 T, 返回 T;
- (6) 对结点 i, 以 D_i 为训练集, 以 $A - \{A_g\}$ 为特征集, 递归地调用步 (1)-步 (5) 得到子树 T_i, 返回 T_i

目录

CART 算法原理 I

分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为：

$$Gini(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2 \tag{6}$$

注：p_k 表示选中的样本属于 k 类别的概率，则这个样本被分错的概率为 $(1-p_k)$. 对于给定的样本集合 D，其基尼指数为：

$$Gini(D) = 1 - \sum_{k=1}^{K} (\frac{|C_k|}{|D|})^2 \tag{7}$$

注：这里 C_k 是 D 中属于第 k 类的样本，K 是类的个数。

CART 算法原理 II

如果样本集合 D 根据特征 X 是否取某一可能值 $\hat{x}$ 被分割成 D_1 和 D_2 两部分，即：

$$D_1 = \{(x, y) \in D | x = \hat{x}\}, D_2 = D - D_1$$

则在特征 X 的条件下，集合 D 的基尼指数定义为：

$$Gini(D, X) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{8}$$

基尼指数 Gini(D) 表示集合 D 的不确定性，基尼指数 Gini(D,X) 表示经 $X = \hat{x}$ 分割后集合 D 的不确定性。基尼指数值越大，样本集合的不确定性也就越大，这一点跟熵相似。

CART 算法 I

算法 2（CART 生成算法）

输入：训练数据集 D，停止计算的条件；

输出：CART 决策树。

根据训练数据集，从根节点开始，递归地对每个结点进行以下操作，构建二叉决策树：

- (1) 设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数。
- (2) 选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点。将训练数据集依特征分配到两个子节点中去。
- (3) 对两个子节点递归地调用步骤 (1)、(2)，直至满足停止条件。
- (4) 生成 CART 决策树。

算法停止的条件是节点中的样本个数小于预定的阈值，或样本集的基尼指数小于预定阈值，或者没有更多特征。

CART 算法 I

```
from sklearn import tree
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
wine = load_wine()

Xtrain, Xtest, Ytrain, Ytest = train_test_split(
            wine.data,wine.target,test_size=0.3)
#建立模型
clf = tree.DecisionTreeClassifier(criterion="gini")
clf = clf.fit(Xtrain, Ytrain)
print(clf.apply(Xtest))
#predict返回每个测试样本的分类/回归结果
print(clf.predict(Xtest))
score = clf.score(Xtest, Ytest) #返回预测的准确度
```

目录

随机森林 I

随机森林是决策树的集成学习（Ensemble Learning）实现，方法是同时训练多个决策树，综合考虑多个结果，最终得出预测结果。随机森林较决策树的优势：

- 1)、减少过拟合的情况
  2)、减少预测的 variance，预测值不会因为训练数据的大小变化而剧烈变化

算法停止的条件是节点中的样本个数小于预定的阈值，或样本集的基尼指数小于预定阈值，或者没有更多特征。

随机森林 I

- bagging 是一种在原始数据集上，通过有放回抽样分别选出 k 个新数据集，来训练分类器的集成算法。分类器之间没有依赖关系。
- 随机森林在 bagging 的基础上更进一步：
  1. 样本的随机：从样本集中用 Bootstrap 随机选取 n 个样本
  2. 特征的随机：从所有属性中随机选取 K 个属性，选择最佳分割属性作为节点建立 CART 决策树
  3. 重复以上两步 m 次，即建立了 m 棵 CART 决策树
  4. 这 m 个 CART 形成随机森林，通过投票表决结果，决定数据属于哪一类（投票机制有一票否决制、少数服从多数、加权多数）

随机森林 I

```
from numpy import genfromtxt
from sklearn.ensemble import RandomForestClassifier
# 加载数据
dataset = genfromtxt('data.csv', delimiter=",")
x = dataset[1:, 0:4]
y = dataset[1:, 4]
clf = RandomForestClassifier(n_jobs=2, oob_score=True)
#袋外错误率oob error（out-of-bag error）
clf = clf.fit(x, y)
# 预测
print(clf.predict_proba([[33, 0, 80, 1]]))
```

目录

提升算法 I

提升算法也称为 boosting 算法，它是将弱学习算法提升为强学习算法的一类算法，可用来提升弱分类器的准确度。

强学习：在概率近似正确框架中，一个概念或者类，如果存在一个多项式的学习算法能够学习它，且正确率很高，称为强可学习。

弱学习：一个概念或者类，如果存在一个多项式的学习算法能够学习它，且它的学习正确率仅仅比随机猜测略好，则为弱可学习。

Adaboost I

Adaboost（adaptive Boosting）算法实际上就是 boosting 算法的一个升级版。

1. 其算法原理: 通过调整样本权重和弱分类器权值，从训练出的弱分类器中筛选出权值系数最小的弱分类器组合成一个最终强分类器。
2. 与 Boosting 算法不同的是
   它是使用整个训练集来训练弱学习机，其中训练样本在每次迭代的过程中都会重新被赋予一个权重，在上一个弱学习机错误的基础上进行学习来构建一个更加强大的分类器。

Adaboost 算法 I

Require: : 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$; 其中 $y = \{-1, +1\}$; 弱分类器算法

Ensure: : 最终分类器 $G(x)$

1: 初始化训练数据的分布权重

$$D_1 = \{w_{11}\ w_{12}\ ....\ w_{1N}\}, w_{1i} = \frac{1}{N}\ i = 1\ 2\ 3, ..., N$$

2: 要生成 M 个基学习器，则对于 m = 1,2,.....,M:

(a)、使用具有权重分布 D_m 的训练数据进行生成基学习器:

$$G_m(x) : X \to \{-1, +1\}$$

(b)、计算 $G_m(x)$ 在训练数据集上的分类误差率:

$$e_m = \sum_{i=1}^{N} P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

Adaboost 算法 II

(c) 计算 $G_m(x)$ 的系数

$$a_m = \frac{1}{2} log \frac{1 - e_m}{e_m}$$

(d) 更新训练数据集的权重分布

$$D_{m+1} = \{w_{m+1,1}\ w_{m+1,2}\ \cdots\ w_{m+1,N}\}$$

$$w_{m+1,i} = \frac{w_{m,i}}{Z_m} exp(-a_m y_i G_m(x_i)), i = 1, 2..., N$$

3: 构建基本分离器的线性组合

$$f(x) = \sum_{m=1}^{M} a_m G_m(x)$$

$$G(x) = sign(f(x))$$

Adaboost 算法 I

```
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

if __name__ == "__main__":
    X, Y = load_breast_cancer(return_X_y=True)
    x1, x2, y1, y2 = train_test_split(X, Y,
         test_size=1 / 3, random_state=0)

    clf = AdaBoostClassifier(base_estimator
        =DecisionTreeClassifier(max_depth=1))
    clf.fit(x1, y1)
    print("预测正确率:", clf.score(x2, y2))
```