集成学习

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

学习目标

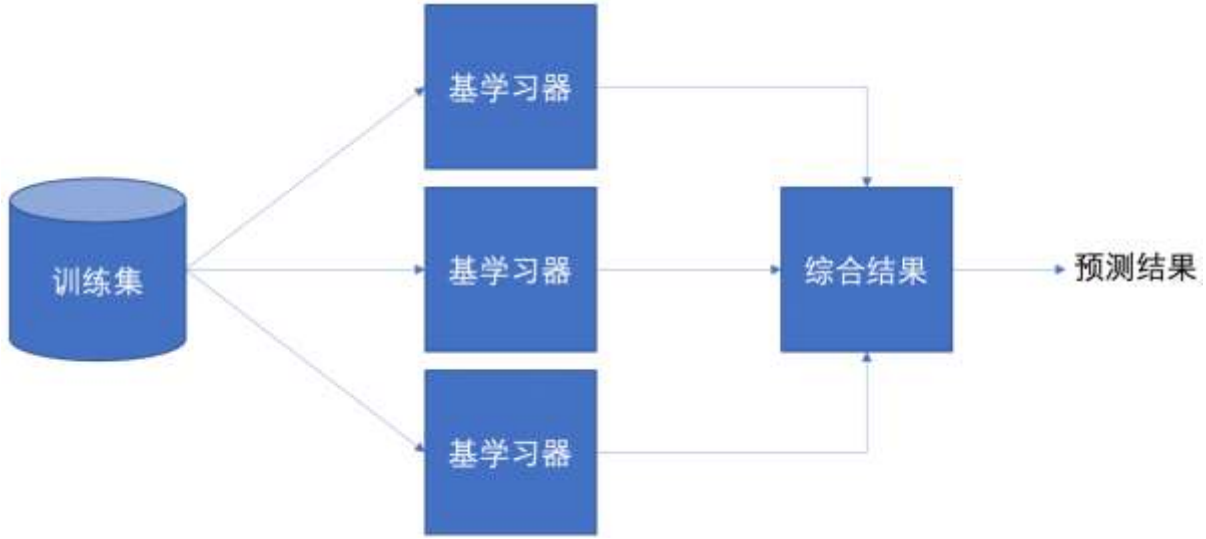
Learning Objectives

1. 知道集成学习是什么?
2. 了解集成学习的分类
3. 理解bagging集成的思想
4. 理解boosting集成的思想

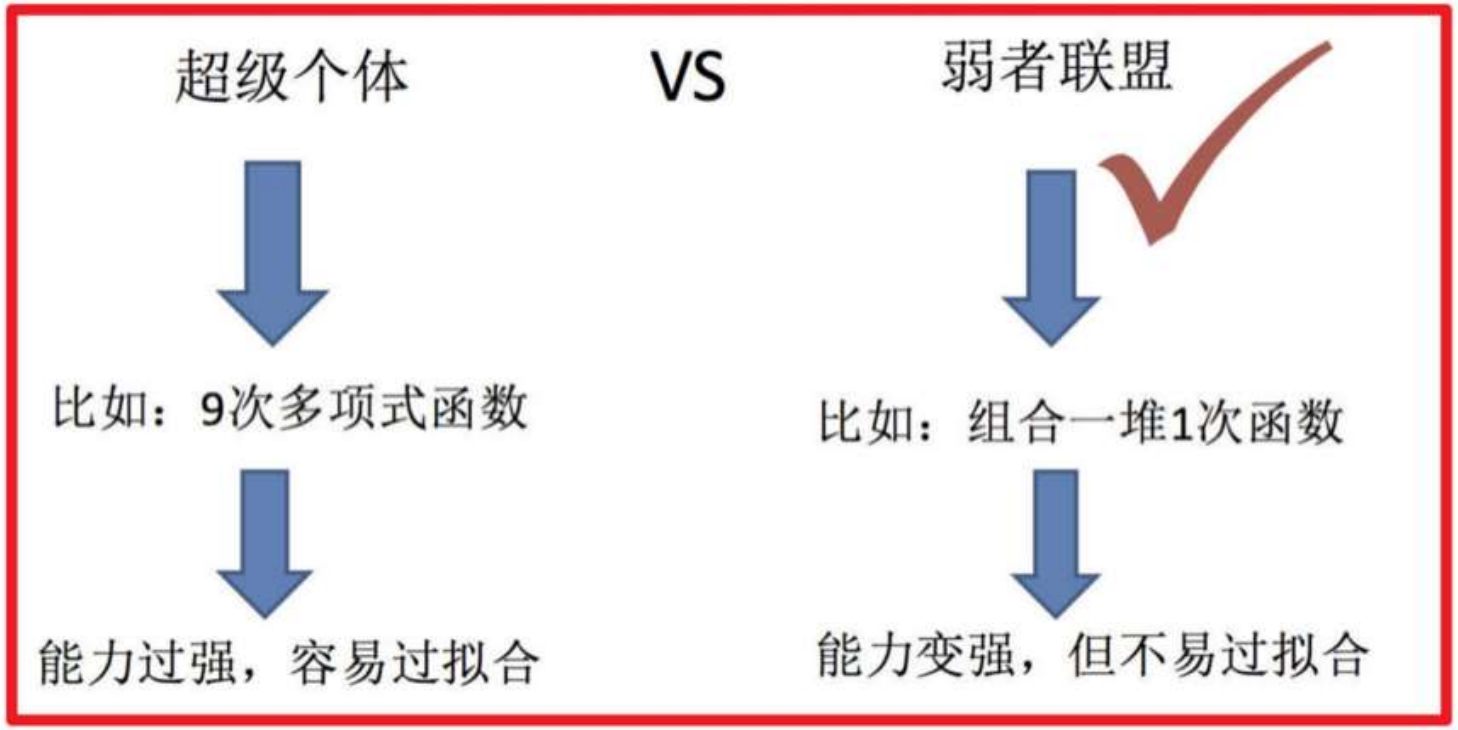
集成学习

集成学习是机器学习中的一种思想，它通过多个模型的组合形成一个精度更高的模型，参与组合的模型称为弱学习器（基学习器）。

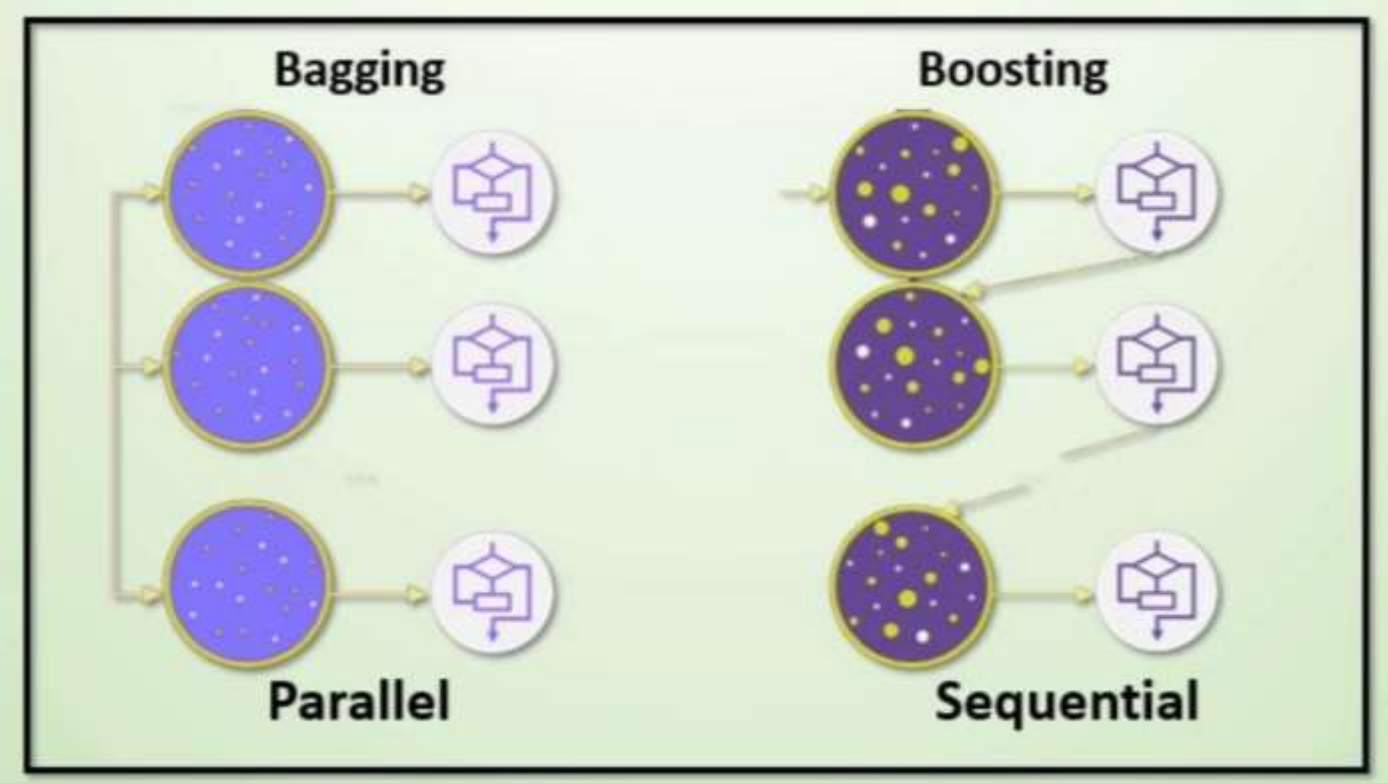
训练时，使用训练集依次训练出这些弱学习器，对未知样本进行预测时，使用这些弱学习器联合进行预测。



集成学习



集成学习分类



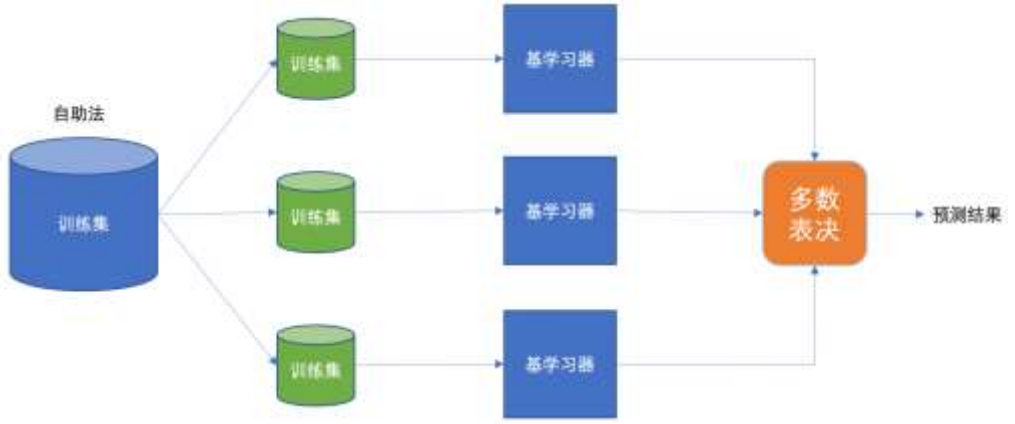
- Bagging：随机森林
- Boosting：Adaboost、GBDT、XGBoost、LightGBM

集成学习 - Bagging思想

- Bagging思想
  - 又称装袋算法或者自举汇聚法
  - 有放回的抽样(bootstrap抽样)产生不同的训练集，从而训练不同的学习器
  - 通过平权投票、多数表决的方式决定预测结果

    在分类问题中，会使用多数投票统计结果

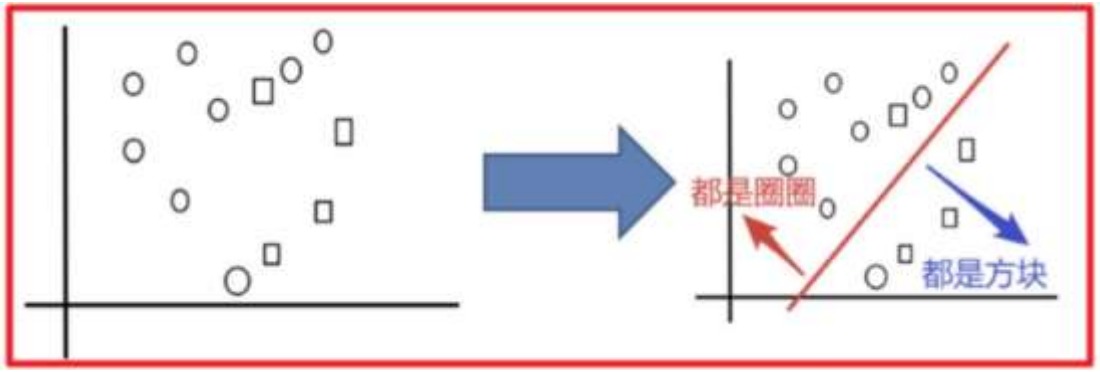
    在回归问题中，会使用求均值统计结果
  - 弱学习器可以并行训练
  - 基本的弱学习器算法模型，如: Linear、Ridge、Lasso、Logistic、Softmax、ID3、C4.5、CART、SVM、KNN均可

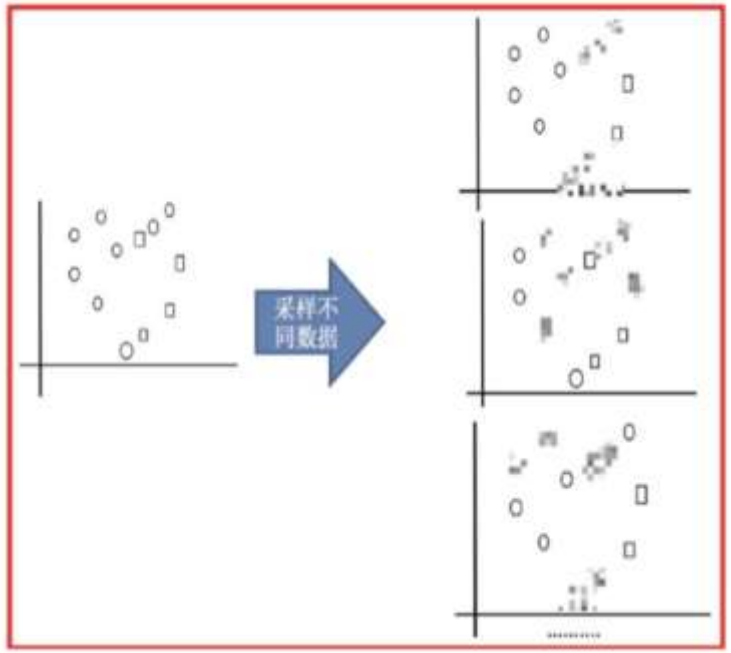


集成学习 - Bagging思想

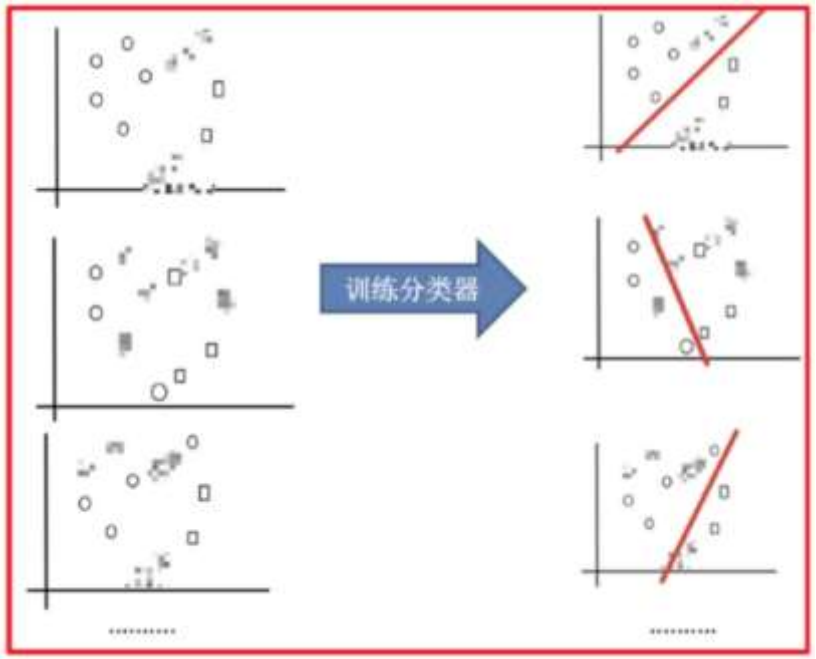
- Bagging思想图

目标：把右图的圈和方块进行分类

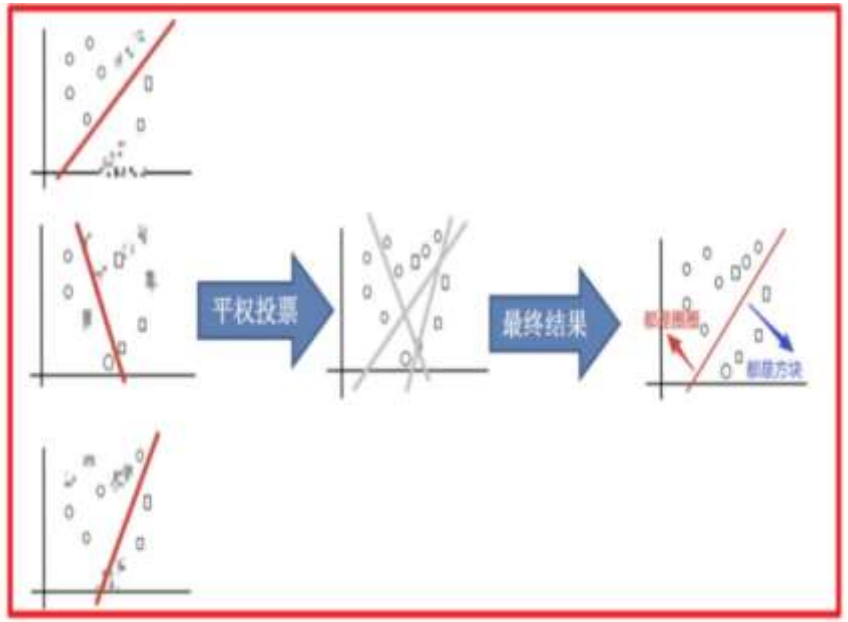




1 采样不同数据集



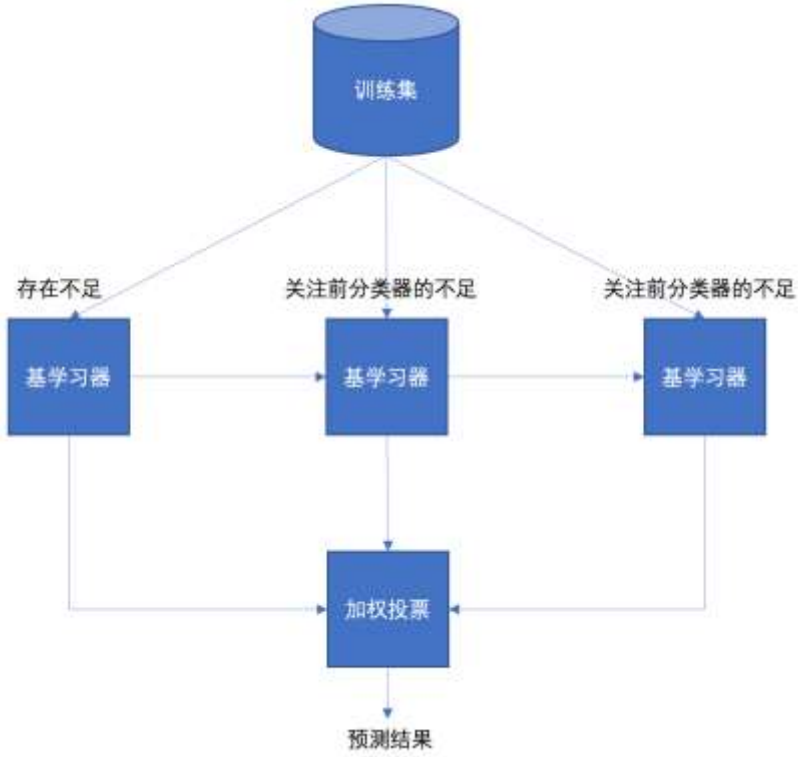
2 训练分类器



3 平权投票，获取最终结果

集成学习 - Boosting思想

- Boosting思想
  - 每一个训练器重点关注前一个训练器不足的地方进行训练
  - 通过加权投票的方式，得出预测结果
  - 串行的训练方式



集成学习 - Boosting思想

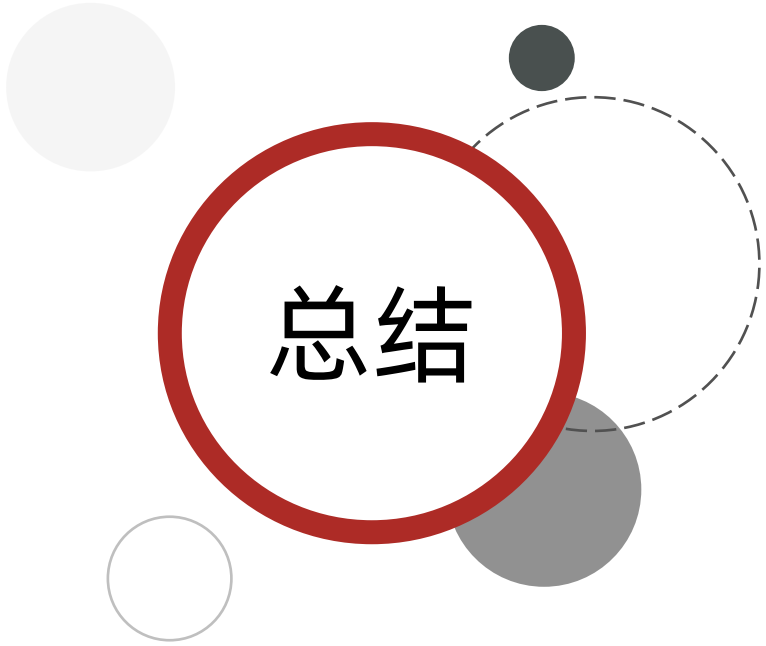
- Boosting思想生活中的举例

滚球兽→亚古兽→暴龙兽→机械暴龙兽→战斗暴龙兽

- 随着学习的积累从弱到强
- 每新加入一个弱学习器，整体能力就会得到提升
- 代表算法：Adaboost，GBDT，XGBoost，LightGBM

Bagging & Boosting对比

	Bagging	Boosting
数据采样	对数据进行有放回的采样训练	全部样本，根据前一轮学习结果调整数据的重要性
投票方式	所有学习器平权投票	对学习器进行加权投票
学习顺序	并行的，每个学习器没有依赖关系	串行，学习有先后顺序



1 集成学习是什么？

- 多个弱学习器组合成一个更强大的学习器，解决单一预测，进步一得到更好性能。

2 bagging思想

- 有放回的抽样
- 平权投票、多数表决的方式决定预测结果
- 并行训练

3 boosting思想

- 重点关注前一个训练器不足的地方进行训练
- 加权投票的方式
- 串行的训练方式

练习

1、Bagging 是什么意思，下面说法正确的是（）

A 一个精度比较高的分类器

B 从数据集中随机采样出多个子集，每个子集训练一个弱分类器，然后将这些弱分类器组合成一个强分类器

C 从特征集中随机选择多个特征，使用这些特征训练一个弱分类器，然后将多个弱分类器组合成一个强分类器

D 以上都不是

正确答案：BC

目录

Contents

学习目标

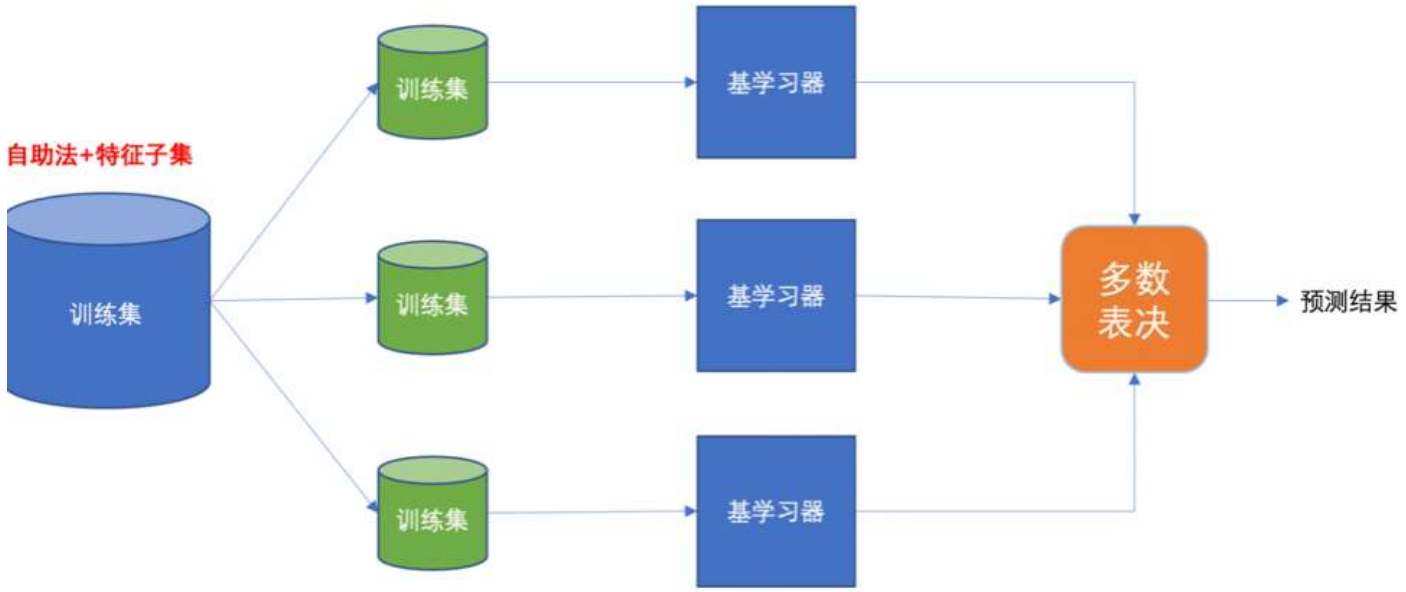
Learning Objectives

1. 理解随机森林的构建方法
2. 知道随机森林的API
3. 能够使用随机森林完成分类任务

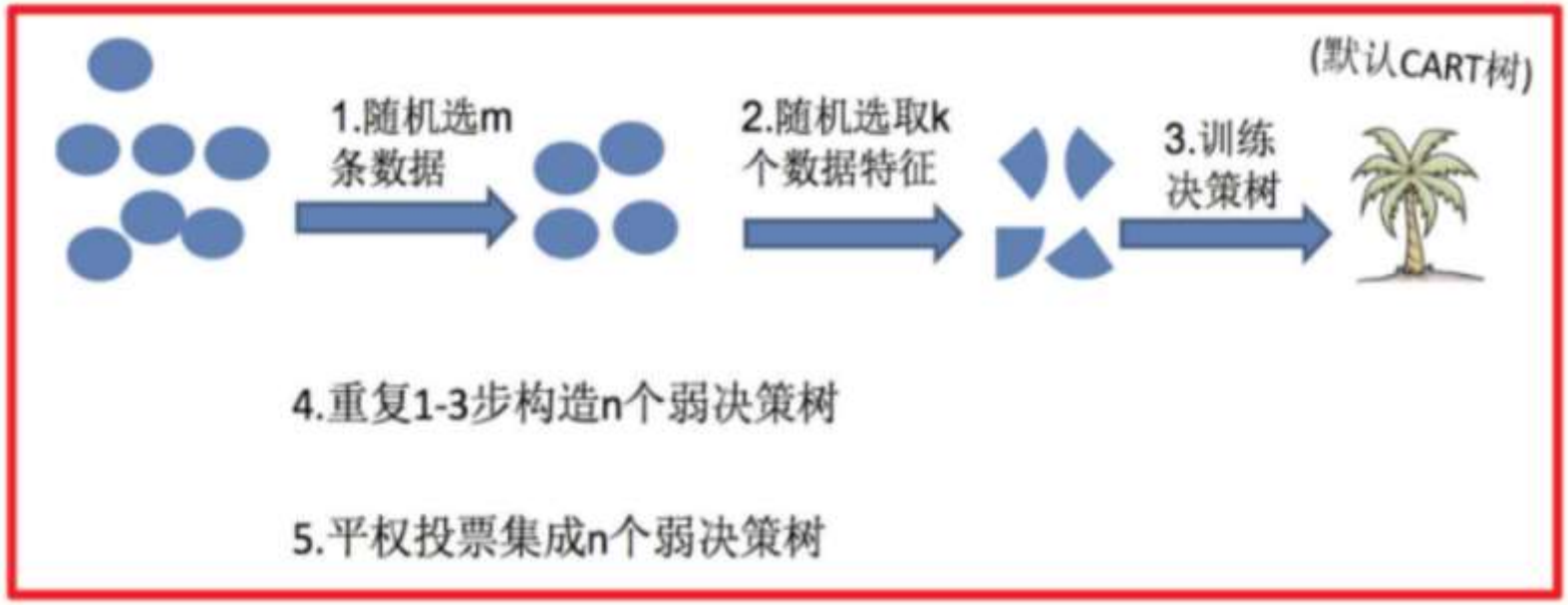
随机森林算法

随机森林是基于 Bagging 思想实现的一种集成学习算法，采用决策树模型作为每一个弱学习器。

- 训练：
  （1）有放回的产生训练样本
  （2）随机挑选 n 个特征（n 小于总特征数量）
- 预测：平权投票，多数表决输出预测结果



随机森林步骤


1.随机选m
条数据
2.随机选取k
个数据特征
3.训练
决策树
(默认CART树)
4.重复1-3步构造n个弱决策树
5.平权投票集成n个弱决策树

随机森林算法 - 概念

- 思考题1：为什么要随机抽样训练集？

  如果不进行随机抽样，每棵树的训练集都一样，那么最终训练出的树分类结果也是完全一样。

- 思考题2：为什么要有放回地抽样？

  - 如果不是有放回的抽样，那么每棵树的训练样本都是不同的，都是没有交集的，这样每棵树都是“有偏的”，也就是说每棵树训练出来都是有很大的差异的；而随机森林最后分类取决于多棵树（弱分类器）的投票表决。
  - 综上：弱学习器的训练样本既有交集也有差异数据，更容易发挥投票表决效果

随机森林算法 - API

sklearn.ensemble.RandomForestClassifier()

- **n_estimators**：决策树数量，(default = 10)
- **Criterion**：entropy、或者 gini, (default = gini)
- **max_depth**：指定树的最大深度，(default = None 表示树会尽可能的生长)
- **max_features="auto”**, 决策树构建时使用的最大特征数量
  - If "auto", then `max_features=sqrt(n_features)`.
  - If "sqrt", then `max_features=sqrt(n_features)`(same as "auto").
  - If "log2", then `max_features=log2(n_features)`.
  - If None, then `max_features=n_features`.
- **bootstrap**：是否采用有放回抽样，如果为 False 将会使用全部训练样本，(default = True)

随机森林算法 - API

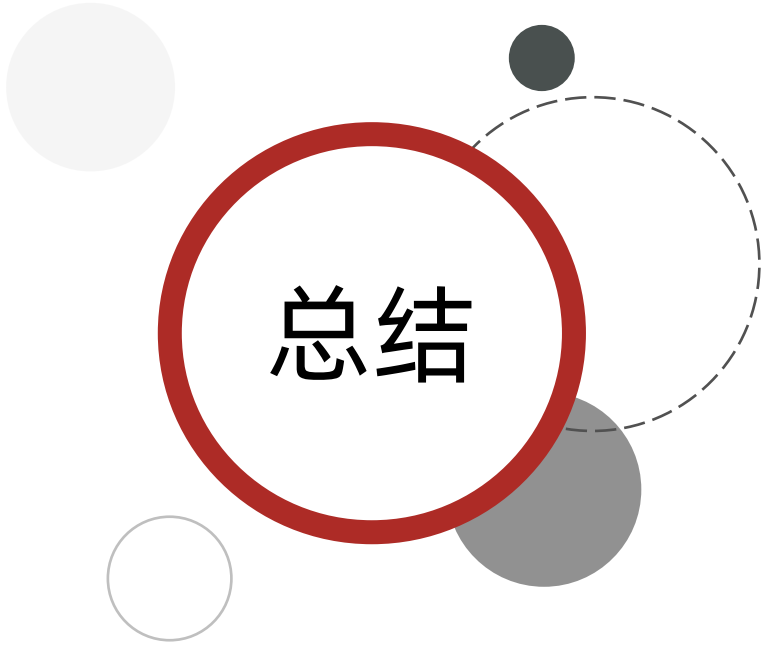
- **min_samples_split**： 结点分裂所需最小样本数，(default = 2)
  - 如果节点样本数少于min_samples_split，则不会再进行划分.
  - 如果样本量不大，不需要设置这个值.
  - 如果样本量数量级非常大，则推荐增大这个值.
- **min_samples_leaf**： 叶子节点的最小样本数，(default = 1)
  - 如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝.
  - 较小的叶子结点样本数量使模型更容易捕捉训练数据中的噪声.
- **min_impurity_split:** 节点划分最小不纯度
  - 如果某节点的不纯度(基尼系数，均方差)小于这个阈值，则该节点不再生成子节点，并变为叶子节点.
  - 一般不推荐改动默认值1e-7。

泰坦尼克号案例

```
# 1.导入依赖包
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

def dm01_随机森林():
    # 2.获取数据集
    titan = pd.read_csv("./data/titanic/train.csv")

    # 3.数据处理,
    # 3.1 确定特征值和目标值
    x = titan[["Pclass", "Age", "Sex"]].copy()
    y = titan["Survived"]. copy()
    # 3.2 处理数据-处理缺失值
    x['Age'].fillna(value=titan["Age"].mean(), inplace=True)
    print(x.head())
    # 3.3 one-hot编码
    x = pd.get_dummies(x)
    # 3.4 数据集划分
    x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=22, test_size=0.2)
```

泰坦尼克号案例

```
# 4.模型训练
# 4.1 使用决策树进行模型训练和评估
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
dtc_y_pred = dtc.predict(x_test)
accuracy = dtc.score(x_test, y_test)
print('单一决策树accuracy-->\n', accuracy)

# 4.2 随机森林进行模型训练和评估
rfc = RandomForestClassifier(max_depth=6, random_state=9)
rfc.fit(x_train, y_train)
rfc_y_pred = rfc.predict(x_test)
accuracy = rfc.score(x_test, y_test)
print('随机森林进accuracy-->\n', accuracy)

# 4.3 随机森林 交叉验证网格搜索 进行模型训练和评估
estimator = RandomForestClassifier()
param = {"n_estimators": [40, 50, 60, 70], "max_depth": [2, 4, 6, 8, 10], "random_state":[9]}
grid_search = GridSearchCV(estimator, param_grid=param, cv=2)
grid_search.fit(x_train, y_train)
accuracy = grid_search.score(x_test, y_test)
print("随机森林网格搜索accuracy:", accuracy)
print(grid_search.best_estimator_)
```



1 随机森林概念

bagging思想的代表算法，bagging+决策树

2 随机森林构建过程

1.随机选数据

2.随机选特征

3.训练弱学习器

4.重复1-3训练n个

5.平权投票

3 随机森林API

sklearn.ensemble.RandomForestClassifier()

练习

1、请对下列随机森林的构建方法进行排序：

A）重复采样，构建出多颗决策树

B）随机选取部分样本，并随机选取部分特征交给其中一颗决策树训练

C）如果是分类场景则采用平权投票的方式决定最终随机森林的预测结果，
如果是回归场景则采用简单平均法获取最终结果

D）将相同的测试数据交给所有构建出来的决策树进行结果预测

正确答案： **B→A→D→C**

目录

Contents

学习目标

Learning Objectives

1. 理解Adaboost算法的思想
2. 知道Adaboost的构建过程
3. 实践泰坦尼克号生存预测案例

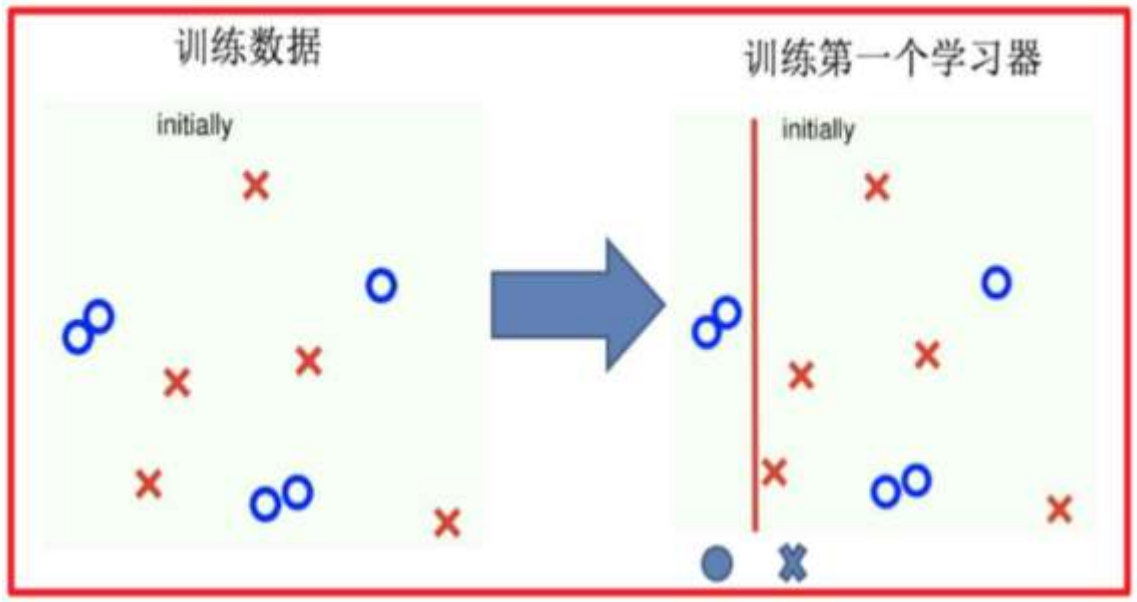
Adaboost算法

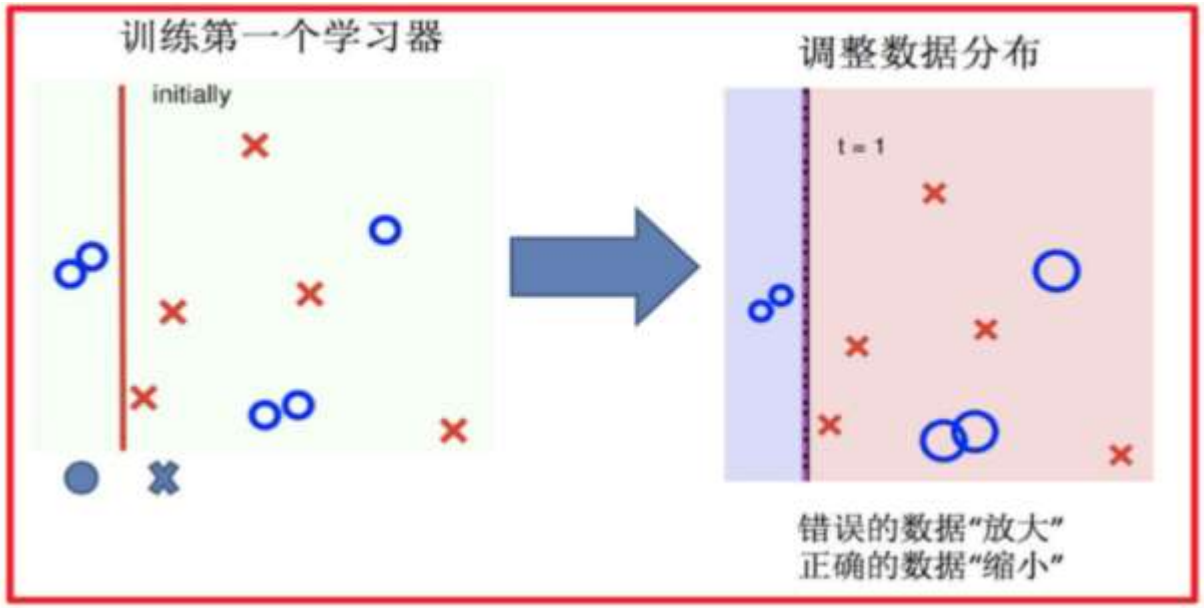
Adaptive Boosting(自适应提升)基于 Boosting思想实现的一种集成学习算法

核心思想是通过逐步提高那些被前一步分类错误的样本的权重来训练一个强分类器。

1.训练第一个学习器

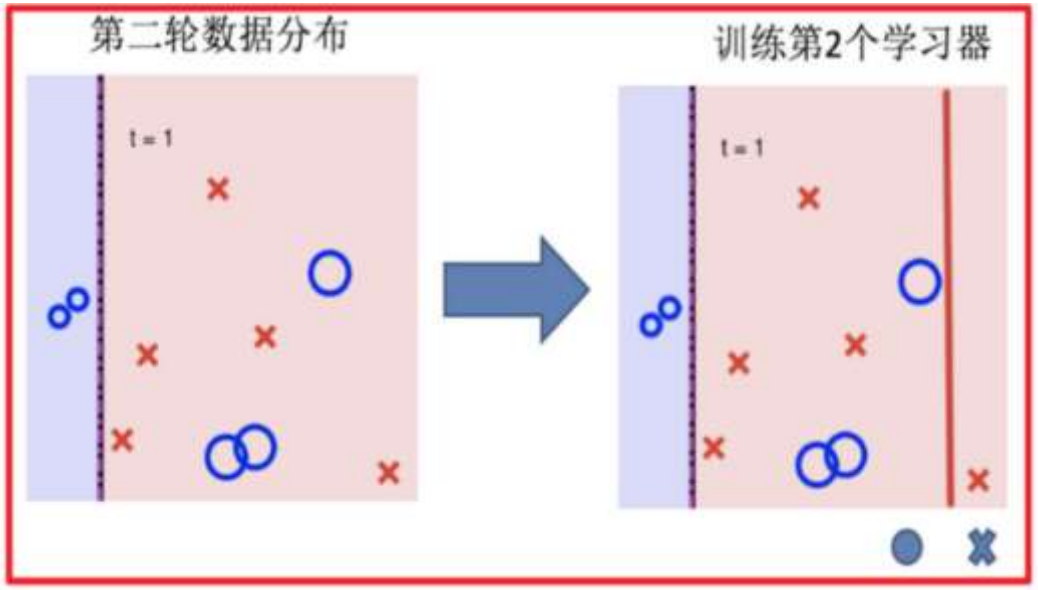
2.调整数据分布



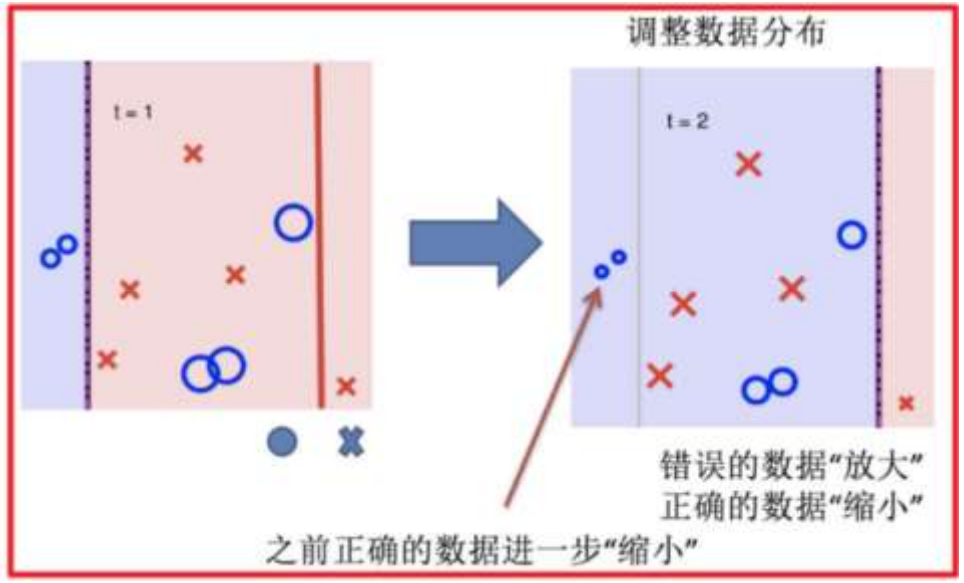


Adaboost算法

3.训练第二个学习器

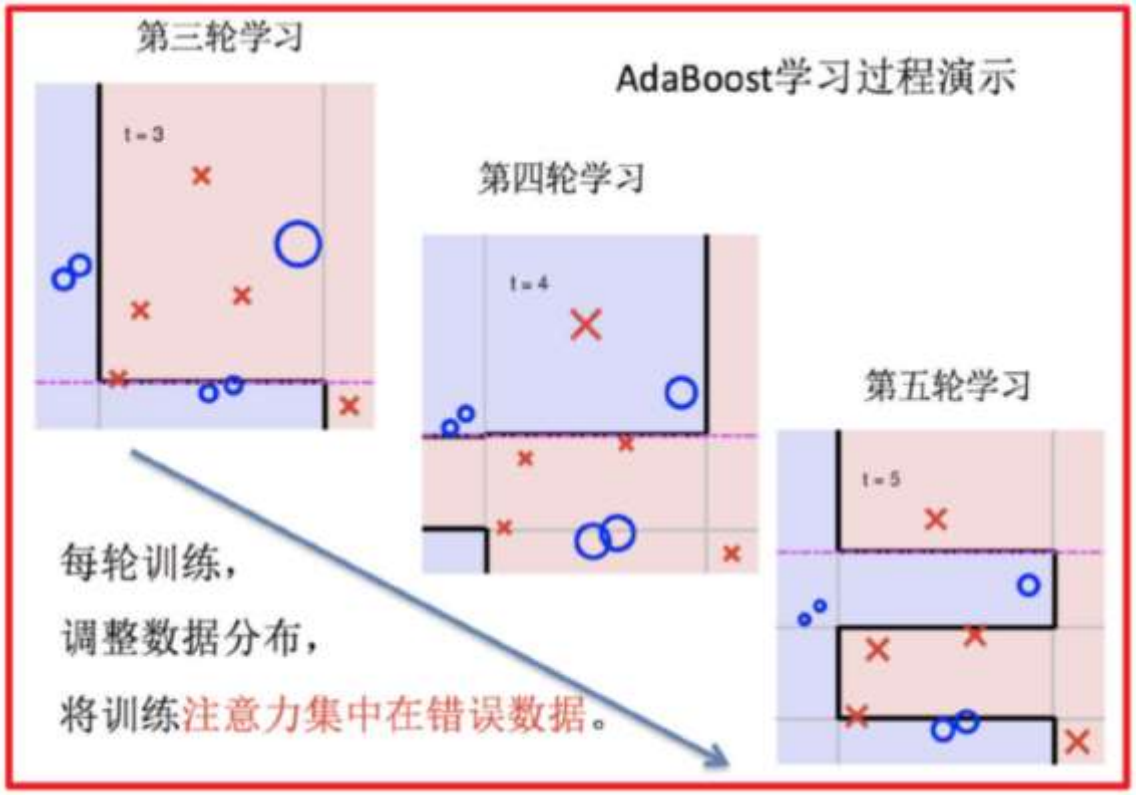


4.再次调整数据分布

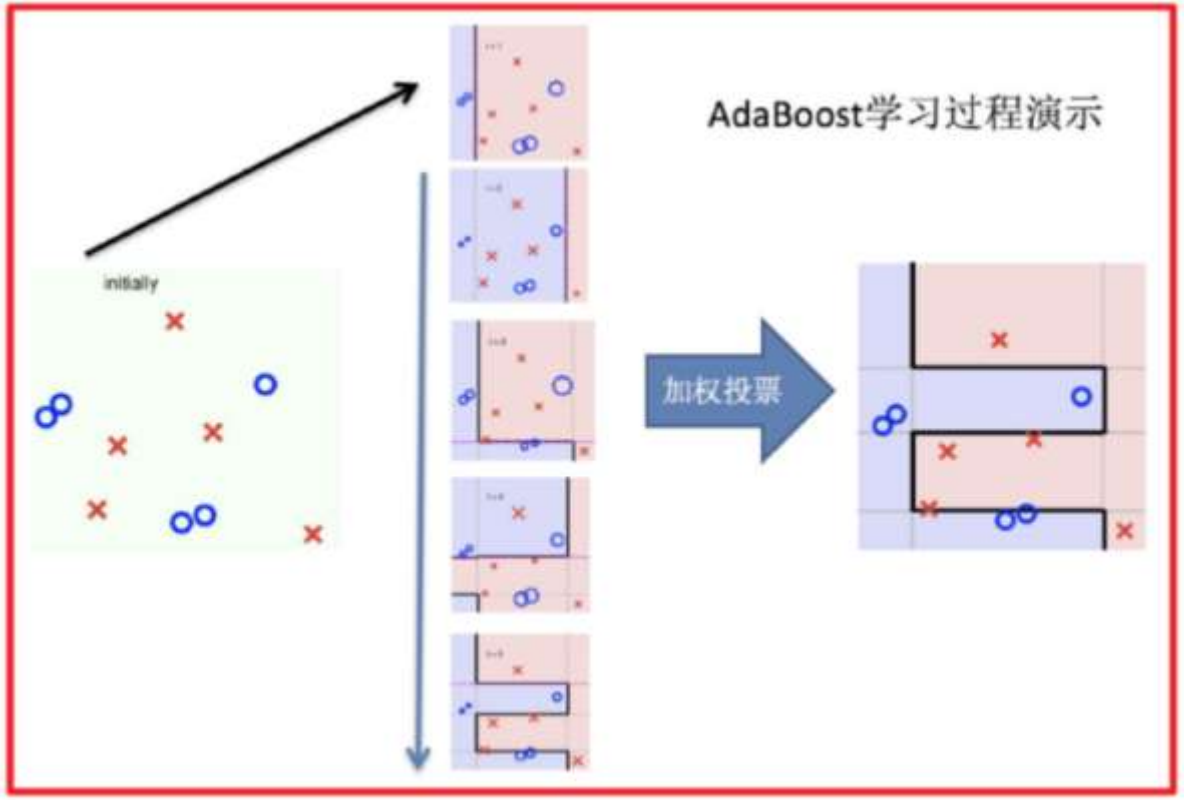


Adaboost算法

5.依次训练学习器，调整数据分布



6.整体过程实现



Adaboost算法推导

1 初始化训练数据权重相等，训练第 1 个学习器

- 如果有 100 个样本，则每个样本的初始化权重为：1/100
- 根据预测结果找一个错误率最小的分裂点，计算、更新：样本权重、模型权重

2 根据新权重的样本集 训练第 2 个学习器

- 根据预测结果找一个错误率最小的分裂点计算、更新：样本权重、模型权重

3 迭代训练在前一个学习器的基础上，根据新的样本权重训练当前学习器

- 直到训练出 m 个弱学习器

4 m个弱学习器集成预测公式：H(x) = sign($\sum_{i=1}^{m} a_i h_i(x)$)

- a_i 为模型的权重，输出结果大于 0 则归为正类，小于 0 则归为负类。

5 模型权重计算公式：

$a_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ a_t 为模型权重

ε_t表示第 t 个弱学习器的错误率

6 样本权重计算公式：

$$D_{t+1}(x) = \frac{D_t(x)}{Z_t} * \begin{cases} e^{-a_t}, \text{预测值} = \text{真实值} \\ e^{a_t}, \text{预测值} \neq \text{真实值} \end{cases}$$

其中Z_t 为归一化值(所有样本权重总和)

$D_t(x)$为样本权重

a_t 为模型权重

Adaboost算法 – 构建过程

已知训练数据见下面表格，假设弱分类器由 x 产生，预测结果使该分类器在训练数据集上的分类误差率最低，试用 Adaboost 算法学习一个强分类器。

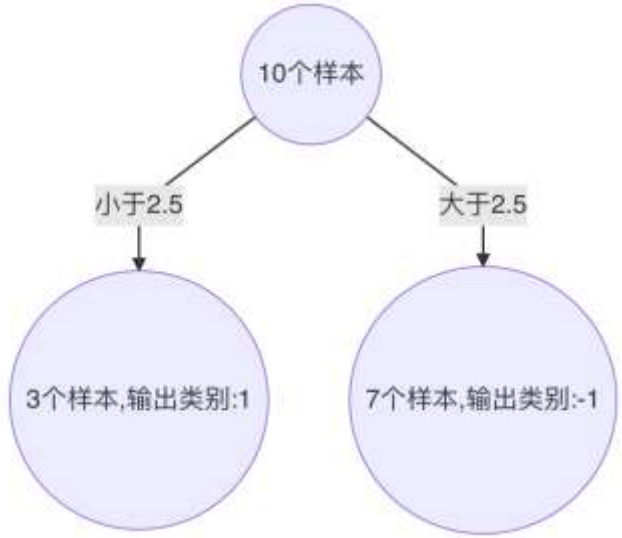
序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

Adaboost算法-构建第1个弱分类器

1. 初始化工作：初始化 10 个样本的权重，每个样本的权重为：0.1

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
y	1	1	1	-1	-1	-1	1	1	1	-1

1. 初始化工作：初始化 10 个样本的权重，每个样本的权重为：0.1
2. 构建第一个基学习器：
    1. 寻找最优分裂点
        1. 对特征值 x 进行排序，确定分裂点为：0.5、1.5、2.5、3.5、4.5、5.5、6.5、7.5、8.5
        2. 当以 0.5 为分裂点时，有 5 个样本分类错误
        3. 当以 1.5 为分裂点时，有 4 个样本分类错误
        4. 当以 2.5 为分裂点时，有 3 个样本分类错误
        5. 当以 3.5 为分裂点时，有 4 个样本分类错误
        6. 当以 4.5 为分裂点时，有 5 个样本分类错误
        7. 当以 5.5 为分裂点时，有 4 个样本分类错误
        8. 当以 6.5 为分裂点时，有 5 个样本分类错误
        9. 当以 7.5 为分裂点时，有 4 个样本分类错误
        10. 当以 8.5 为分裂点时，有 3 个样本分类错误
        11. 最终，选择以 2.5 作为分裂点，计算得出基学习器错误率为：3/10=0.3



Adaboost算法 -构建第1个弱分类器

$a_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ a_t 为模型权重

ε_t错误率 = 分类错误样本权重之和

$D_{t+1}(x) = \frac{D_t(x)}{Z_t} * \begin{cases} e^{-a_t}, 预测值 = 真实值 \\ e^{a_t}, 预测值 \neq 真实值 \end{cases}$

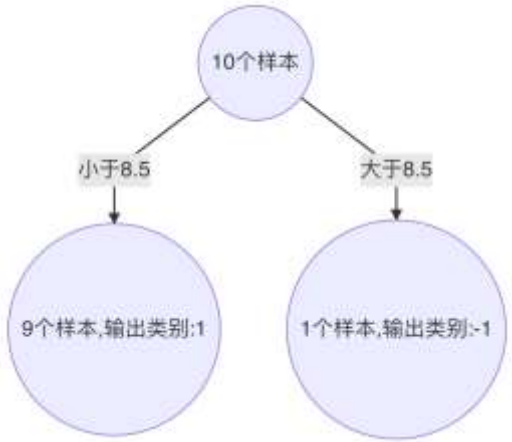
2. 计算模型权重: `1/2 * np.log((1-0.3)/0.3)=0.4236`
3. 更新样本权重:
    1. 分类正确样本为：1、2、3、4、5、6、10 共 7 个，其计算公式为：$e^{-\alpha_t}$，则正确样本权重变化系数为：$e^{-0.4236} = 0.6547$
    2. 分类错误样本为：7、8、9 共 3 个，其计算公式为：e^{α_t}，则错误样本权重变化系数为：$e^{0.4236} = 1.5275$
    3. 样本 1、2、3、4、5、6、10 权重值为：`0.06547` 【0.1*0.6547】
    4. 样本 7、8、9 的样本权重值为：`0.15275` 【0.1*1.5275】
    5. 归一化 Z_t 值为：`0.06547 * 7 + 0.15275 * 3 = 0.9165`
    6. 样本 1、2、3、4、5、6、10 最终权重值为：`0.07143`
    7. 样本 7、8、9 的样本权重值为：`0.1667`

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
w	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16667	0.16667	0.16667	0.07143
y	1	1	1	-1	-1	-1	1	1	1	-1

Adaboost算法-构建第2个弱学习器

1. 寻找最优分裂点：
    1. 对特征值 x 进行排序，确定分裂点为：0.5、1.5、2.5、3.5、4.5、5.5、6.5、7.5、8.5
    2. 当以 0.5 为分裂点时，有 5 个样本分类错误，错误率为：**0.07143 * 2 + 0.16667 * 3 = 0.64287**
    3. 当以 1.5 为分裂点时，有 4 个样本分类错误，错误率为：**0.07143 * 1 + 0.16667 * 3 = 0.57144**
    4. 当以 2.5 为分裂点时，有 3 个样本分类错误，错误率为：**0.16667 * 3 = 0.50001**

        。。。。。。
    5. 当以 8.5 为分裂点时，有 3 个样本分类错误，错误率为：**0.07143 * 3 = 0.21429**
    6. 最终，选择以 8.5 作为分裂点，计算得出基学习器错误率为：0.21429



Adaboost算法 -构建第2个弱学习器

$a_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ a_t 为模型权重

ε_t错误率 = 分类错误样本权重之和

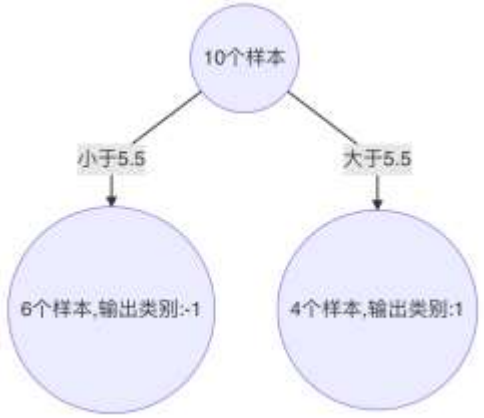
$$D_{t+1}(x) = \frac{D_t(x)}{Z_t} * \begin{cases} e^{-a_t}, \text{预测值} = \text{真实值} \\ e^{a_t}, \text{预测值} \neq \text{真实值} \end{cases}$$

2. 计算模型权重：`1/2 * np.log((1-0.21429)/0.21429)=0.64963`
3. 分类正确的样本：1、2、3、7、8、9、10，其权重调整系数为：0.5222
4. 分类错误的样本：4、5、6，其权重调整系数为：1.9148
5. 分类正确样本权重值：
    1. 样本 1、2、3、10为：0.0373 【0.07143*0.5222】
    2. 样本 7、8、9 为：0.087 【0.16667*0.5222】
6. 分类错误样本权重值：0.1368 【0.07143*1.9148】
7. 归一化 Z_t 值为：`0.0373 * 4 + 0.087 * 3 + 0.1368 * 3 = 0.8206`
8. 最终权重：
    1. 样本 1、2、3、10为：0.0455
    2. 样本 7、8、9 为：0.1060
    3. 样本 4、5、6 为：0.1667

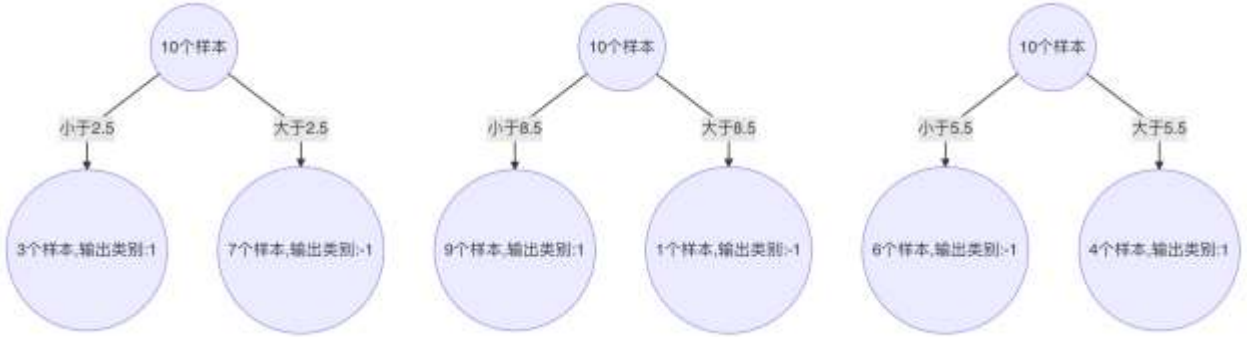
序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
w	0.0455	0.0455	0.0455	0.16667	0.16667	0.16667	0.1060	0.1060	0.1060	0.0455
y	1	1	1	-1	-1	-1	1	1	1	-1

Adaboost算法 - 构建第3个弱学习器

错误率：0.1820，模型权重：0.7514



- 最终强学习器



H(x) = sign(0.4236 * $h_1(x)$ + 0.64963 * $h_2(x)$ + 0.7515 * $h_3(x)$)

若H(x)的值大于0则归于正类；H(x)的值小于0归于负类

X = 3 带入公式为： 0.4236*(-1) + 0.64963*(1) + 0.7514*(-1) = -0.52537 < 0 负类

案例AdaBoost实战葡萄酒数据

- 需求　已知葡萄酒数据，根据数据进行葡萄酒分类

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	nflavanoid phe	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
2	1	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065
3	1	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050
4	1	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185
5	1	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480
6	1	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735
7	1	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
8	1	14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290
9	1	14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295
10	1	14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045
177	3	13.27	4.28	2.26	20	120	1.59	0.69	0.43	1.35	10.2	0.59	1.56	835
178	3	13.17	2.59	2.37	20	120	1.65	0.68	0.53	1.46	9.3	0.6	1.62	840
179	3	14.13	4.1	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2	0.61	1.6	560

- 思路分析

1 读取数据

2 特征处理

2-1 Adaboost一般做二分类 去掉一类(1,2,3)

2-2 准备特征值和目标值

2-3 类别转化 (2,3)=>(0,1)

2-4 划分数据

3 实例化单决策树 实例化Adaboost-由500颗树组成

4 单决策树模型训练和评估

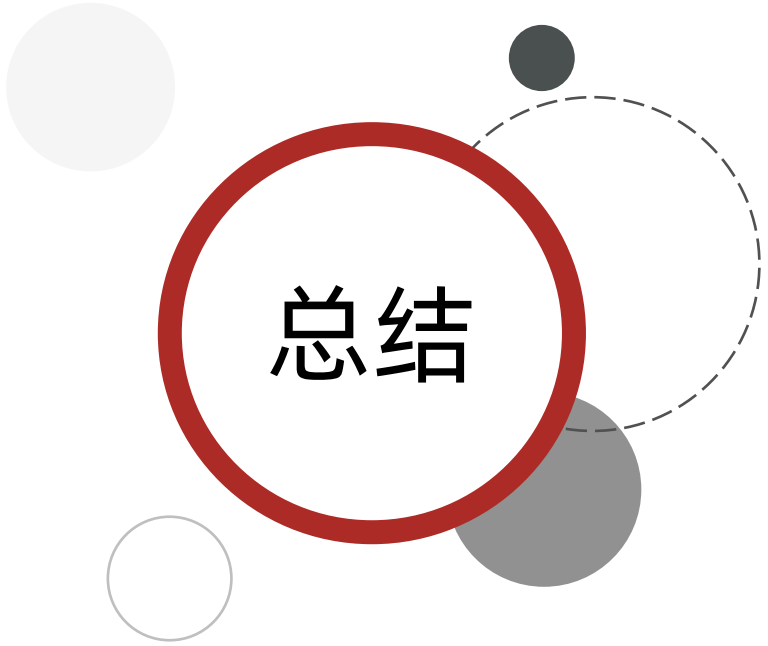
5 AdaBoost模型训练和评估

案例AdaBoost实战葡萄酒数据

```
# 1.导入依赖包
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier    # 集成学习
from sklearn.metrics import accuracy_score

def dm01_adaboost():
    # 2.读取数据
    df_wine = pd.read_csv('./data/wine0501.csv')
    # print(df_wine.info)

    # 3.特征处理
    # 3.1  Adaboost一般做二分类 去掉一类(1,2,3)
    df_wine = df_wine[df_wine['Class label'] != 1]

    # 3.2 准备特征值和目标值 Alcohol酒精含量 Hue颜色
    x = df_wine[['Alcohol', 'Hue']].values
    y = df_wine['Class label']

    # 3.3 类别转化y (2,3)=>(0,1)
    y = LabelEncoder().fit_transform(y)

    # 3.4 划分数据
    X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=22, test_size=0.2)
    print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

案例AdaBoost实战葡萄酒数据

```
# 4.实例化单决策树 实例化Adaboost-由500颗树组成
mytree = DecisionTreeClassifier(criterion='entropy', max_depth=1, random_state=0)
myada = AdaBoostClassifier(base_estimator=mytree, n_estimators=500, learning_rate=0.1, random_state=0)

# 5.单决策树模型训练和评估
mytree.fit(X_train, y_train)
myscore = mytree.score(X_test, y_test)
print('myscore-->', myscore)

# 6.AdaBoost模型训练和评估
myada.fit(X_train, y_train)
myscore = myada.score(X_test, y_test)
print('myscore-->', myscore)
```



1 Adaboost概念

- 通过逐步提高被分类错误的样本的权重来训练一个强分类器。提升的思想

2 Adaboost构建过程

- 1 初始化数据权重，来训练第1个弱学习器。找最小的错误率计算模型权重，再更新模数据权重。
- 2 根据更新的数据集权重，来训练第2个弱学习器，再找最小的错误率计算模型权重，再更新模数据权重。
- 3 依次重复第2步，训练n个弱学习器。组合起来进行预测。结果大于0为正类、结果小于0为负类

练习

1、下列关于Adaboost的说法正确的是的是？（多选）

A）Adaboost算法一般用来做二分类，特别在视觉领域应用较多

B）AdaBoost算法不能提高精度

C）AdaBoost算法API函数可以配置学习率参数，学习率参数作用于每一颗树的数据权重更新上

D）AdaBoost算法使用的树深度不要过深，否则容易过拟合

答案：AD

目录

Contents

学习目标

Learning Objectives

1. 能说出残差提升树的概念
2. 能说出残差提升树的基本构建过程
3. 能说出梯度提升树GBD的基本构建过程

提升树 （Boosting Decision Tree ）

- 思想
  - 通过拟合残差的思想来进行提升
  - 残差：真实值 - 预测值

- 生活中的例子
  - 预测某人的年龄为100岁
  - 第1次预测：对100岁预测，预测成80岁；100 - 80 = 20（残差）
  - 第2次预测：上一轮残差20岁作为目标值，预测成16岁；20 - 16 = 4 （残差）
  - 第3次预测：上一轮的残差4岁作为目标值，预测成3.2岁；4 - 3.2 = 0.8（残差）
  - 若三次预测的结果串联起来： 80 + 16 + 3.2 = 99.2

  通过拟合残差可将多个弱学习器组成一个强学习器，这就是提升树的最朴素思想

梯度提升树 （Gradient Boosting Decision Tree）

梯度提升树不再拟合残差，而是利用梯度下降的近似方法，利用损失函数的负梯度作为提升树算法中的残差近似值。

假设：

1.前一轮迭代得到的强学习器是：$f_{i-1}(x)$

2.损失函数为平方损失是：$L(y,f_{i-1}(x))$

3.本轮迭代的目标是找到一个弱学习器：$h_i(x)$

4.让本轮的损失最小化: $L(y, f_t(x))=L(y, f_{t-1}(x) + h_t(x)) = \frac{1}{2}(y_i - f(x_i))^2$

5.则要拟合的负梯度为:

$$\frac{\partial L(y, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$$

$$-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right] = y_i - f(x_i)$$

GBDT 拟合的负梯度就是残差。如果我们的 GBDT 进行的是分类问题，则损失函数变为 logloss，此时拟合的目标值就是该损失函数的负梯度值

梯度提升树 -案例

- 已知：

x	1	2	3	4	5	6	7	8	9	10
目标值	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

- 初始化弱学习器(CART树)：

当模型预测值为何值时，会使得第一个弱学习器的平方误差最小，即：求损失函数对 $f(x_i)$ 的导数，并令导数为0.

$L(y, f(x)) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2$ 求平方误差最小

$$\frac{\partial L(y, f(x))}{\partial f(x_i)} = \sum_{i=1}^{n}(y_i - f(x_i)) * (y_i - f(x_i))' = \sum_{i=1}^{n}(y_i - f(x_i)) * (-1)$$

$$= \sum_{i=1}^{n}(-y_i + f(x_i)) = 0$$

$nf(x_i) = \sum_{i=1}^{n} y_i$ 整理得： $f(x_i) = \frac{\sum_{i=1}^{n} y_i}{n}$

从公式可得：初始化树桩输出值为目标值的均值时，

可使得第一个弱学习器的平方误差最小

梯度提升树 - 例子2

2 构建第1个弱学习器，根据负梯度的计算方法得到下表：

x	1	2	3	4	5	6	7	8	9	10
目标值	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05
预测值	7.31	7.31	7.31	7.31	7.31	7.31	7.31	7.31	7.31	7.31
负梯度	-1.75	-1.61	-1.40	-0.91	-0.51	-0.26	1.59	1.39	1.69	1.74

当1.5为切分点：拟合负梯度-1.75, -1.61, -1.40, -0.91, ... , 1.74

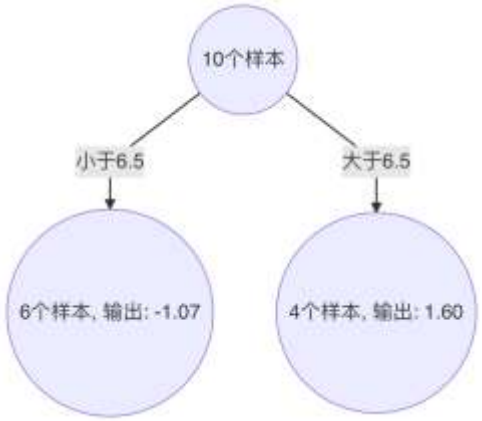
左子树：1个样本 -1.75， 右子树9个样本：-1.61，-1.40，-0.91...

右子树均值为：((-1.61) + (-1.40)+(-0.91)+(-0.51)+(-0.26)+1.59 +1.39 + 1.69 + 1.74)/9=0.19；左子树均值为：- 1.75

计算平方损失：左子树0 + 右子树：$(-1.61-0.19)^2$ + $(-1.40-0.19)^2$ + $(-0.91-0.19)^2$ + $(-0.51-0.19)^2$ +$(-0.26-0.19)^2$ + $(1.59-0.19)^2$ + $(1.39-0.19)^2$ + $(1.69-0.19)^2$ + $(1.74-0.19)^2$ =15.72308

切分点	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
平方损失	15.72	12.08	8.37	5.78	3.91	1.93	8.01	11.74	15.74

3 当 6.5 作为切分点时，平方损失最小，此时得到第1棵决策树



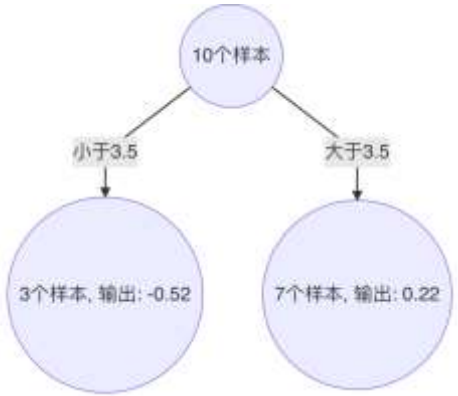
梯度提升树 - 例子3

4 构建第2个弱学习器

x	1	2	3	4	5	6	7	8	9	10
目标值	-1.75	-1.61	-1.40	-0.91	-0.51	-0.26	1.59	1.39	1.69	1.74
预测值	-1.07	-1.07	-1.07	-1.07	-1.07	-1.07	1.60	1.60	1.60	1.60
负梯度	-0.68	-0.54	-0.33	0.16	0.56	0.81	-0.01	-0.21	0.09	0.14

切分点	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
平方损失	1.42	1.00	0.79	1.13	1.66	1.93	1.93	1.9	1.91

5 以3.5 作为切分点时，平方损失最小，此时得到第2棵决策树



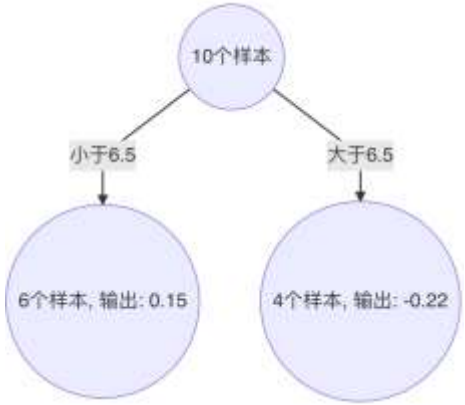
梯度提升树 - 例子4

6 构建第3个弱学习器

x	1	2	3	4	5	6	7	8	9	10
目标值	-0.68	-0.54	-0.33	0.16	0.56	0.81	-0.01	-0.21	0.09	0.14
预测值	-0.52	-0.52	-0.52	0.22	0.22	0.22	0.22	0.22	0.22	0.22
负梯度	-0.16	-0.02	0.19	-0.06	0.34	0.59	-0.23	-0.43	-0.13	-0.08

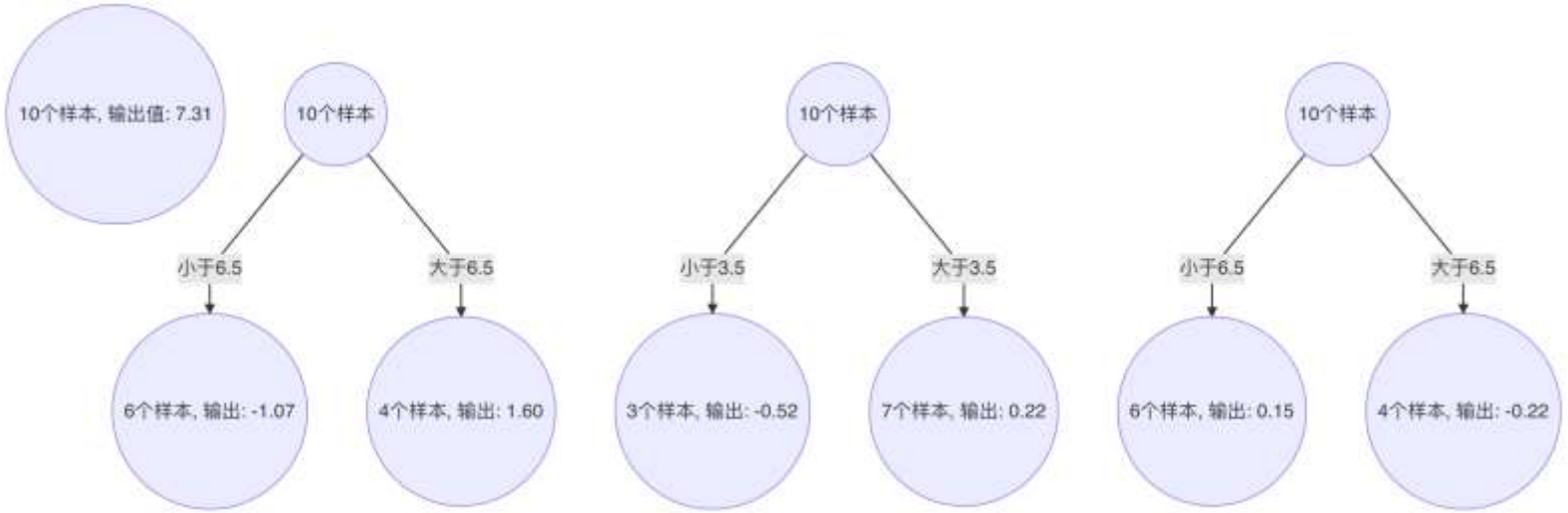
切分点	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
平方损失	0.76	0.77	0.79	0.79	0.76	0.47	0.59	0.76	0.78

7 以6.5 作为切分点时，平方损失最小，此时得到第3棵决策树



梯度提升树 - 例子5

8 构建最终弱学习器



x	1	2	3	4	5	6	7	8	9	10
目标值	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05
预测值	5.87	5.87	5.87	6.61	6.61	6.61	8.91	8.91	8.91	8.91

以x=6样本为例：输入到最终学习器中的结果： 7.31 + (-1.07) + 0.22 + 0.15 = 6.61

梯度提升树的构建流程

1 初始化弱学习器（目标值的均值作为预测值）

2 迭代构建学习器，每一个学习器拟合上一个学习器的负梯度

3 直到达到指定的学习器个数

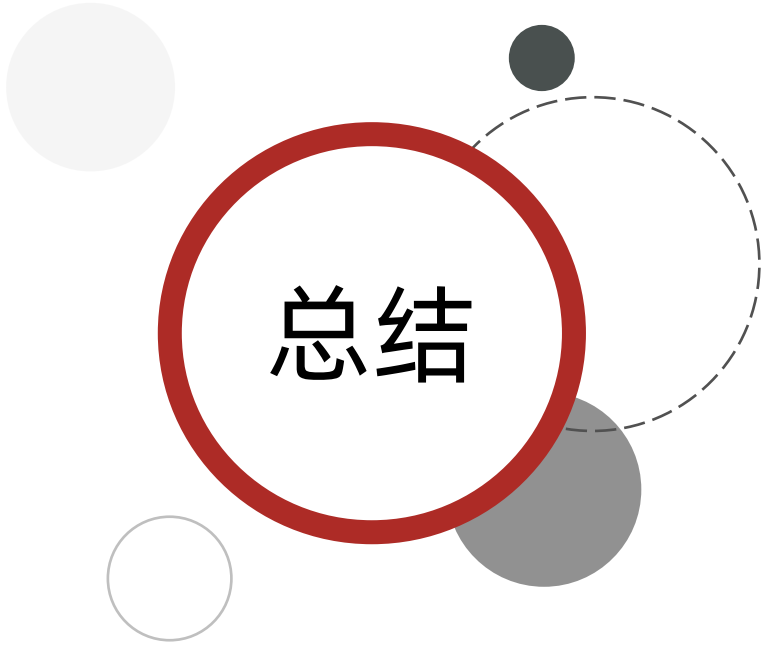
4 当输入未知样本时，将所有弱学习器的输出结果组合起来作为强学习器的输出

梯度提升树 - 案例泰坦尼克号生存预测

```
# 1.导入依赖包
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

def dm01_gbdtapi():
    # 2.读取数据
    taitan_df = pd.read_csv('./data/titanic/train.csv')
    # taitan_df.info()
    # print('taitan_df.describe()-->', taitan_df.describe())

    # 3.数据基本处理准备
    # 3.1 获取x y
    x = taitan_df[['Pclass', 'Age', 'Sex']].copy()
    y = taitan_df['Survived'].copy()
    # 3.2 缺失值处理
    x['Age'].fillna(x['Age'].mean(), inplace=True)
    # 3.3 pclass离散型数据需one-hot编码
    x = pd.get_dummies(x)
```

梯度提升树 - 案例泰坦尼克号生存预测

```
# 3.4 数据集划分
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=22, test_size=0.2)

# 4.GBDT 训练和评估
estimator = GradientBoostingClassifier()
estimator.fit(x_train, y_train)
mysorce = estimator.score(x_test, y_test)
print("gbdt mysorce-->1", mysorce)

# 5.GBDT 网格搜索交叉验证
estimator = GradientBoostingClassifier()
param = {"n_estimators": [100, 110, 120, 130], "max_depth": [2, 3, 4], "random_state": [9]}
estimator = GridSearchCV(estimator, param_grid=param, cv=3)
estimator.fit(x_train, y_train)
mysorce = estimator.score(x_test, y_test)
print("gbdt mysorce-->2", mysorce)
print(estimator.best_estimator_)
```



1 提升树？

每一个弱学习器通过拟合残差来构建强学习器

2 梯度提升树

每一个弱学习器通过拟合负梯度来构建强学习器

练习

1、下列关于GBDT的说法正确的是？（**多选**）

A）它使用的弱学习器是决策树

B）它使用了Boosting的思想

C）去拟合每次弱学习器学习后的负梯度信息

D）GBDT可以解决回归问题

答案：ABCD

目录

Contents

学习目标

Learning Objectives

1. 知道XGBoost算法的思想
2. 理解XGBoost目标函数
3. 了解XGBoost的算法API
4. 实现红酒品质预测案例

XGBoost (eXtreme Gradient Boosting)

- 极端梯度提升树，集成学习方法的王牌，在数据挖掘比赛中，大部分获胜者用了XGBoost。
- XGBoost的构建思想：

1、构建模型的方法是最小化训练数据的损失函数

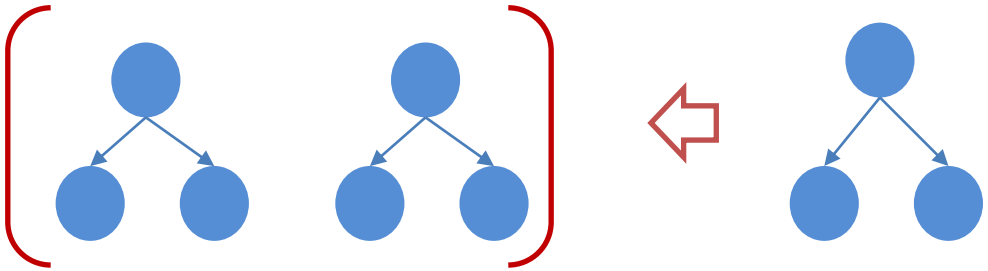
$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

训练的模型复杂度较高，易过拟合。

2、在损失函数中加入正则化项

$$\min \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \Omega(f)$$

提高对未知的测试数据的泛化性能 。



XGBoost

XGBoost（Extreme Gradient Boosting）是对GBDT的改进，并且在损失函数中加入了正则化项

$$\text{obj}(\theta) = \sum_{i}^{n} L\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right)$$

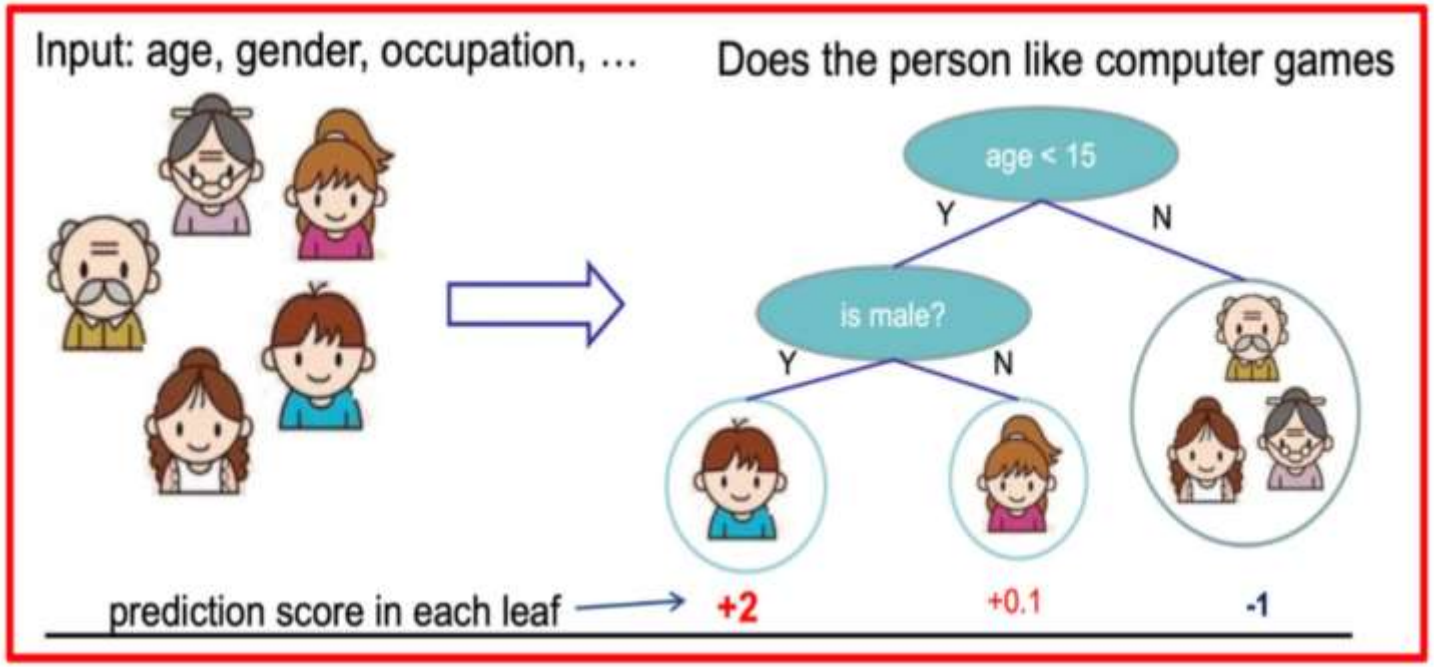
- 正则化项用来降低模型的复杂度

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

  - γT 中的 T 表示一棵树的叶子结点数量。
  - λ‖w‖²中的 w 表示叶子结点输出值组成的向量， ‖w‖ 向量的模； λ对该项的调节系数

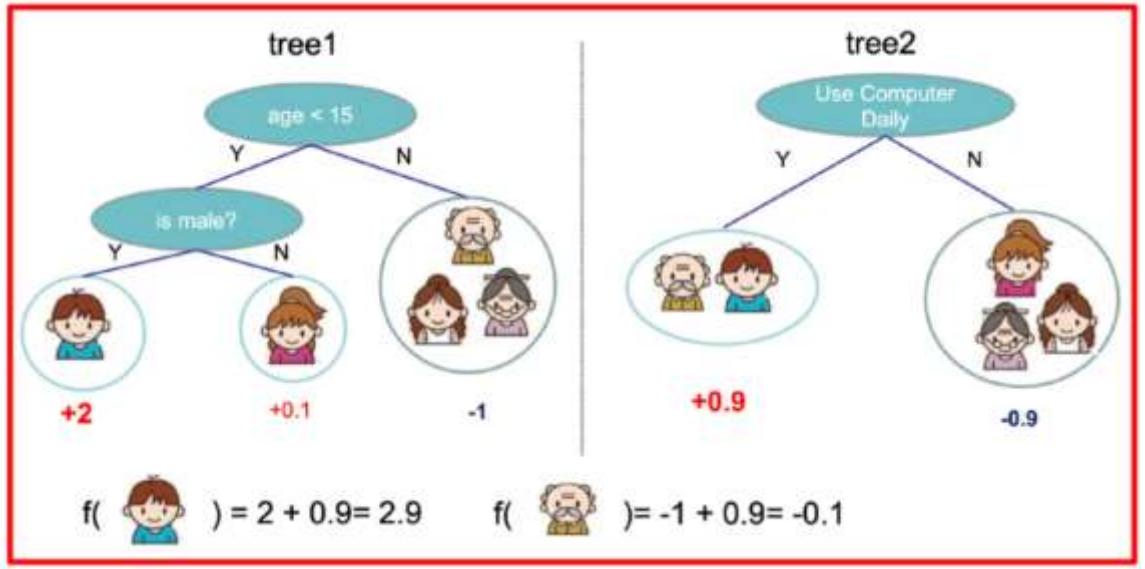
XGBoost

假设我们要预测一家人对电子游戏的喜好程度，考虑到年轻和年老相比，年轻更可能喜欢电子游戏，以及男性和女性相比，男性更喜欢电子游戏，故先根据年龄大小区分小孩和大人，然后再通过性别区分开是男是女，逐一给各人在电子游戏喜好程度上打分：

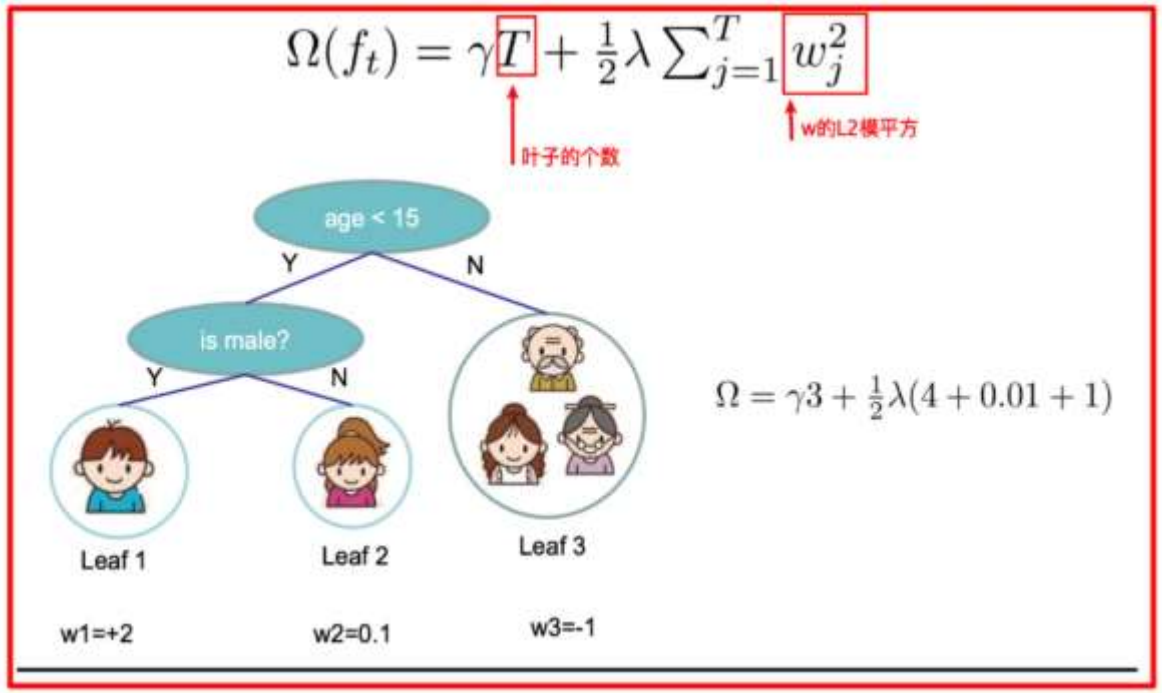


XGBoost

训练出tree1和tree2，类似之前GBDT的原理，两棵树的结论累加起来便是最终的结论



树tree1的复杂度表示为



XGBoost

进行 t 次迭代的学习模型的目标函数如下为：

$$obj^{(t)} = \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{t} \Omega\left(f_k\right)$$

$$= \sum_{i=1}^{n} L\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \sum_{k=1}^{t-1} \Omega\left(f_k\right) + \Omega\left(f_t\right)$$

直接对目标函数求解比较困难，通过泰勒展开将目标函数换一种近似的表示方式

XGBoost（复习）

- 泰勒展开

  将一个函数在某一点处展开成无限项的多项式表达式

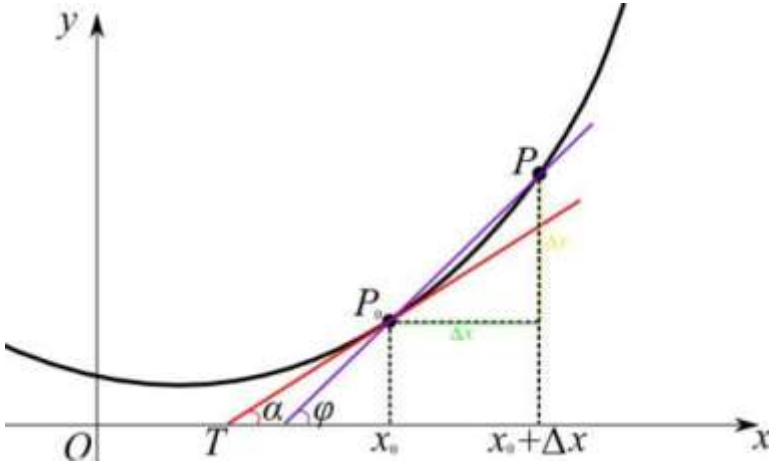
$$f(x+\Delta x)=f(x)+f^{'}(x)\cdot\Delta x+\frac{1}{2}f^{''}(x)\cdot\Delta x^2+.....+\frac{1}{n!}f^{(n)}(x)\cdot\Delta x^n$$

- 一阶泰勒展开

$$f(x+\Delta x)\approx f(x)+f^{'}(x)\cdot\Delta x$$

- 二阶泰勒展开

$$f(x+\Delta x)\approx f(x)+f^{'}(x)\cdot\Delta x+\frac{1}{2}f^{''}(x)\cdot\Delta x^2$$



XGBoost提升树 - 目标函数推导2 - 泰勒展开3

- 目标函数对 $y_i^{(t-1)}$ 进行泰勒二阶展开，得到如下近似表示的公式

.

$$obj^{(t)} \approx \sum_{i=1}^{m}\left[L\left(y_i, \hat{y}_i^{(t-1)}\right)+g_i f_t\left(x_i\right)+\frac{1}{2} h_i f_t^2\left(x_i\right)\right]+\sum_{k=1}^{t-1} \Omega\left(f_k\right)+\Omega\left(f_t\right)$$

- 其中g_i 和 h_i 的分别为损失函数的一阶导、二阶导：

$$g_i=\partial_{\hat{y}^{t-1}} L\left(y_i, \hat{y}^{(t-1)}\right)$$
$$h_i=\partial_{\hat{y}^{(t-1)}}^2 L\left(y_i, \hat{y}^{(t-1)}\right)$$

- 观察目标函数，发现以下两项表示t-1个弱学习器构成学习器的目标函数，都是常数，我们可以将其去掉：

$$obj^{(t)} \approx \sum_{i=1}^{m}[\, L(y_i, \hat{y}_i^{(t-1)})+g_i f_t(x_i)+\frac{1}{2} h_i f_t^2(x_i)\,]+\sum_{k=1}^{t-1} \Omega(f_k)+\Omega(f_t)$$

$$obj^{(t)} \approx \sum_{i=1}^{m}\left[g_i f_t\left(x_i\right)+\frac{1}{2} h_i f_t^2\left(x_i\right)\right]+\Omega\left(f_t\right)$$

$$obj^{(t)} \approx \sum_{i=1}^{m}\left[g_i f_t\left(x_i\right)+\frac{1}{2} h_i f_t^2\left(x_i\right)\right]+\gamma T+\frac{1}{2} \lambda\|w\|^2$$

XGBoost

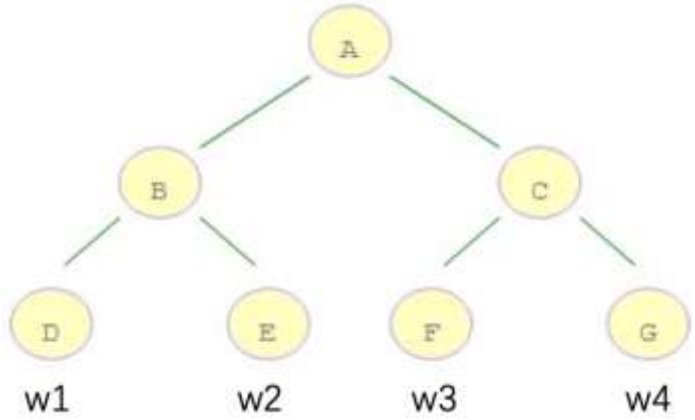
从样本角度转为按照叶子节点输出角度，优化损失函数

$$obj^{(t)} \approx \sum_{i=1}^{m}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\|w\|^2$$

上式中：

1. g_i 表示每个样本的一阶导，h_i 表示每个样本的二阶导
2. $f_t(x_i)$ 表示样本的预测值
3. T 表示叶子结点的数目
4. $\|w\|^2$ 由叶子结点值组成向量的模

举个栗子：请计算10样本在叶子结点上的输出表示



例如：m=10 个样本，落在 D 结点 3 个样本，落在 E 结点 2 个样本，落在 F 结点 2 个样本，落在 G 结点 3 个样本；计算 $g_i * f_t(x_3)$的表达形式如下：

1. D 结点计算： w1 * gi1 + w1 * gi2 + w1 * gi3 = (gi1 + gi2 + gi3) * w1
2. E 结点计算： w2 * gi4 + w2 * gi5 = (gi4 + gi5) * w2
3. F 结点计算： w3 * gi6 + w3 * gi6 = (gi6 + gi7) * w3
4. G 节点计算： w4 * gi8 + w4 * gi9 + w4 * gi10 = (gi8 + gi9 + gi10) * w4

XGBoost

目标函数中的各项可以做以下转换：

$g_i f_t(x_i)$ 表示样本的预测值，表示为：

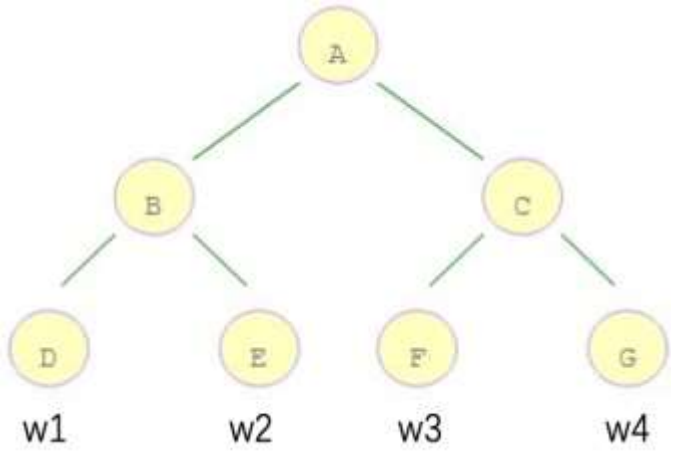
$$\sum_{i=1}^{m} g_i f_t(x_i) = \sum_{j=1}^{T} \left(\sum_{i \in I_j} g_i \right) w_j$$

$h_i f_t^2(x_i)$ 转换从叶子结点的问题，表示为：

$$\sum_{i=1}^{m} \frac{1}{2} h_i f_t^2(x_i) = \frac{1}{2} \sum_{j=1}^{T} \left(\sum_{i \in I_j} h_i \right) {w_j}^2$$

$\lambda \|w\|^2$ 由于本身就是从叶子角度来看，表示为：

$$\frac{1}{2} \lambda ||w||^2 = \frac{1}{2} \lambda \sum_{i=1}^{T} {w_i}^2$$



XGBoost提升树 - 目标函数推导3 - 转化为叶子节点输出角度3

$$
\begin{aligned}
obj^{(t)} &\approx \sum_{i=1}^{m}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\gamma T+\frac{1}{2}\lambda\|w\|^2 \\
&=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j} h_i\right)w_j^2\right]+\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T}w_j^2 \\
&=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j} h_i\right)w_j^2+\frac{1}{2}\lambda w_j^2\right]+\gamma T \\
&=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j} h_i+\lambda\right)w_j^2\right]+\gamma T
\end{aligned}
$$

令：

$$G_i=\sum_{i\in I_j} g_i$$

Gi 表示所有样本的一阶导之和

$$H_i=\sum_{i\in I_j} h_i$$

Hi 表示所有样本的二阶导之和

最终：

$$obj^{(t)}=\sum_{i=1}^{T}\left[G_i w_i+\frac{1}{2}(H_i+\lambda)w_i^2\right]+\gamma T$$

XGBoost提升树 - 目标函数推导4 - 目标函数最优解1

- 求损失函数最小值

$$obj^{(t)} = \sum_{i=1}^{T}\left[G_i w_i + \frac{1}{2}(H_i+\lambda)w_i^2\right] + \gamma T$$

对 w 求导并令其等于 0，可得到 w 的最优值

$$w_i = -\frac{G_i}{H_i+\lambda}$$

最优w，带入公式可求目标函数的最小值：

$$\begin{aligned}
obj^{(t)} &= \sum_{i=1}^{T}\left[G_i\left(-\frac{G_i}{H_i+\lambda}\right) + \frac{1}{2}(H_i+\lambda)\left(-\frac{G_i}{H_i+\lambda}\right)^2\right] + \gamma T \\
&= \sum_{i=1}^{T}\left[G_i\left(-\frac{G_i}{H_i+\lambda}\right) + \frac{1}{2}(H_i+\lambda)\left(-\frac{G_i}{H_i+\lambda}\right)^2\right] + \gamma T \\
&= \sum_{i=1}^{T}\left[-\frac{G_i^2}{H_i+\lambda} + \frac{1}{2}\left(\frac{G_i^2}{H_i+\lambda}\right)\right] + \gamma T \\
&= -\frac{1}{2}\sum_{i=1}^{T}\left(\frac{G_i^2}{H_i+\lambda}\right) + \gamma T
\end{aligned}$$

XGBoost

目标函数最终为：

$$obj^{(t)} = -\frac{1}{2}\sum_{i=1}^{T}\left(\frac{G_i^2}{H_i+\lambda}\right)+\gamma T$$

该公式也叫做打分函数 (scoring function)，从损失函数、树的复杂度两个角度来衡量一棵树的优劣。当我们构建树时，可以用来选择树的划分点，具体操作如下式所示：

$$\begin{aligned}
\text{Gain} &= Obj_{L+R} - (Obj_L + Obj_R) \\
&= \left[-\frac{1}{2}\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}+\gamma T\right] - \left[-\frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}\right)+\gamma(T+1)\right] \\
&= \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma
\end{aligned}$$

XGBoost

根据上一页PPT中计算的Gain值：

1.对树中的每个叶子结点尝试进行分裂

2.计算分裂前 - 分裂后的分数：

1. 如果Gain > 0，则分裂之后树的损失更小，会考虑此次分裂
2. 如果Gain< 0，说明分裂后的分数比分裂前的分数大，此时不建议分裂

3.当触发以下条件时停止分裂：

1. 达到最大深度
2. 叶子结点数量低于某个阈值
3. 所有的结点在分裂不能降低损失
4. 等等...

XGBoost算法API

- XGBoost的安装和使用
  - 在sklean机器学习库中没有集成XGBoost。想要使用XGBoost，需要手工安装

    pip3 install xgboost

    可以在XGBoost的官网上查看最新版本：https://xgboost.readthedocs.io/en/latest/

- XGBoost的编码风格
  - 支持非sklearn方式，也即是自己的风格
  - 支持sklearn方式，调用方式保持sklearn的形式

XGB案例：红酒品质分类

- 已知　数据集共包含 11 个特征，共计 3269 条数据. 我们通过训练模型来预测红酒的品质, 品质共有 6个类别，分别使用数字:0、 1、2、3、4、5 来表示

C6	C7	C8	C9	C10	C11	C12	C13
lorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
09	7.0	20.0	0.9959	3.2	0.56	9.6	2
08	4.0	11.0	0.9962	3.28	0.59	9.5	2
07476044189025793	13.578011493839266	21.338453384097207	0.9934903934609627	3.2247911621948413	0.7076044189025794	12.046375606719 25	5
12	3.0	7.0	0.99454	3.22	0.58	11.2	3
05823919752857635	4.88040123571178	9.23919752857644	0.9962721604942847	3.397939814643233	0.5607638887859399	10.9	1
094	10.0	45.0	0.99576	3.24	0.5	9.8	2
107	5.0	13.0	0.998	3.28	0.83	11.5	2
084	32.0	55.0	0.9988	3.34	0.75	8.7	3
08082537063469215	13.77853613002932	27.167804195043978	0.9968034737535625	3.3855707226005864	0.48335608390087 96	10.183902097521988	1
1074861280034141	6.499661658619856	20.499661658619857	0.9979791744470324	3.1549898497585955	0.7799593990343827	11.649627824481842	4
06213114582485178	21.13114582485179	52.22380433897578	0.9919603704246884	3.651432699548247	0.729807563446361	13.79615126892722	5
07877349139802776	8.248599103615758	18.430926702300937	0.9959482359982735	3.1683976290295837	0.8750280179276848	11.638674569901388	5
07527393642968154	4.607410535689055	8.21482107137811	0.9956525979106714	3.4035553678586568	0.5617776839293284	11.096294732155473	1
096	20.0	38.0	0.9978	3.4	0.53	9.5	2
06517135813542929	24.17135813542929	55.396199793491434	0.9923318050591546	3.5536171730340143	0.7152674176131643	13.505348352263285	5
06590337975549823	17.72060971474793	31.817229959249705	0.9961729463315395	3.518172299592497	0.5444121942949586	11.073108016300118	1
25	5.0	20.0	0.999	3.31	0.79	10.7	3
087	5.0	28.0	0.9988	3.14	0.6	10.2	2
122	4.0	17.0	1.0006	3.13	0.7	10.2	2
0895587871464634	4.924702971717071	14.0	0.9979075297028283	3.3131175742929266	0.5307529702828293	9.83870544575756	1
06389630677273989	18.410369322726012	42.6474076693185	0.9947291257111588	3.285318536852295	0.7841036932272601	11.900593037841444	4
07878601044077631	6.606994779611847	17.0	0.9951708355712859	3.165174869490296	0.8593005220388152	12.124896345728294	5
0679862753868205	11.696248896225196	34.645013245297655	0.9956505030905695	3.449249779245039	0.4603751103774804 7	10.731747571695431	1

- 需求：对红酒品质进行多分类
- 分析：从数据可知 1、目标是多分类　2、数据存在样本不均衡问题

案例：红酒品质分类

```
# 1.导入依赖包
import joblib
import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.model_selection import StratifiedKFold

def dm01_realdata():
    # 2.数据读取及基本数据处理
    # 2.1 加载训练集
    data = pd.read_csv('./data/红酒品质分类.csv')

    # 2.2 数据预处理
    x = data.iloc[:, :-1]
    y = data.iloc[:, -1] – 3

    # 2.3 数据集划分
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=22)

    # 2.4 数据存储
    pd.concat([x_train, y_train], axis=1).to_csv('data/红酒品质分类-train.csv')
    pd.concat([x_test, y_test], axis=1).to_csv('data/红酒品质分类-test.csv')
```

XGB案例：红酒品质分类

```
def dm02_训练模型():
    # 2.数据读取及预处理
    # 2.1 加载数据集
    train_data = pd.read_csv('./data/红酒品质分类-train.csv')
    test_data = pd.read_csv('./data/红酒品质分类-test.csv')

    # 2.2 准备数据 训练集测试集
    x_train = train_data.iloc[:, :-1]
    y_train = train_data.iloc[:, -1]
    x_test = test_data.iloc[:, :-1]
    y_test = test_data.iloc[:, -1]
    print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)

    # 3.XGBoost模型训练
    estimator = XGBClassifier(n_estimators=100, objective='multi:softmax',eval_metric='merror', eta=0.1, use_label_encoder=False, random_state=22)
    estimator.fit(x_train, y_train)

    # 4.XGBoost模型预测及评估
    y_pred = estimator.predict(x_test)
    print( classification_report(y_true=y_test, y_pred=y_pred))

    # 5.模型保存
    joblib.dump(estimator, './data/mymodelxgboost.pth')
```

XGB案例：红酒品质分类

```
# 1.导入依赖包
from sklearn.utils import class_weight

def dm03_训练模型():
    # 2.数据读取及数据预处理
    # 2.1 加载数据集
    train_data = pd.read_csv('./data/红酒品质分类-train.csv')
    test_data = pd.read_csv('./data/红酒品质分类-test.csv')
    # 2.2 准备数据 训练集测试集
    x_train = train_data.iloc[:, :-1]
    y_train = train_data.iloc[:, -1]
    x_test = test_data.iloc[:, :-1]
    y_test = test_data.iloc[:, -1]
    print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
    # 2.3 样本不均衡问题处理
    classes_weights = class_weight.compute_sample_weight(class_weight='balanced', y=y_train)

    # 3.XGBoost模型训练
    estimator = XGBClassifier(n_estimators=100, objective='multi:softmax',eval_metric='merror', eta=0.1, use_label_encoder=False, random_state=22)
    # 训练的时候，指定样本的权重
    estimator.fit(x_train, y_train, sample_weight=classes_weights)
    # 4.XGBoost模型预测及评估
    y_pred = estimator.predict(x_test)
    print(classification_report(y_true=y_test, y_pred=y_pred))
```

XGB案例：红酒品质分类

```
# 1.导入依赖包
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV

def dm04_交叉验证网格搜索():
    # 2.读取数据及数据预处理
    # 2.1 加载数据集
    train_data = pd.read_csv('./data/红酒品质分类-train.csv')
    test_data = pd.read_csv('./data/红酒品质分类-test.csv')

    # 2.2 准备数据 训练集测试集
    x_train = train_data.iloc[:, :-1]
    y_train = train_data.iloc[:, -1]
    x_test = test_data.iloc[:, :-1]
    y_test = test_data.iloc[:, -1]
    print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)

    # 2.3 交叉验证时,采用分层抽取
    spliter = StratifiedKFold(n_splits=5, shuffle=True)
```

```
    # 3.模型训练
    # 3.1 定义超参数
    param_grid = {'max_depth': np.arange(3, 5, 1),
                  'n_estimators': np.arange(50, 150, 50),
                  'eta': np.arange(0.1, 1, 0.3)}

    # 3.2 实例化XGBoost
    estimator = XGBClassifier(n_estimators=100,
                              objective='multi:softmax',
                              eval_metric='merror',
                              eta=0.1,
                              use_label_encoder=False,
                              random_state=22)

    # 3.3 实例化cv工具
    estimator = GridSearchCV(estimator=estimator, param_grid=param_grid, cv=spliter)
    # 3.4 训练模型
    estimator.fit(x_train, y_train)

    # 4.模型预测及评估
    y_pred = estimator.predict(x_test)
    print(classification_report(y_true=y_test, y_pred=y_pred))
    print('estimator.best_estimator_-->', estimator.best_estimator_)
    print('estimator.best_params_-->', estimator.best_params_)
```

总结

1 XGBoost的目标函数

$$\text{obj}(\theta) = \sum_{i}^{n} L\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right)$$

2 XGBoost的模型复杂度

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

3 XGBoost API

XGBClassifier(n_estimators, max_depth, learning_rate, objective)

练习

1、下列关于XGBoost的描述错误的是？

A）它是极限梯度提升树（Extreme Gradient Boosting）的缩写

B）它在数据挖掘方面拥有更好的性能

C）Xgbboost使用了正则化

D）xgboost算法不可以使用线性模型进行集成

答案：D

2、下列关于XGBoost损失函数的正则化项描述错误的是？

A) 它使用的是CART回归树作为弱学习器

B) 它的正则化项只包含一棵树的结果

C) 它的正则化项由树的叶子节点的个数以及L2正则化项组成

D) 模型可以通过超参数来调整正则化项对模型的惩罚力度

答案：B

练习

3、下列关于XGBoost损失函数的描述错误的是？

A）它的第T棵树的损失与第T-1棵树无关

B）在求第T棵树的结构时可将前T-1棵树的结构作为常数

C）它使用了二阶泰勒展开式去近似目标函数

D）最终得出的损失函数值越小代表模型的效果越好

答案：A

4、下列关于XGBoost树的描述错误的是？

A）它可以使用打分函数确定某个节点是否能够继续分裂

B）它可以使用打分函数确定某个特征的最佳分割点

C）最大树深度和最小叶子节点样本数可以用来调节树结构

D）超参数gamma的大小对树结构没有影响

答案：D


黑马程序员
www.itheima.com
传智教育旗下高端IT教育品牌