分类算法

连续属性离散化

- 分类数据有二元属性、标称属性等几种不同类型的离散属性
- 二元属性只有两个可能值，如“是”或“否”“对“或“错”，在分裂时，可以产生两个分支。对于二元属性，无须对其数据进行特别的处理
- 标称属性存在多个可能值，针对所使用的决策树算法的不同，标称属性的分裂存在两种方式：多路划分和二元划分
  - 对于ID3、C4.5等算法，均采取多分支划分的方法，标称属性有多少种可能的取值，就设计多少个分支
  - CART算法采用二分递归分割的方法，因此该算法生成的决策树均为二叉树
- 标称属性中有类特别的属性为序数属性， 其属性的取值是有先后顺序的。对于序数属性的分类，往往要结合实际情况来考虑

连续属性离散化

- 非监督离散化不需要使用分类属性值，相对简单，有等宽离散化、等频离散化、聚类等方法
  - 等宽离散化将属性划分为宽度一致的若干个区间
  - 等频离散化将属性划分为若干个区间，每个区间的数量相等
  - 聚类将属性间根据特性划分为不同的簇，以此形式将连续属性离散化
- 非监督离散化的方法能够完成对连续数据进行离散化的要求，但是相比之下，对连续属性监督离散化很多时候能够产生更好的结果。常用的方法是通过选取极大化区间纯度的临界值来进行划分
  - C4.5与CART算法中的连续属性离散化方法均属于监督离散化方法
  - CART 算法使用Gini系数作为区间纯度的度量标准
  - C4. 5算法使用熵作为区间纯度的度量标准

过拟合问题

- 训练误差代表分类方法对于现有训练样本集的拟合程度
- 泛化误差代表此方法的泛化能力，即对于新的样本数据的分类能力如何
- 模型的训练误差比较高，则称此分类模型欠拟合
- 模型的训练误差低但是泛化误差比较高，则称此分类模型过拟合
- 对于欠拟合问题，可以通过增加分类属性的数量、选取合适的分类属性等方法，提高模型对于训练样本的拟合程度

过拟合问题

- 对口罩销售定价进行分类
- 样本集

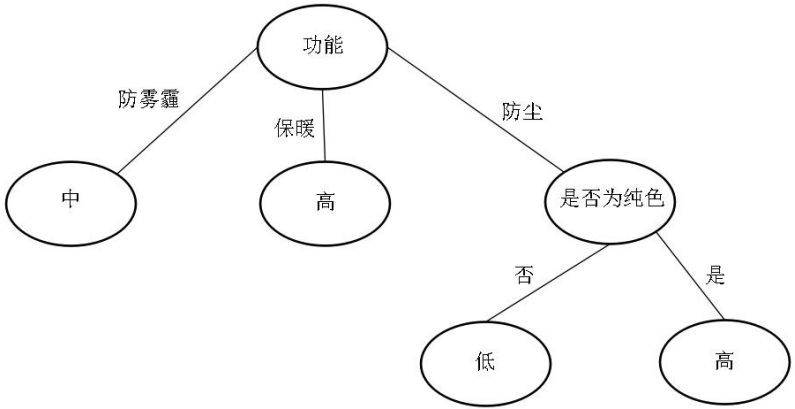
产品名	功能	是否为纯色	销售价位
加厚口罩	防尘	否	低
保暖口罩	保暖	否	高
护耳口罩	保暖	是	高
活性炭口罩	防雾霾	是	中
三层防尘口罩	防尘	否	低
艺人同款口罩	防尘	是	高
呼吸阀口罩	防雾霾	是	中

- 测试集

产品名	功能	是否为纯色	销售价位
儿童口罩	防尘	是	低
情侣口罩	保暖	否	高
一次性口罩	防尘	否	低
无纺布口罩	防尘	是	低
颗粒物防护口罩	防雾霾	否	中

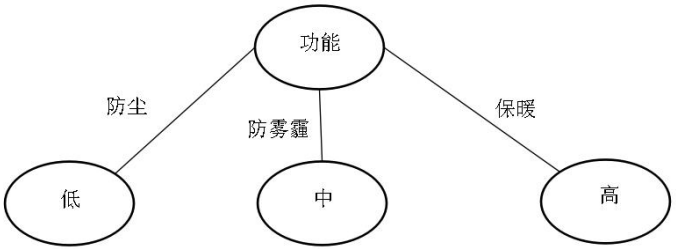
过拟合问题

- 三层决策树
- 训练误差为0，测试误差高达2/5



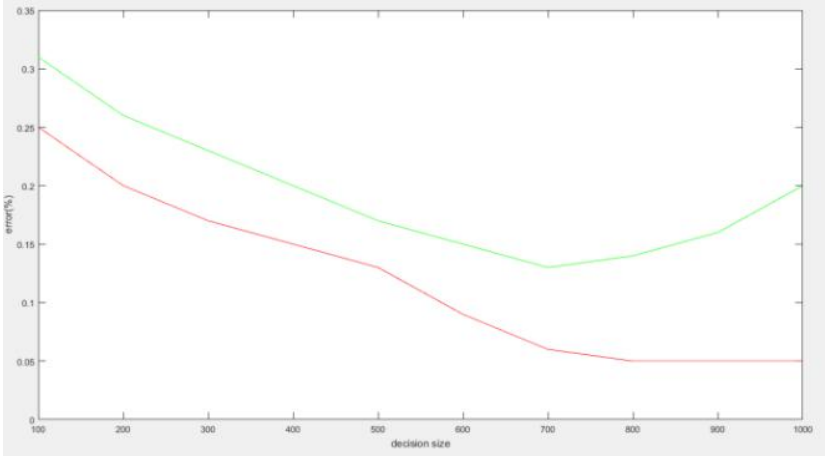
过拟合问题

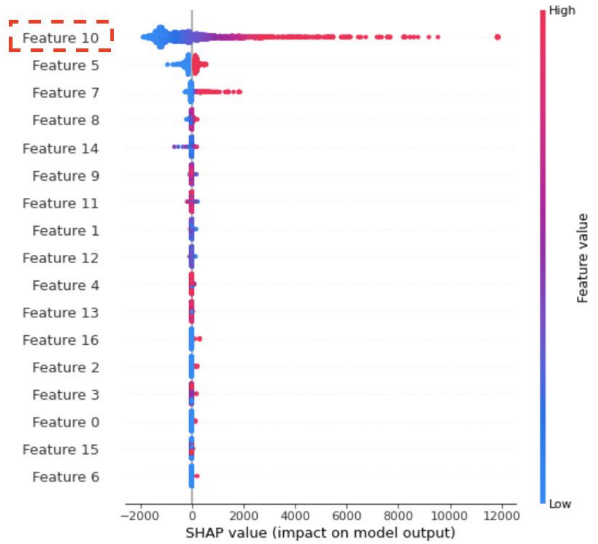
- 两层决策树
- 训练集拟合程度相比较低，但测试集表现更好



过拟合问题

- 过拟合现象会导致随着决策树的继续增长，尽管训练误差仍在下降，但是泛化误差停止下降，甚至还会提升
- 决策树误差曲线
- 防止数据泄露





过拟合问题

- 解决过拟合问题，一方面要注意数据训练集的质量，选取具有代表性样本的训练样本集。另一方面要避免决策树过度增长，通过限制树的深度来减少数据中的噪声对于决策树构建的影响，一般可以采取剪枝的方法
- 剪枝是用来缩小决策树的规模，从而降低最终算法的复杂度并提高预测准确度，包括预剪枝和后剪枝两类
- 预剪枝的思路是提前终止决策树的增长，在形成完全拟合训练样本集的决策树之前就停止树的增长，避免决策树规模过大而产生过拟合
- 后剪枝策略先让决策树完全生长，之后针对子树进行判断，用叶子结点或者子树中最常用的分支替换子树，以此方式不断改进决策树，直至无法改进为止

分类效果评价

- 对于一般分类问题，有训练误差、泛化误差、准确率、错误率等指标
- 对于常见的二分类问题，样本只有两种分类结果，将其定义为正例与反例。那么在进行分类时，对于一个样本，可能出现的分类情况共有四种:
  - 样本为正例，被分类为正例，称为真正类(TP)
  - 样本为正例，被分类为反例，称为假反类(FN)
  - 样本为反例，被分类为正例，称为假正类(FP)
  - 样本为反例，被分类为反例，称为真反类(TN)

分类效果评价

- 准确率：分类模型正确分类的样本数（包括正例与反例）与样本总数的比值

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- 精确率(precision)：模型正确分类的正例样本数与总的正例样本总数（即正确分类的正例样本数目与错误分类的正确样本数目之和）的比值

$$precision = \frac{TP}{TP + FP}$$

- 召回率(recall，也称为查全率)：模型分类正确的正例样本数与分类正确的样本总数（分类正确的正例和分类正确的反例之和）的比值

$$recall = \frac{TP}{TP + FN}$$

分类效果评价

- F值为精确率和召回率的调和平均

$$F = \frac{(\alpha^2 + 1) \times precision \times recall}{\alpha^2 precision + recall}$$

- 其中α为调和参数值，当α取值为1时，F值就是最常见的F1值

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

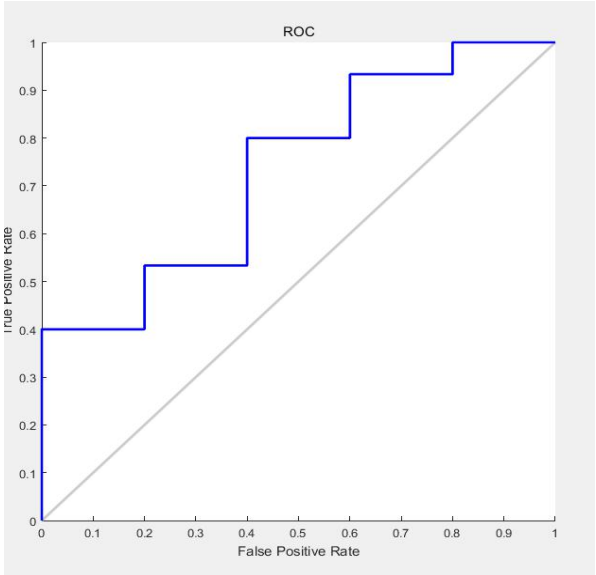
分类效果评价

- 受试者工作特征曲线 (ROC)曲线也是一种常用的综合评价指标。假设检验集中共有20个样本，每个样本为正类或反类，根据分类算法模型可以得出每个样本属于正类的概率，将样本按照此概率由高到低排列

样本编号	分类	预测为正类的概率
1	正类	0.98
2	正类	0.96
3	正类	0.92
4	正类	0.88
5	正类	0.85
6	正类	0.83
7	反类	0.82
8	正类	0.8
9	正类	0.78
10	反类	0.71
11	正类	0.68
12	正类	0.64
13	正类	0.59
14	正类	0.55
15	反类	0.52
16	正类	0.51
17	正类	0.5
18	反类	0.48
19	正类	0.42
20	反类	0.2

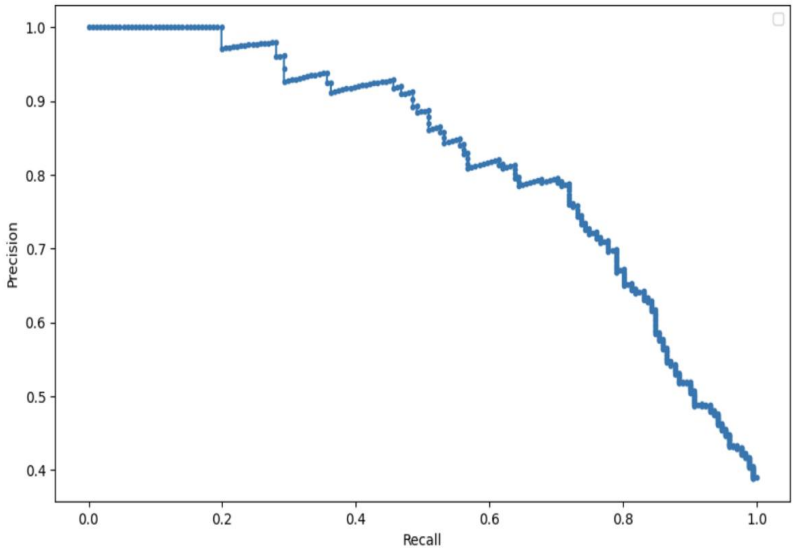
分类效果评价

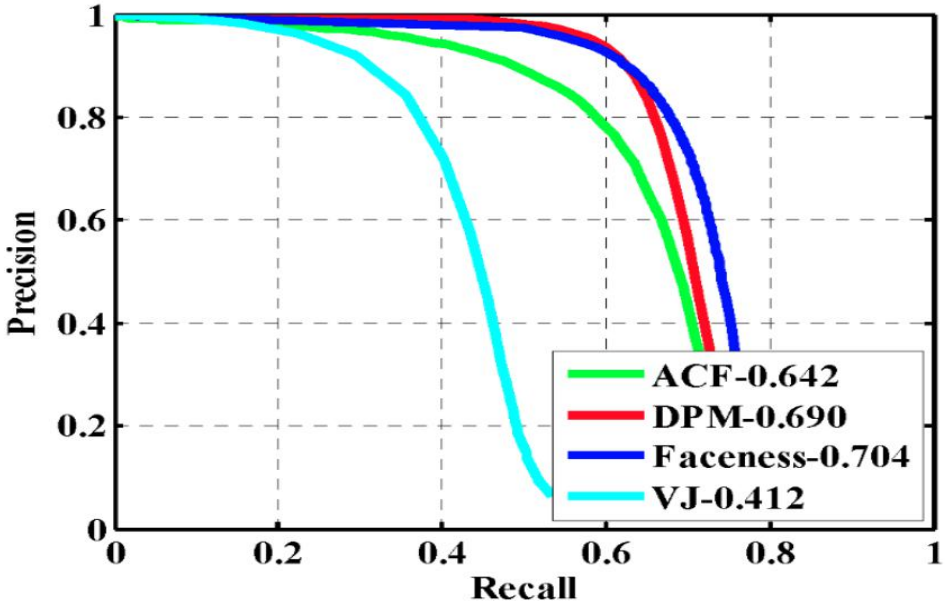
- ROC曲线下的面积称为AUC(Area under Curve)，AUC值越大，表示分类模型的预测准确性越高，ROC曲线越光滑，一般代表过拟合现象越轻



分类效果评价

Precision-Recall图



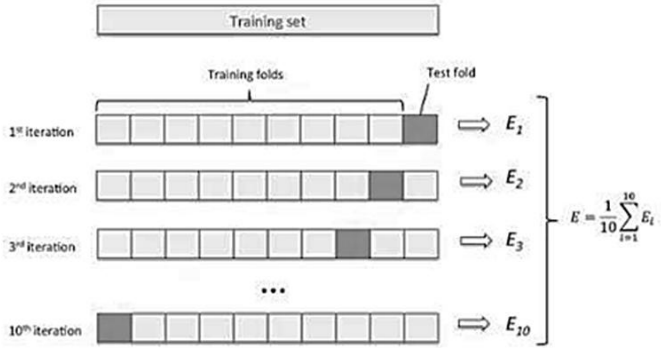


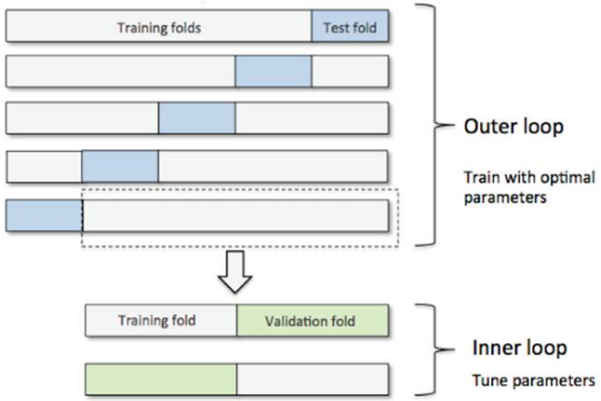
分类效果评价方法

- 保留法将样本集按照定比例划分为训练集与检验集两个集合，两个集合中样本随机分配且不重叠。对于比例的确定，一般情况下， 训练集会大于检验集，例如训练集占70%，检验集占30%，具体比例可结合实际情况进行判定
- 蒙特卡洛交叉验证，也称重复随机二次采样验证，这种验证方法随机将数据集划分为训练集与检验集，使用检验集检验训练集训练的模型效果，多次重复此过程取平均值作为模型好坏的评价标准。蒙特卡洛交叉验证法也可看作是多次进行保留法
- k折交叉验证法将样本集随机地划分为k个大小相等的子集，在每一轮交叉验证中， 选择一个子集作为检验集，其余子集作为训练集，重复k轮，保证每一个子集都作为检验集出现，用K轮检验结果取平均值作为模型好坏的评价标准。最常用的k折交叉验证法为十折交叉验证

分类效果评价方法

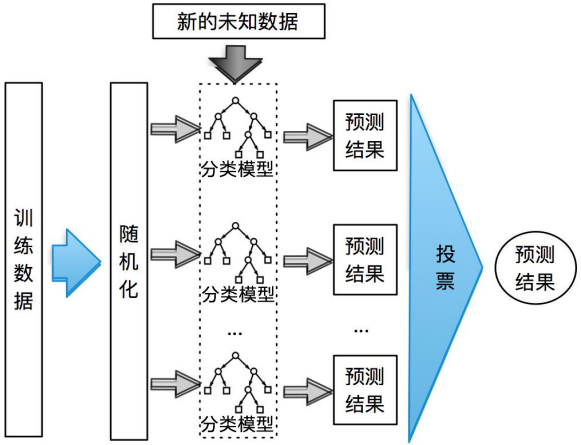
- 留一法指每次检验集中只包含一个样本的交叉验证方法
- 留p法是每次使用p个样本作为检验集的交叉验证方法
- 自助法是统计学中的一种有放回均匀抽样方法，即从一个大小为n的样本数据集S中构建一个大小为n'的训练样本集St需要进行n'次抽取，每次均可能抽取到n个样本中的任何一个。n'次抽取之后，剩余的未被抽取到的样本成为检验集

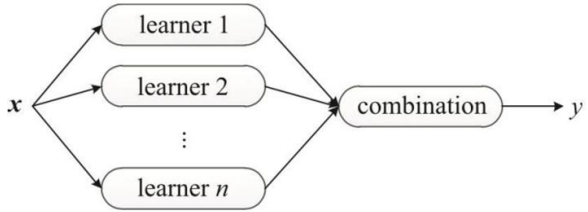




集成学习

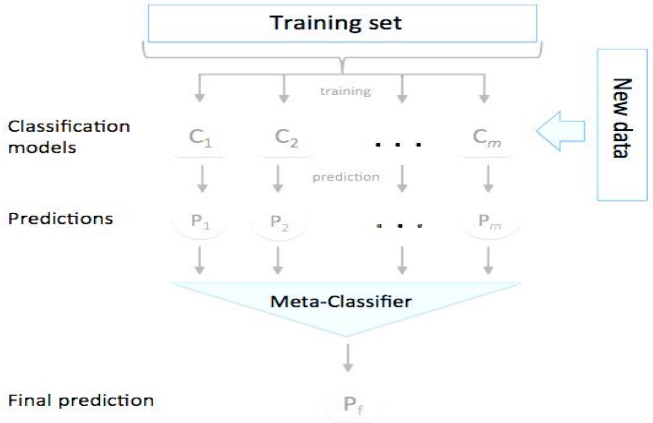
- 集成学习(Ensemble learning)是机器学习中近年来的一大热门领域。其中的集成方法是用多种学习方法的组合来获取比原方法更优的结果
- 使用于组合的算法是弱学习算法，即分类正确率仅比随机猜测略高的学习算法，但是组合之后的效果仍可能高于强学习算法，即集成之后的算法准确率和效率都很高





Stacking

- Stacking方法（知识蒸馏）是指训练一个模型用于组合其他各个模型。
- 先训练多个不同的模型，然后把训练得到的各个模型的输出作为输入来训练一个模型，以得到一个最终的输出。



装袋法

- 装袋法(Bagging)又称为Bootstrap Aggregating,其原理是通过组合多个训练集的分类结果来提升分类效果
- 装袋法由于多次采样，每个样本被选中的概率相同，因此噪声数据的影响下降，所以装袋法太容易受到过拟合的影响
- 使用sklearn库实现的决策树装袋法提升分类效果。其中X和Y分别是鸢尾花（iris）数据集中的自变量（花的特征）和因变量（花的类别）

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
#加载iris数据集
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

装袋法

```
#分类器及交叉验证
seed = 42
kfold = KFold(n_splits=10, random_state=seed)
cart = DecisionTreeClassifier(criterion='gini',max_depth=2)
cart = cart.fit(X, Y)
result = cross_val_score(cart, X, Y, cv=kfold)
print("CART树结果：",result.mean())
model=BaggingClassifier(base_estimator=cart,n_estimators=100, random_state=seed)
result = cross_val_score(model, X, Y, cv=kfold)
print("装袋法提升后结果：",result.mean())
```

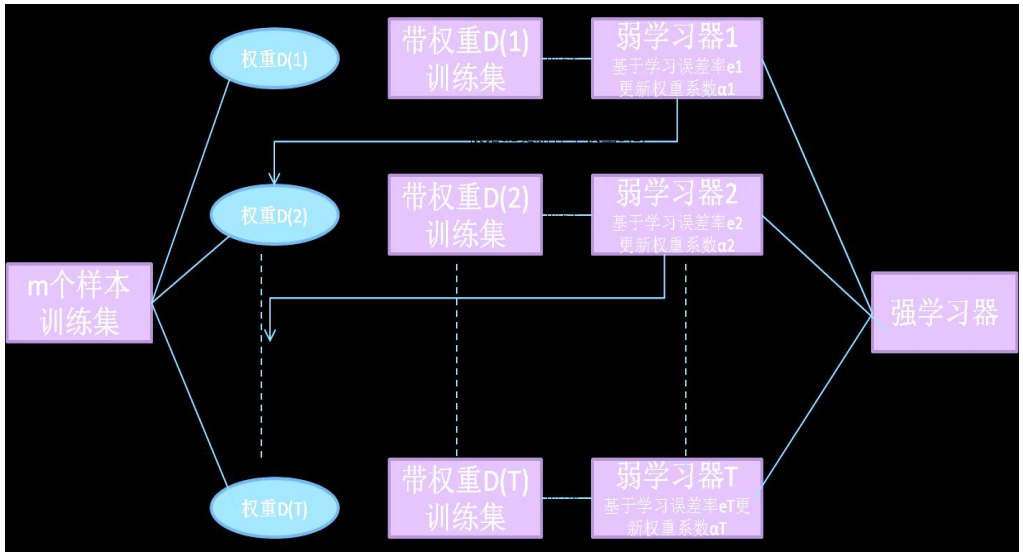
- 运行之后的结果如下
  - CART树结果： 0.933333333333
  - 装袋法提升后结果： 0.946666666667
- 可以看到装袋法对模型结果有一定提升。当然，提升程度与原模型的结构和数据质量有关。如果分类回归树的树高度设置为3或5，原算法本身的效果就会比较好，装袋法就没有提升空间

提升法

- 提升法(Boosting)与装袋法相比每次的训练样本均为同一组，并且引入了权重的概念，给每个单独的训练样本都会分配个相同的初始权重。然后进行T轮训练，每-轮中使用一个分类方法训练出一个分类模型，使用此分类模型对所有样本进行分类并更新所有样本的权重:分类正确的样本权重降低，分类错误的样本权重增加，从而达到更改样本分布的目的。
- 每一轮训练后，都会生成一个分类模型，而每次生成的这个分类模型都会更加注意在之前分类错误的样本，从而提高样本分类的准确率。对于新的样本，将T轮训练出的T个分类模型得出的预测结果加权平均，即可得出最终的预测结果。

提升法

- 如何计算学习误差率?
- 如何得到弱学习器权重系数?
- 如何更新样本权重?
- 使用何种集成策略?



AdBoost算法（1）

假设我们的训练集样本是

$$T = \{(x,y_1), (x_2, y_2), \ldots (x_m, y_m)\}$$

训练集的在第k个弱学习器的输出权重为

$$D(k) = (w_{k1}, w_{k2}, \ldots w_{km}); \;\; w_{1i} = \frac{1}{m}; \;\; i = 1, 2 \ldots m$$

首先我们看看Adaboost的分类问题。

分类问题的误差率很好理解和计算。由于多元分类是二元分类的推广，这里假设我们是二元分类问题，输出为{-1，1}，则第k个弱分类器$G_k(x)$在训练集上的加权误差率为

$$e_k = P(G_k(x_i) \neq y_i) = \sum_{i=1}^{m} w_{ki} I(G_k(x_i) \neq y_i)$$

接着我们看弱学习器权重系数,对于二元分类问题，第k个弱分类器$G_k(x)$的权重系数为

$$\alpha_k = \frac{1}{2} log \frac{1 - e_k}{e_k}$$

AdBoost算法（2）

第三个问题，更新更新样本权重D。假设第k个弱分类器的样本集权重系数为$D(k) = (w_{k1}, w_{k2}, \ldots w_{km})$，则对应的第k+1个弱分类器的样本集权重系数为

$$w_{k+1,i} = \frac{w_{ki}}{Z_K} exp(-\alpha_k y_i G_k(x_i))$$

这里Z_k是规范化因子

$$Z_k = \sum_{i=1}^{m} w_{ki} exp(-\alpha_k y_i G_k(x_i))$$

从$w_{k+1,i}$计算公式可以看出，如果第i个样本分类错误，则$y_i G_k(x_i) < 0$，导致样本的权重在第k+1个弱分类器中增大，如果分类正确，则权重在第k+1个弱分类器中减少.具体为什么采用样本权重更新公式，我们在讲Adaboost的损失函数优化时再讲。

最后一个问题是集合策略。Adaboost分类采用的是加权表决法，最终的强分类器为

$$f(x) = sign(\sum_{k=1}^{K} \alpha_k G_k(x))$$

提升法

- 基于sklearn库中的提升法分类器对决策树进行优化，提高分类准确率。Python代码如下，其中load_breast_cancer()方法加载乳腺癌数据集，自变量（细胞核的特征）和因变量（良性、恶性）分别赋给X和Y变量

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
dataset_all = datasets.load_breast_cancer()
X = dataset_all.data
Y = dataset_all.target
seed = 42
```

提升法

```
kfold = KFold(n_splits=10, random_state=seed)
dtree = DecisionTreeClassifier(criterion='gini',max_depth=3)
dtree = dtree.fit(X, Y)
result = cross_val_score(dtree, X, Y, cv=kfold)
print("决策树结果：",result.mean())
model = AdaBoostClassifier(base_estimator=dtree,
n_estimators=100,random_state=seed)
result = cross_val_score(model, X, Y, cv=kfold)
print("提升法改进结果：",result.mean())
```

- 运行之后的结果如下。
  - 决策树结果： 0.92969924812
  - 提升法改进结果： 0.970112781955
- 可以看到提升法对当前决策树分类器的分类效果改进较大

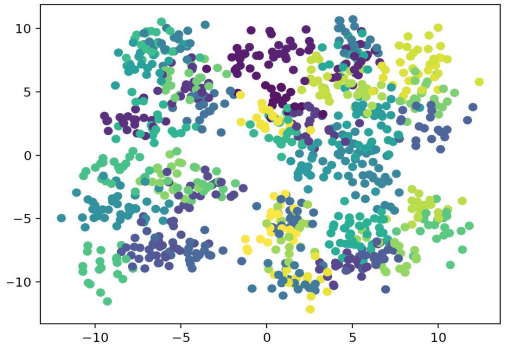
随机森林

- 随机森林是专为决策树分类器设计的集成方式，是装袋法的一种拓展。随机森林与装袋法采取相同的样本抽取方式。装袋法中的决策树每次从所有属性中选取一个最优的属性作为其分支属性，而随机森林算法每次从所有属性中随机抽取t个属性，然后从这t个属性中选取一个最优的属性作为其分支属性，这样就使得整个模型的随机性更强，从而使模型的泛化能力更强。而对于参数t的选取，决定了模型的随机性，若样本属性共有M个，t=1意味着随机选择一个属性来作为分支属性，t=属性总数时就变成了装袋法集成方式，通常t的取值为小于$\log_2(\mathrm{M}+1)$的最大整数。而随机森林算法使用的弱分类决策树通常为CART算法

随机森林

- 使用sklearn库中的随机森林算法和决策树算法进行效果对比，数据集由生成器随机生成，示例代码如下

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
X, y = make_blobs(n_samples=1000, n_features=6, centers=50,
    random_state=0)
pyplot.scatter(X[:, 0], X[:, 1], c=y)
pyplot.show()
```



GBDT

- 梯度提升决策树算法是利用梯度下降的思想，使用损失函数的负梯度在当前模型的值，作为提升树中残差的近似值，以此来拟合回归决策树。梯度提升决策树的算法过程如下：
- 初始化决策树，估计一个使损失函数最小化的常数构建一个只有根节点的树。
- 不断提升迭代：
  - 计算当前模型中损失函数的负梯度值，作为残差的估计值；
  - 估计回归树中叶子节点的区域，拟合残差的近似值；
  - 利用线性搜索估计叶子节点区域的值，使损失函数极小化；
  - 更新决策树。
- 经过若干轮的提升法迭代过程之后，输出最终的模型

GBDT

- 对于GBDT算法的具体实现，最为出色的是XGBoost树提升系统
- 下面是在Python环境下使用XGBoost模块进行回归的调用示例，首先用pandas构造一个最简单的数据集df，其中x的值为[1,2,3]，y的值为[10,20,30]，并构建训练集矩阵T_train_xbg。代码如下

```
import pandas as pd
import xgboost as xgb
df = pd.DataFrame({'x':[1,2,3], 'y':[10,20,30]})
X_train = df.drop('y',axis=1)
Y_train = df['y']
T_train_xgb = xgb.DMatrix(X_train, Y_train)
params = {"objective": "reg:linear", "booster":"gblinear"}
gbm = xgb.train(dtrain=T_train_xgb,params=params)
Y_pred = gbm.predict(xgb.DMatrix(pd.DataFrame({'x':[4,5]})))
print(Y_pred)
```

相关标准库和扩展库

- 用于数据分析、科学计算与可视化的扩展模块主要有：Numpy、SciPy、Pandas、SymPy、Matplotlib、Traits、TraitsUI、Chaco、TVTK、Mayavi、VPython、OpenCV。

- Numpy：科学计算包，支持N维数组运算、处理大型矩阵、成熟的广播函数库、矢量运算、线性代数、傅里叶变换、随机数生成，并可与C++/Fortran语言无缝结合。树莓派Python v3默认安装已经包含了Numpy。

- Matplotlib模块依赖于扩展库Numpy和标准库tkinter，可以绘制多种形式的图形，包括折线图、散点图、柱状图、饼状图、雷达图、误差线图以及二维直角坐标系图形、三维直角坐标系图形、极坐标系图形等，图形质量可满足出版要求，是数据可视化和科学计算可视化的重要工具。

- Pandas（Python Data Analysis Library）是基于Numpy的数据分析模块，提供了大量标准数据模型和高效操作大型数据集所需要的工具，可以说Pandas是使得Python能够成为高效且强大的数据分析语言的重要因素之一。