

CDA大数据分析师就业班 之 Python れ器学习

Explicit is better than implicit. Simple s better than complex. Complex is better than complicated. Flat is better than Readability counts. Special cases aren't break the rules. Although practicality beats purity. Errors should never pass silently. Unless **explicitly** silenced. In the face of mbiguity, **refuse** the temptation to guess. There should be **one** more of those! op s,191 - eapt one honking great way may not be obvious at first unless you're Dutch. Now is Namespaces are better than never. Although never is **often** better than *right* now. If the implementation is *hard* to explain, it's a **bad** may be a good idea. is easy to explain, it idea. If the implementation now. If the implementation is hard to explain, it's a bad better than never. Although never is often better than right Namespaces are way may not be obvious at first unless you're Dutch. Now is - and preferably only one - obvious way to do it. Although that ambiguity, retuse the temptation to guess. There should be one pass silently. Unless explicitly silenced. In the face of Although practicality beats purity. Errors should never break the rules. of Aguons leipage Readability counts. Special cases aren't nested. Sparse is better than dense, than complicated. Flat is better than is better than complex. Complex is better Explicit is better than implicit. Simple

Beautiful is better than ugly.

python

集成学习 Ensemble Learning

集成学习



我们之前讨论的学习器都是单一的,独立的。

整体表现比较差的学习器,在一些样本上的表现是否有可能会超过"最好"的学习器。

集成学习



当做重要决定时,大家可能都会考虑吸取多个专家而不只是一个人的意见。集成学习也是如此。

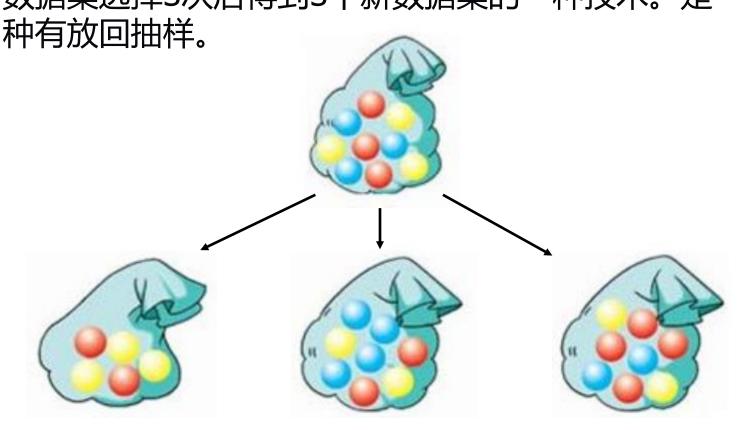
集成学习就是组合多个学习器,最后可以得到一个更好的学习器。

个体学习器之间不存在强依赖关系,装袋(bagging) 个体学习器之间存在强依赖关系,提升(boosting) 随机森林(Random Forest) Stacking



直觉:数据量越大,学习器性能越好。

bagging也叫做bootstrap aggregating,是在原始数据集选择S次后得到S个新数据集的一种技术。是一



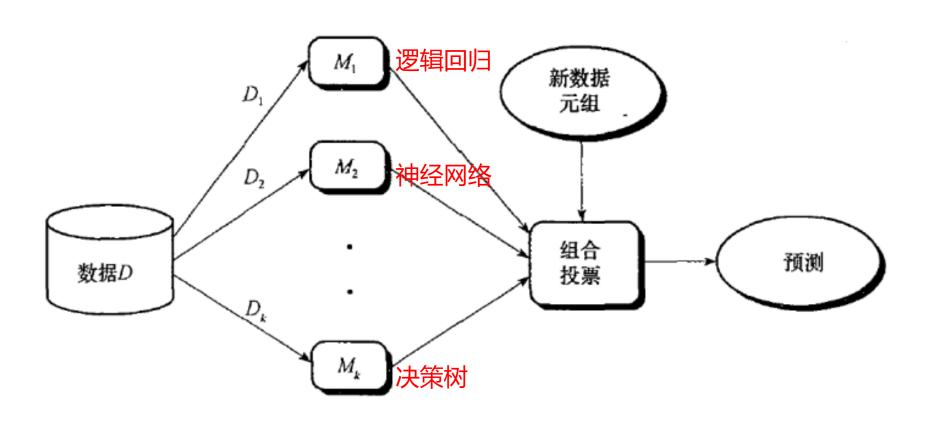


原始训练数据集

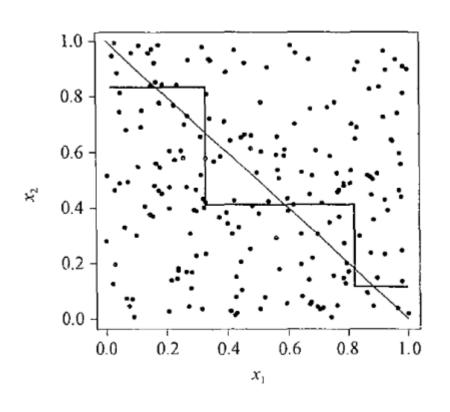
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

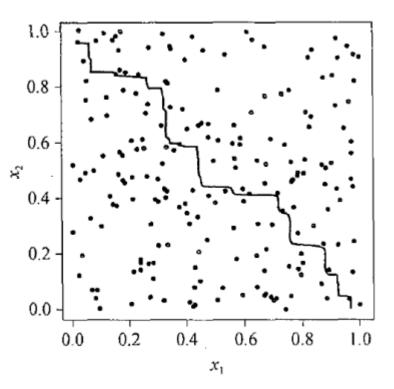
Bootstrap采样











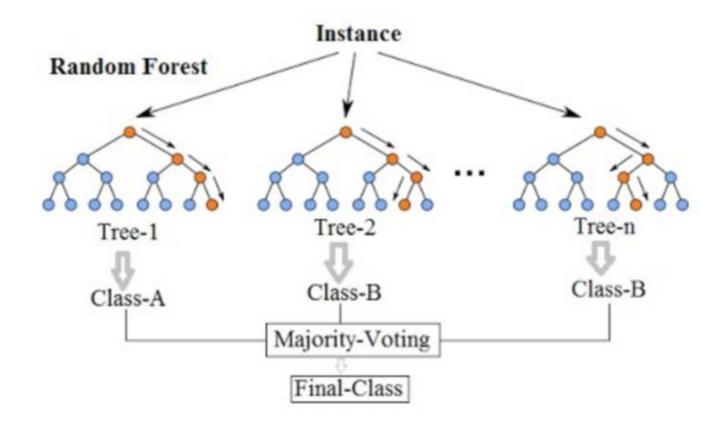


Talk is cheap Show me the

随机森林(Random Forest)



RF = 决策树+Bagging+随机属性选择



RF算法流程



- 1.样本的随机:从样本集中用bagging的方式,随机选择n个样本。
- 2.特征的随机:从所有属性d中随机选择k个属性(k<d), 然后从k个属性中选择最佳分割属性作为节点建立 CART决策树。
- 3.重复以上两个步骤m次,建立m棵CART决策树。 4.这m棵CART决策树形成随机森林,通过投票表决结果,决定数据属于哪一类。



Talk is cheap Show me the



AdaBoost是英文"Adaptive Boosting"(自适应增强)的缩写,它的自适应在于:前一个基本分类器被错误分类的样本的权值会增大,而正确分类的样本的权值会减小,并再次用来训练下一个基本分类器。同时,在每一轮迭代中,加入一个新的弱分类器,直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数才确定最终的强分类器。



直觉:将学习器的重点放在"容易"出错的样

本上。可以提升学习器的性能。

原始训练数据集

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|--|
|---|---|---|---|---|---|---|---|---|---|--|

Boosting采样

Iteration 1: 7 2 6 7 5 4 8 8 1 0

Iteration 2: 1 3 8 4 3 5 4 0 1 4

Iteration 3: 4 9 4 2 4 4 3 0 1 4



Adaboost算法可以简述为三个步骤:

- (1) 首先,是初始化训练数据的权值分布 D_1 。假设有N个训练样本数据,则每一个训练样本最开始时,都被赋予相同的权值: W_1 =1/N。
- (2)然后,训练弱分类器h, 具体训练过程中是:如果某个训练样本点,被弱分类器h, 准确地分类,那么在构造下一个训练集中,它对应的权值要减小;相反,如果某个训练样本点被错误分类,那么它的权值就应该增大。权值更新过的样本集被用于训练下一个分类器,整个训练过程如此迭代地进行下去。



(3)最后,将各个训练得到的弱分类器组合成一个强分类器。各个弱分类器的训练过程结束后,加大分类误差率小的弱分类器的权重,使其在最终的分类函数中起着较大的决定作用,而降低分类误差率大的弱分类器的权重,使其在最终的分类函数中起着较小的决定作用。

换而言之,误差率低的弱分类器在最终分类器中占的权重较大,否则较小。

Adaboost训练过程



算法: Adaboost.一种提升算法——创建分类器的组合。每个给出一个加权投票。

输入:

- · D: 类标记的训练元组集。
- · k: 轮数(每轮产生一个分类器)。
- 一种分类学习方案。

输出:一个复合模型。

方法:

- (1) 将D中每个元组的权重初始化为1/d;
- (2) for i = 1 to k do

// 对于每一轮

- (3) 根据元组的权重从D中有放回抽样,得到D;
- (4) 使用训练集D,导出模型M;
- (5) 计算M_i的错误率error(M_i)(8.34式)
- (6) iferror(M_i) > 0.5 then
- (7) 转步骤(3)重试;
- (8) endif
- (9) forD,的每个被正确分类的元组do
- (10) 元组的权重乘以error(M,)/(1-error(M,)); // 更新权重
- (11) 规范化每个元组的权重;
- (12) endfor

Adaboost判断过程



使用组合分类器对元组x分类:

(1) 将每个类的权重初始化为0;

(2)
$$fori = 1 to k do$$

(3)
$$w_i = \log \frac{1 - error(M_i)}{error(M_i)};$$

- $(4) c = M_i(\mathbf{x});$
- (5) 将w_i加到类c的权重;
- (6) endfor
- (7) 返回具有最大权重的类;

// 对于每个分类器

// 分类器的投票权重

// 从M,得到X的类预测

Adaboost算法流程2



```
输入: 训练集 D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_m, y_m)\};
           基学习算法 £; 可以是各种机器学习算法
           训练轮数 T. T轮可以得到T个基学习器
过程:
1: \mathcal{D}_1(\boldsymbol{x}) = 1/m. 训练集刚开始是均匀分布,初始化权值都是1/m
2: for t = 1, 2, ..., T do
         h_t = \mathfrak{L}(D, \mathcal{D}_t); 机器学习算法在D训练集基于D<sub>t</sub>分布下训练得到h_t基学习器
3:
      \epsilon_t = P_{m{x} \sim \mathcal{D}_t}(h_t(m{x}) 
eq f(m{x}));计算h_t基学习器在训练集D_t分布下的错误率\epsilon_t
        if \epsilon_t > 0.5 then break
        lpha_t = rac{1}{2} \ln \left( rac{1-\epsilon_t}{\epsilon_t} 
ight); 由错误率得到一个权重lpha_t ,错误率越低,权值越大,lpha_t>=0
6:
        \mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} 	imes \left\{ egin{array}{l} \exp(-lpha_t), \leq 1 	ext{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \, eta 正确,权重变小 \exp(lpha_t), \geq 1 	ext{if } h_t(\boldsymbol{x}) 
eq f(\boldsymbol{x}) \, eta 类错误,权重变大
                        =rac{\mathcal{D}_t(oldsymbol{x})	ext{exp}(-lpha_tf(oldsymbol{x})h_t(oldsymbol{x}))}{Z_t}
8: end for
输出: H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)
```

Adaboost

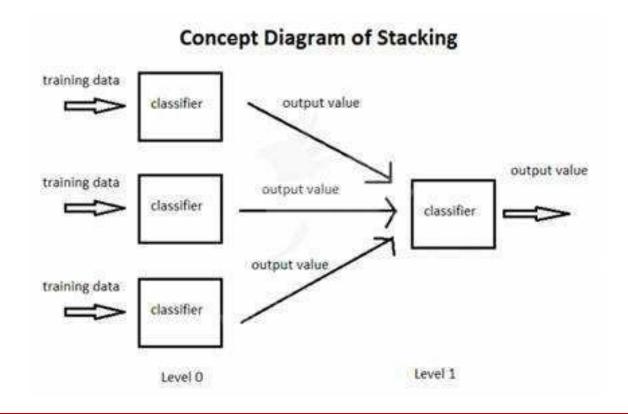


Talk is cheap Show me the

Stacking

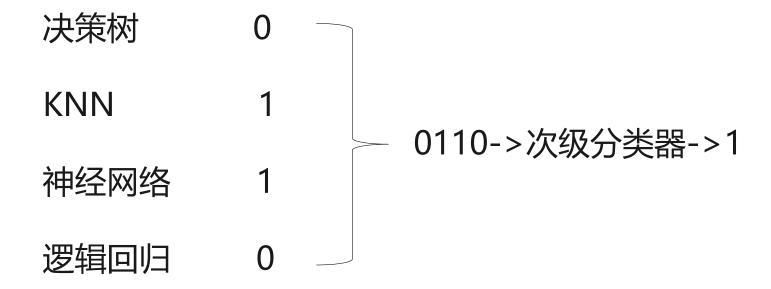


使用多个不同的分类器对训练集进预测,把预测得到的结果作为一个次级分类器的输入。次级分类器的输出是整个模型的预测结果。



Stacking





stacking



Talk is cheap Show me the

总结

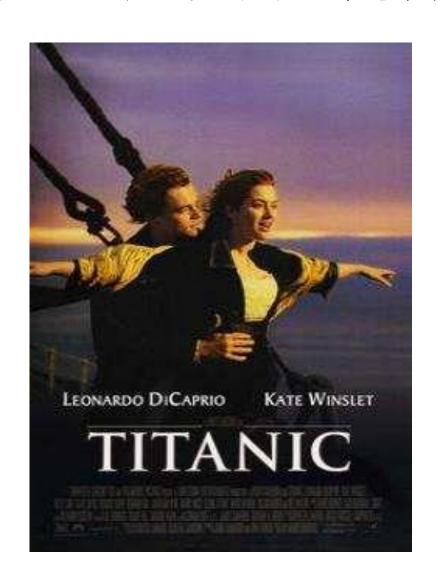


"人多力量大"

好的集成学习:多样性+准确性

| *** | 测试例1 | 测试例2 | 测试例3 | Ŋ | 则试例1 | 测试例2 | 测试例3 | 沙 | 间试例1 | 测试例2 | 测试例3 | |
|-------|--------------|--------------|--------------|-------|--------------|--------------|------|-------|------------|--------------|--------------|--|
| h_1 | √ | √ | × | h_1 | √ | √ | × | h_1 | √ | × | × | |
| h_2 | × × | \checkmark | \checkmark | h_2 | \checkmark | \checkmark | × | h_2 | × | \checkmark | × | |
| h_3 | \checkmark | × | \checkmark | h_3 | \checkmark | \checkmark | × | h_3 | × | × | \checkmark | |
| 集点 | ₹ √ | √ | √ | 集成 | √ | · √ | × | 集成 | × | × | × | |
| | (a) 集成提升性能 | | | | (b) 集成不起作用 | | | | (c) 集成起负作用 | | | |

泰坦尼克号船员获救预测项目



泰坦尼克号船员获救预测



Passengerld: 乘客编号

Survived : 存活情况(存活:1;死亡:0)

Pclass : 客场等级

Name : 乘客姓名

Sex : 性别

Age : 年龄

SibSp : 同乘的兄弟姐妹/配偶数

Parch: 同乘的父母/小孩数

Ticket : 船票编号

Fare : 船票价格

Cabin :客舱号

Embarked : 登船港口