# Bagging&&AdaBoost

集成学习分两种：

1. 利用相同的训练数据同时搭建多个独立的分类模型，然后通过投票的方式，以少数服从多数的原则做出最终的分类决策。今天学习的随机森林就是这种方式，即在相同训练数据上同时搭建多颗决策树。

在决策树中学到过一颗标准的决策树是根据每维特征对预测结果的影响程度进行排序，进而决定不同特征从上到下构建分裂节点的顺序；如果这里还按照这种方式随机森林会因为这一策略影响而构建的所有树都一致，从而丧失了多样性。因此随机森林在构建的过程中，每一颗决策树都会放弃这一个固定的算法，转而随机选取特征。

（2）按一定的次序搭建多个分类模型。这些模型之间彼此存在依赖关系。一般后一个模型的加入都需要对现有的集成模型有一定贡献，进而不断提高更新过后的集成模型性能，并借助多个弱分类器搭建出强分类器。代表行的有Bossting（AdaBoost）算法。该算法与第一种的随机森林主要区别在于每一颗决策树在生成的过程中都会尽可能降低模型在训练集上的拟合或训练误差。

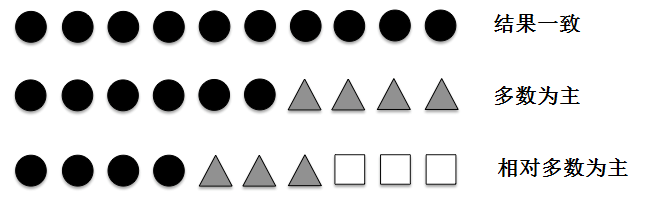
# 1.集成学习

俗话说：“三个臭皮匠赛过诸葛亮”。

当使用某一种分类器不能使我们达到很好的效果的时候，我们不妨设想将这些分类效果不好的分类器组合一下，再去看看效果是否有提升，这就是集成学习的思想。

集成算法（Ensemble Learning）思想：通过构建并结合多个学习器来完成学习任务，有时候我们也叫作“多分类器系统”。如下图C1，C2都是某一种个体的分类器，我们采用某种策略将他们组合起来。

我们有10个分类器，多数表决如下图，其中三角形、正方形或圆分别代表一个类别。



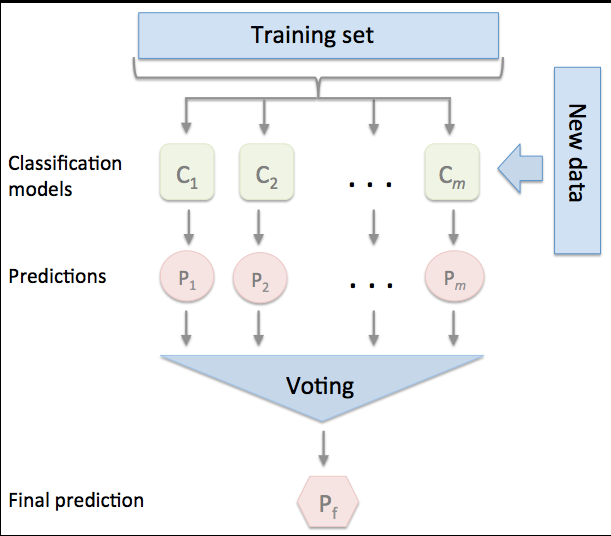
个体的分类器由一个现有的学习算法从训练数据产生，例如：C4.5决策树算法，BP神经网络算法等。我们一般把个体分类器全部为同种的分类器称为“同质的”，如全部为决策树模型。同质的集成学习中的个体学习器称为“基学习器（base learner）”，相应的算法称为基学习算法。反之，集成中包含不同种的学习器，我们称之为“异质”的，异质集成中的个体学习器包含不同的学习算法组成的，这种情况下的学习器称为“组合学习器”。相信这些名词大家能够了解。



图1： 集成学习的系统示意图

集成学习通过将多个学习器组合，常获得比单一学习器显著优越的泛化性能。这对弱学习器特别明显，这里的弱学习器我们一般会使用决策树，BP神经网络和逻辑斯特回归，有时候SVM也可以作为个体学习器。

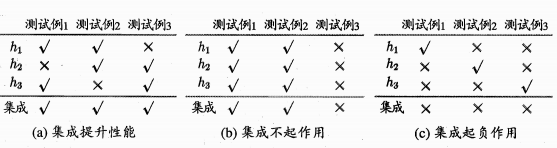
下图是使用多数投票法通用集成方法的概念：



# 例子引入

一般经验中如果把好坏不等的东西掺到一起，通常结果会比最坏的好一些，比最好的坏一些。集成学习把多个学习器结合起来，如何获得比最好的单一学习器更好的性能呢  
？

考虑一个例子：二分类问题中，假定三个分类器在三个测试样本上表现，如下图所示。打对勾的表示正确分类，打叉号的表示分类错误。集成学习的结果通过投票法voting产生。即少数服从多数。第一个图中每个分类器有66.6%的精度，但集成学习却达到了100%。第二个图中三个分类器没有差别，但是集成之后性能却没有什么提高。第三幅图中每个分类器的精度都只有33.3%，集成学习的结果更糟糕。



这个例子我们可以总结出：要获得好的集成，个体学习器应有一定的**“准确性”**，即学习器不能太坏，并且要有“多样性”，即学习器之间具有**差异**。

# Bootstrap sampling自助采样

之前的课程已经讲过模型的评估方法中有留一法（将数据集划分为两个互不相交的集合，一个做测试集，一个做训练集）和交叉验证方法（将数据分成k个大小相似互不相交的子集，每次使用k-1个子集做训练集，剩下的一个子集做测试集，以此循环进行k次的训练和测试，最后返回k次测试结果的均值。）。但是上述两种方法中都保留了一部分样本用于测试，所以实际模型所使用的训练集比源数据都要小，因此就会引入一些因训练样本规模不同而导致的估计偏差。另外一方面留一法受训练样本影响较小，但是计算复杂度又太高。因此为了解决减少训练样本规模不同造成的影响，同时还能比较高效地进行测试集的评估。自助法就是很好的解决方案。

boostrap抽样

在样本集D（样本数为m）内有放回的抽样，抽取数为m，每次抽取的概率相等为1/m，可能重复抽取。我们做一个简单的估计，样本m次采样中适中不被采样的概率为



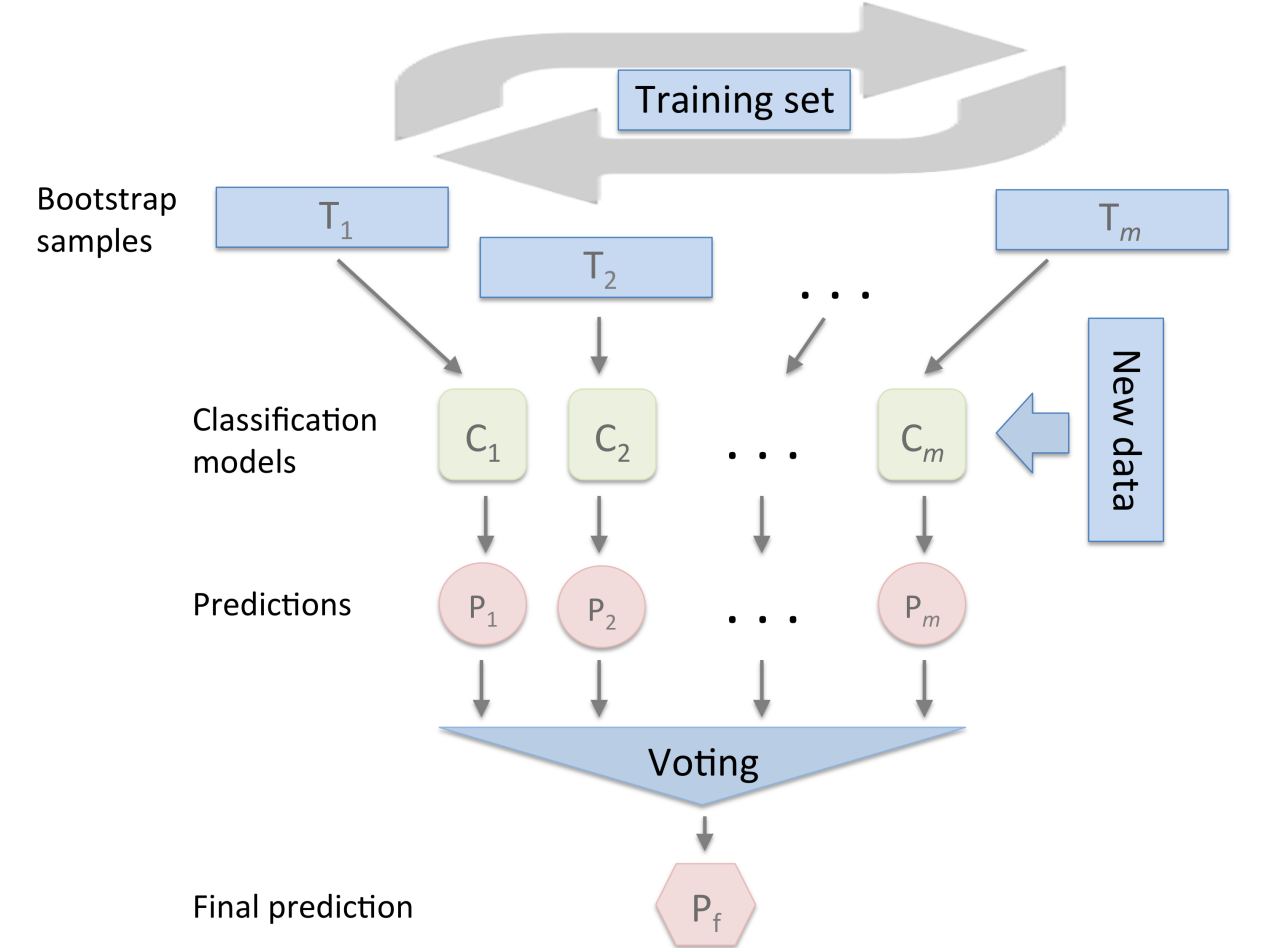
取极限得到原数据集D中36.8%的样本没有出现在采样数据集D1中。我们可以使用D1作为训练集，D-D1作为测试集。这样实际评估的模型与期望的模型都使用m个训练样本，而我们仍有数据总量的1/3的，没有在训练集中出现的样本用于测试。术语“包外估计”可以解释上述过程。



0.632=63.2%被抽样

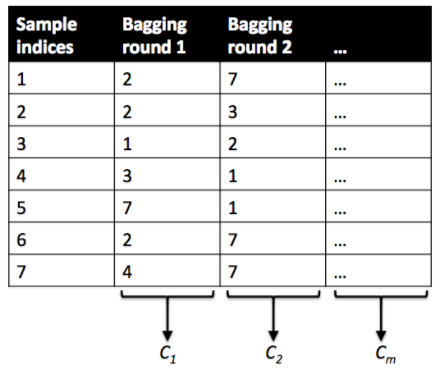
# Bagging算法

此算法没有使用相同的训练集拟合集成分类器中的单个分类器，而是采用了Bootstrap有放回抽样选取训练数据集。



下图有7个不同的训练样例，每一轮的bagging循环中，样本数据均可放回随机抽样，每个bootstrip抽样都是被用于分类器C的训练，这就是一颗典型的未剪枝的决策树。

随机森林是bagging的特例，它使用了随机特征子集去拟合单颗决策树。



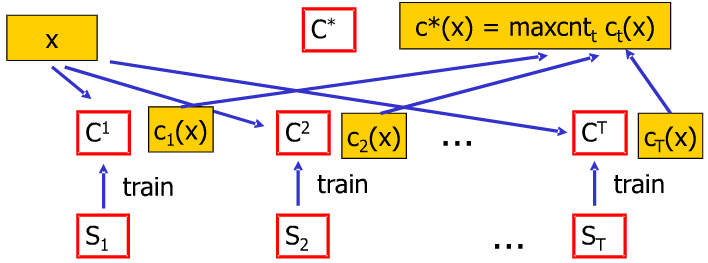
## 4.1算法流程



Bagging基本流程：通过上述自助采样，采出T个含m个训练样本的采样集，然后基于每个采样集训练出一个基学习器，在将这些基学习器进行组合。

在对预测输出进行结合的时候，Bagging通常对分类任务使用简单投票法，对回归任务进行简单的平均法。但是如果投票个数一致，则最简单的做法是随机选择一个类别，当然也可以进一步考察学习器投票的置信度来确定最终的分类。

## 4.2算法图解分析



基本分类器可以是决策树，逻辑回归等基分类器。

对于稳定性不好的分类器很实用，通过多数投票，减小了泛化误差，而对于稳定的分类器，集成效果并不明显。

## 4.3 Bagging性能

（1）Bagging是一个很高效的集成学习算法

（2）Bagging与下面讲的AdaBoost只适用于二分类不同，它能不经修改地用于多分类、回归任务。

（3）自助bootstrap采样过程还给Bagging带来了另一个优点：由于每个基学习器只使用了初始训练集中约63.2%的样本，剩下的约36.8%样本可用作验证集来泛化性能进行“包外样本评估（即：不同于训练数据的样本）”。

（4）从偏差-方差分解角度看，Bagging主要关注降低方差，因此他在不剪枝决策树、神经网络等易受样本扰动的学习器上效果更为明显。

## 4.4Bagging算法总结

Bagging算法首先采用M轮自助采样法，获得M个包含N个训练样本的采样集。然后，基于这些采样集训练出一个基学习器。最后将这M个基学习器进行组合。组合策略为：

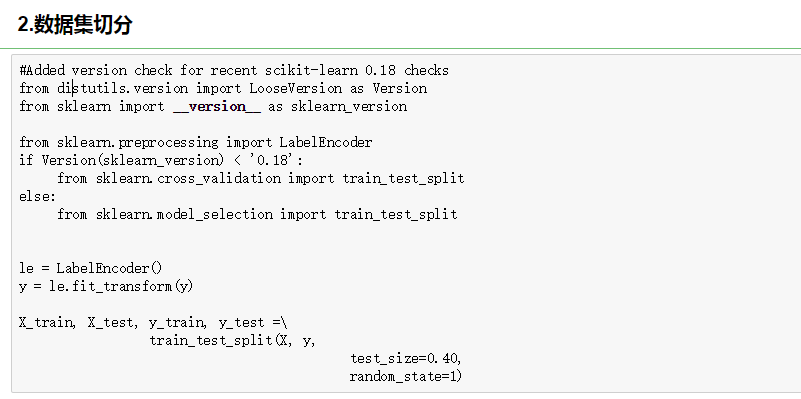
1. 分类任务采用简单投票法：即每个基学习器一票
2. 回归问题使用简单平均法：即每个基学习器的预测值取平均值。

# 4.4Bagging算法实战

## 4.1导包和加载数据



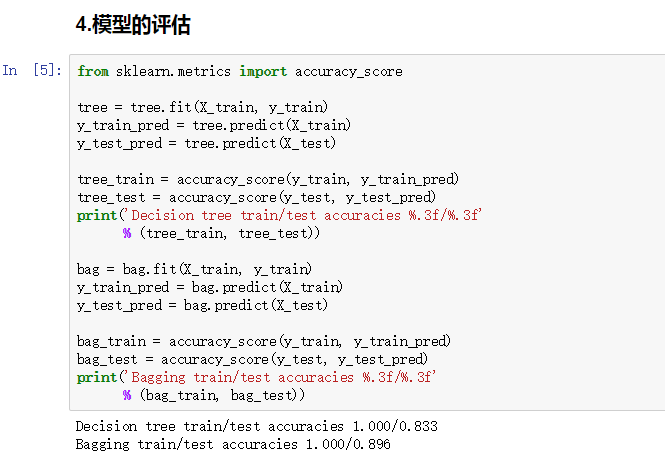
## 4.2数据集切分



## 4.3Bagging分类器初始化



## 4.4模型的评估



总结：对于单颗决策树很容易产生过拟合，Bagging就可以有它的优势。Bagging是一种降低模型方差的一种有效方法，但是对于降低模型得偏差效果不大，这也是我们选择未剪枝决策树等低偏差分类器作为集成算法成员分类器的原因。

# 5.Boosting

## 5.1算法机制

Boosting是一族可将**弱学习器升为强学习器**算法。这类算法的工作机制类似：

1. 先从初始训练集训练出一个基学习器
2. 在根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续得到最大的关注。
3. 然后基于调整后的样本分布来训练下一个基学习器；

（3）如此重复进行，直至基学习器数目达到实现指定的值T为止。

（4 ）再将这T个基学习器进行加权结合得到集成学习器。

Boosting算法的著名代表就是Adaboost算法。

因此，对于Boosting算法，存在两个问题：

1. 在每一轮中如何调整训练集，使训练的弱分类器得以进行；**（调整样本权值）**

2. 如何将各个弱分类器联合起来形成强分类器。**（调整模型权值）**

## 5.2提升的概念强化

提升是一个机器学习技术，可以用于回归和分类问题，它每一步产生一个弱预测模型（如决策树）并加权累加到总模型中；如果每一步的弱预测模型生成都是依据损失函数的梯度方向，则称之为梯度提升（GradientBoosting）。

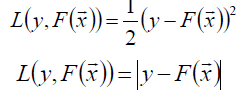
梯度提升算法首先给定一个目标损失函数，它的定义域是所有可行的弱函数集合（基函数）；提升算法通过迭代的选择一个负梯度方向上的基函数来逐渐逼近局部极小值。这种在函数域的梯度提升观点对机器学习有很大影响。

提升的理论意义：如果一个问题存在弱分类器，则可以通过提升的办法得到强分类器。

**提升算法框架：**

给定输入向量(x1,y1),(x2,y2).....(xn,yn),目标是找到近似函数F(x)，使得损失函数L(y,F(x))的损失值最小。

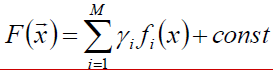
L(y,F(x))常用的两种定义：



假定最优函数为F\*(x),即



提升算法就是假定F(x)是一族基函数fi(x)的加权和



补充：**均方误差的目标函数最优值（取定值）是均值**



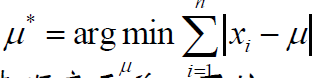
**绝对误差的目标函数的最优解是中位数**

**推导：**

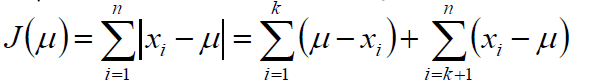
给定样本



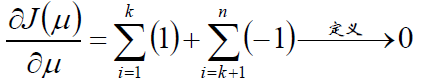
计算



为了便于推导，由于样本顺序无关，不妨假定样本是递增排序的，则



也就是有一部分x是大于u的，有一部分是小于u均值的。



从而，前k个样本数目与后k-1个样本数目相同，即u是中位数。

# AdaBoost算法

Adaptive Boosting称之为自适应boosting，利用该学习器可以提高学习器的性能。

## 5.1算法简介

### 5.1.1AdaBoost简介

AdaBoost自适应提升学习算法和Boosting考虑的点一样

Adaboost自适应在于：**“关注”**被错分的样本，**“器重”**性能好的弱分类器：**（观察下图）**

（1）不同的训练集--->调整样本权重

（2）“关注”--->增加错分样本权重

（3）“器重”--->好的分类器权重大

（4） 样本权重间接影响分类器权重

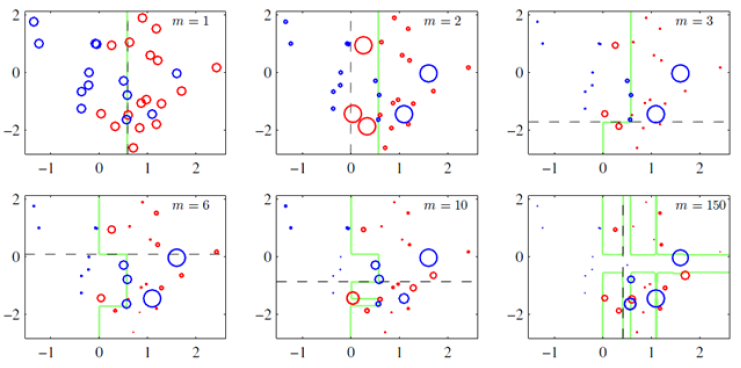
AdaBoost算法的两个核心步骤：

\*\*\*权值调整：AdaBoost算法提高那些被前一轮基分类器错误分类样本的权值，而降低那些被正确分类样本的权值。从而使得那些没有得到正确分类的样本，由于权值的加大而受到后一轮基分类器的更大关注。

\*\*\*基分类器组合：AdaBoost采用加权多数表决的方法。

~加大分类误差率较小的弱分类器的权值，使得它在表决中起到较大的作用。

~减小分类误差率较大的弱分类器的权值，使得它在表决中起较小的作用。



### 5.1.2AdaBoost特点

AdaBoost把多个不同的弱分类算法，用一种非随机的方式组合起来，表现出惊人的性能。

1，可以使用各种方法构建子分类器,Adaboost算法提供的是框架；

2，子分类器容易构造；

3，速度快，且基本不用调参数；

4，泛化错误率低。

### 5.1.3 AdaBoost步骤

Adaboost迭代算法有3步：

1.初始化训练数据的权值分布：假设有N个样本，每个样本赋予相同权值1/N。

2.训练弱分类器：本轮训练中，若某样本分错，则提高它的权值，相反分类正确的样本被降低权值。然后，权值更新过的全体样本被用于训练下一个分类器，使得下一个分类器更关注权重大的难分样本。多次迭代，训练多个弱分类器。

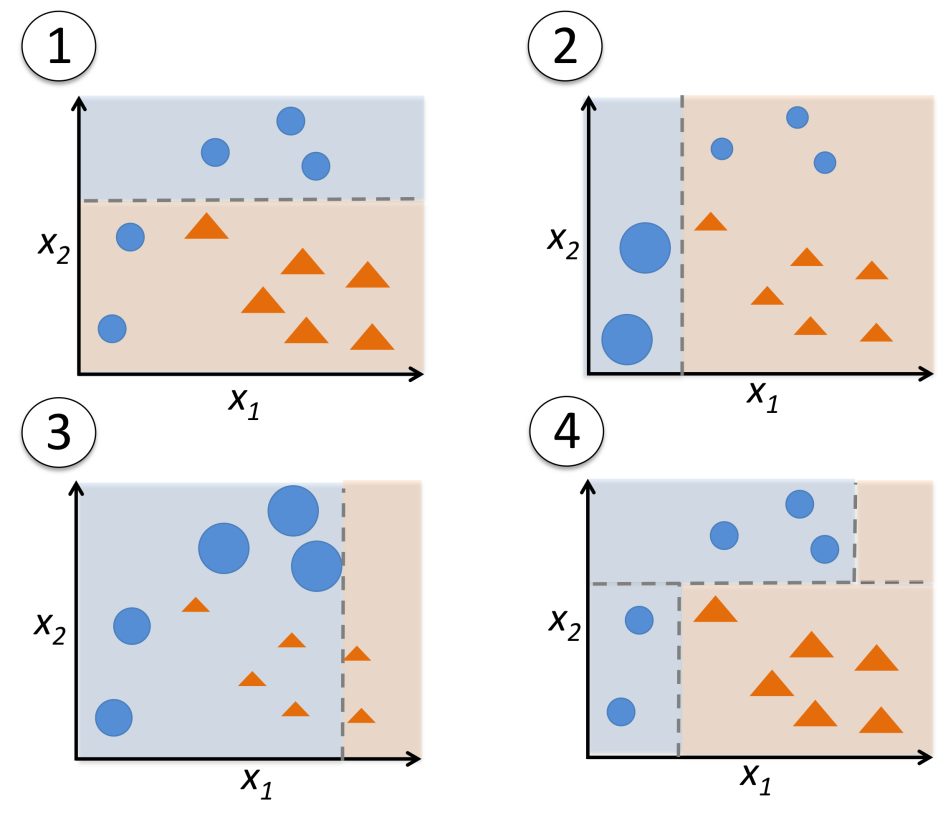
3.加权组合弱分类器：加大分类误差率小的弱分类器的权重，使其在最终表决中起较大作用，而降低分类误差率大的弱分类器的权重，使其在最终表决中起较小作用。

**详细步骤举例：**

AdaBoost和Boosting不同，AdaBoost使用了整个训练集来训练学习器，其中每个训练样本在每次迭代中都会重新被赋值一个权重，在上一弱学习器错误的基础上进行学习进而构建一个更加强大的分类器。下图1代表了一个2类别问题的训练集，其中所有的样本都被赋予了相同的权重，基于此训练集我们得到了一颗单层决策树，图中的虚线表示，它尝试通过最小代价函数（或决策树的不纯度）划分两类样本（三角形或圆形）。

在下一轮中，图2，我们为前面无分类的样本（圆形）赋予了更大的权重，此外，我们还降低了被正确分类的权重，图2错误的划分了圆形类的三个样本。被分错的样本在第三轮Bossting中，被赋予了更大的权重。

假定AdaBoost只包含了3轮的Boosting过程没我们就能够用加权投票方式将不同重采样训练子集上得到的三个弱学习器进行组合，也就是图4（加权组合弱分类器）。



### 5.1.4 AdaBoost数学定义

考虑二分类问题，问题，数据集如下



1. **初始化训练数据的权值分布**



**2.用m=1,2,...,M表示迭代次数**

2.1使用具有权值分布的训练数据集学习得到基本分类器（KNN、决策树、SVM等）



2.2计算在训练数据集上的分类误差率

（不相等返回1，相等返回0）

注：（1）所有误分类点的权重之和。权重越大的误差分类点，其在误差率中占比越大。

（2）若算法终止，构建失败。（后面提到）

2.3计算的相关系数

（em做错的样本，1-em是做对的样本）

该系数表示在最终分类器中的重要性。它是em的单调减函数（说明误差越小的基本分类器，其重要性越高）。系数大于0也要求em<1/2

2.4更新训练数据集的权值分布





这里，Zm是规范化因子，它使成为一个概率分布



1. **构建基本分类器的线性组合**



这里实现了M个基本分类器的加权表决。系数表示基本分类器的重要性，它是em的单调减函数（说明误差越小的基本分类器，其重要性越高）

**4.得到最终分类器**



**推导：为什么？ 若算法终止，构建失败。**

~~AdaBoost提高那些被前一轮弱分类器错误分类样本的权值，而降低那些被争取分类样本的权值。这是通过更新训练数据集的权重向量：来体现的，其中：~~

~~~~

~~对于正确分类的样本，下一轮迭权重为：~~

~~对于错误分类的样本，下一轮权重为：~~

~~分对的样本为1-em：~~

~~~~

~~分错的样本为em：~~

~~~~

~~上述两式子相除，误分类样本的权重是正确分类样本权重的倍，两边同时取指数即可得到结果为：~~

~~（这里可以解释错误率的由来）~~

### 5.1.5构建损失函数并求解am：

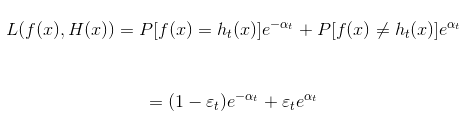
**Adaboost算法利用指数损失函数：**

最小化**指数损失函数**（exponential loss function）



其中f(x)是正确的分类，等于-1或者1，H(x)是分类器的分类结果，等于-1或者1。





所以对该式子求的偏导，得，并令其等于0，得

 其中

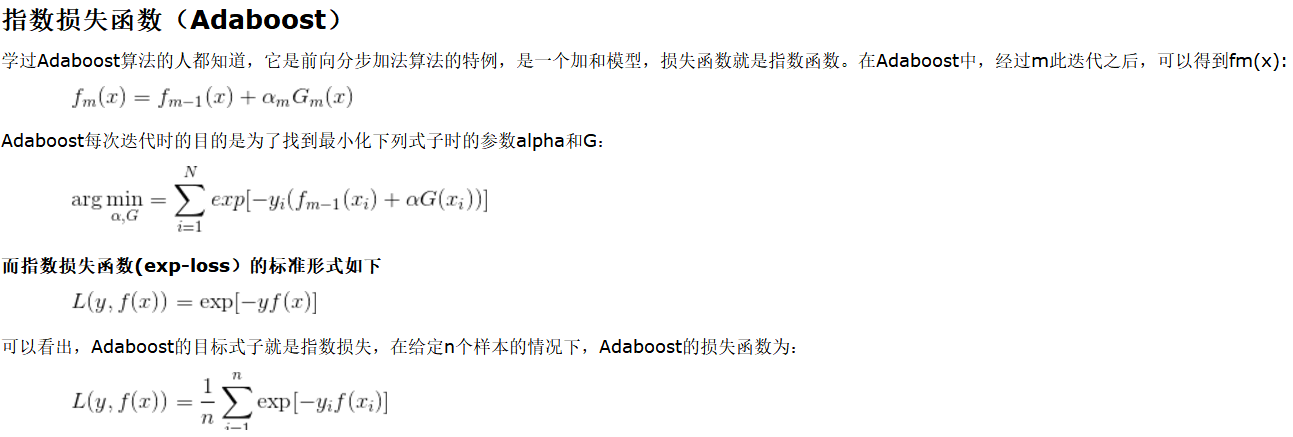
现在考虑要提高的那些被前一轮弱分类器错误分类样本的权值，而降低那些被正确分类样本的权值，就要有，因此。

另外：因此如果在第m次迭代过程中，发现基学习器的误差率，算法终止。因为不满足算法的条件：提高那些被前一轮弱分类器错分类样本的权值，而降低那些被正确分类样本的权值。

综上：当，

也就是错误率小于50%，根据系数am计算公式得到am>0，也就是说凡是错误率小于0.5我们都能得到一个大于0的am，即分类错误率em越小，我们就赋予更高的am权值，反之分类错误率em越大，我们就赋予am更低的权值，即降低权值。

注：



### 5.1.4举例

下表给出了10个训练数据，假设弱分类器由x<v或x>v产生，其中阈值v使得该分类器在训练数据集上分类误差最低，这里v=3.0。试用Adaboost算法学习一个弱分类器。x是一维的样本数据，y为类别标签1/-1，x<=3是+1类，否则-1类。



表中第一列为样本序号，第二列为样本的特征值，第三列为类标签（+1类和-1类），第四列为初始权重，权重值相等均为1/10=0.1，划分标准为x<=0.3第5列存储了预测类标y1。基于权重更新规则，最后一列存储了更新后的权值。

1. 首先计算上述步骤中的权重错误率：



1. 计算Gm(x)的相关系数：



1. 得到相关系数后，根据公式计算更新权重向量



我们首先计算除了Zm的式子：

这里计算被正确分类的样本，预测类标向量y1和真实类标本身y相同，乘积符号为正，本身为正，则加一个负号之后第i个权重的值会被降低：



如果yi预测类标错误，则第i个权重将按照如下方式更新，对分错的样本权重将会增加：



1. 完成上述w权重更新之后在除以Zm进行权重的归一化。

，这里面有7个分对的，3个分错的。

1. 最后一步

由于分正确的样本其对应权重在下一轮boosting中将从初始的0.1降低到0.066/0.914=0.072，分错的样本，其对应权重从0.1提升到0.153/0.914=0.167，权重得到提升。

### 5.1.5例子总结

方法：加大前一轮分错样本权值，减小分对样本权值。而在下一轮迭代中，总是选取让误差率最低的阈值来设计基本分类器，所以误差率e不断降低，在最终的加权投票时的基分类器的权重也相应的增加。

### 5.1.6推广到多分类

接下来讨论多维数据的单层决策树分类器构建方法

前面的例子：给定10个一维样本数据及类别。

1.选某个阈值进行分类（即单节点决策树）：阈值v的一边取+1，另一边取-1，并计算分类误差率；

2.多个不同阈值下分类后，选分类误差率最小的阈值作为本轮的基分类器。

推广到多维数据：

例子中的一维样本数据相当于多维数据中的某个特征向量。

对于多维训练数据，构建单节点决策树时，需要遍历每个特征，最终选择使分类误差率e最小的阈值作为最终分类决策函数。

**Ovr---One vs Rest---将一个n分类问题转化为n个二分类问题**

**Ovo--One vs One----将一个n分类的问题转化为n\*(n-1)/2多个2分类问题**

# 6AdaBoost实战葡萄酒数据

## 6.1导入数据



## 6.2切分数据



## 6.3在500棵单层决策树上训练AdaBoost



## 6.4AdaBoost分类器性能评估



总结：AdaBosst预测准确了所有的训练集类标，与单层决策树相比，它在测试机上表现稍微好一些。单决策树对于训练数据过拟合的程度更加严重一些。总之，我们可以发现Adaboost分类器能够些许提高分类器性能，并且与bagging分类器的准确率接近.

## 6.5绘制决策区域形状



总结：通过观察区域，我们可以看到AdaBosst的决策区域比单层的决策区域更加复杂。

## 6.6总结

集成技术与单独的分类器性能比较，集成学习提高了复杂度，但在实践中，需要衡量是否愿意为适度提高预测性能付出更多的计算成本。

# GradientBoostingDescitionTree（GBDT）

## 7.0提升树

### 7.0.1提升树的原理

1. 提升方法采用**加法模型(**即基函数的线性组合)与**前向分布算法**。以决策树为基函数的提升方法提升方法称为**提升树**。Boosting tree，对分类问题，决策树是二叉决策树；对回归问题，决策树是二叉回归树。

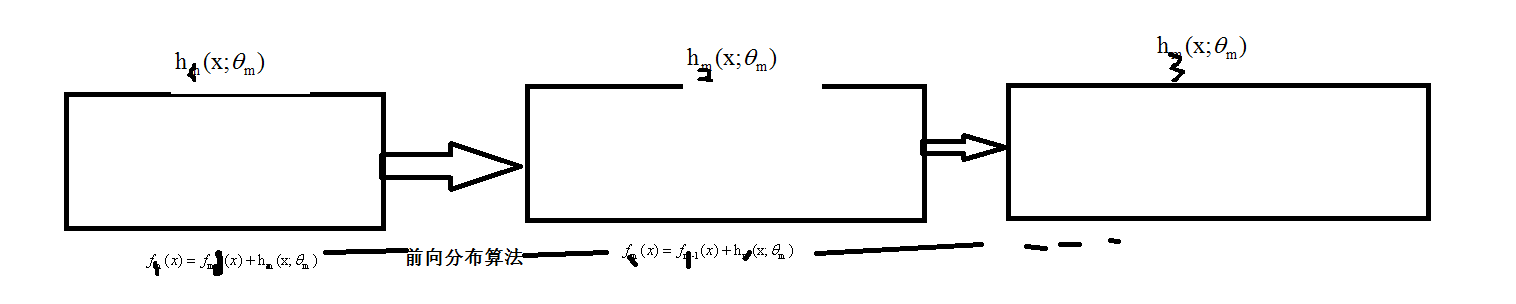
* 提升树模型可以表示为决策树(为基本分类器)的加法模型：



* 提升树算法采用前向分布算法。令为当前模型，则第m步模型为：



其中初始化提升树



* 通过损失函数最小化来确定下一棵决策树参数：



如果使用不同的损失函数，则得到两种不同的提升树算法（预测值为y1，真实值为y）：

（1）回归问题：通常使用平方误差损失函数：



（2）分类问题：通常使用指数损失函数：



* 最后得到损失函数



由于树的线性组合可以很好拟合训练数据，即使数据中的输入和输出之间的关系很复杂也是如此，所以提升树是一个高功能的学习算法。



### 7.0.2提升树算法

**回归树算法:**

输入：训练数据集

输出：提升树

算法步骤：

（1）初始化

1. 对于m=1,2,3......M

\*\*计算残差

\*\*拟合残差学习一颗回归树，得到。

\*\*更新

（3）利用加法模型得到提升树

### 7.0.3GBDT&GBRT

提升是一个机器学习技术，可以用于回归和分类问题，它每一步产生一个弱预测模型，如决策树，并加权累加到总模型中了如果每一步的弱预测模型生成都是依据损失函数的梯度方向，则称之为梯度提升。

梯度提升算法首先给定一个目标损失函数，它的定义域是所有可行的弱函数集合(基函数)；提升算法通过迭代的选择一个负梯度方向上的基函数来逐渐逼近局部极小值。

提升的理论意义：如果一个问题存在弱分类器，则可以通过提升的方式得到强分类器。

***在提升树的基础上，利用梯度下降法，求解目标函数或者损失函数最优解的情况，就称之为梯度提升算法。（理解）***

提升树中，如果损失函数是平方损失函数和指数损失函数时，由于这两种函数求导数跟简单，所以求解下面的参数最优解就比较简单：



如果损失函数是一般函数，该优化问题往往比较难求得，Freidman提出了梯度提升算法来解决该最优值求解问题----利用损失函数的负梯度（梯度下降法）在当前模型的值作为提升树算法中残差的近似值，拟合一颗决策树。在回归问题中，这称之为梯度提升回归树GBRT，在分类问题中，称之为梯度提升决策树GBDT。

梯度提升算法如下：

输入：训练数据集

输出：回归树

算法步骤：

1. 初始化：

（2）对于m=1,2.....M，m指的是串行分类器的个数

1. 对于i=1,2......N， i指的是求残差是yi和f(xi)中的i值

这里计算的是损失负梯度在当前模型的值，将它作为残差的估计。



B.对拟合一颗回归树，得到第m棵树的叶节点区域

C.计算在每个区域上的输出值，



**其中c指的是上面公式中的h(x)决策树或回归树。**

D.更新

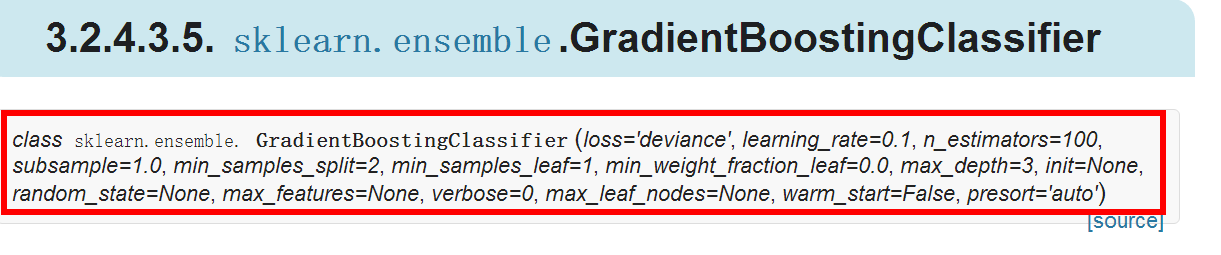
，该决策树将x所属的叶节点的输出作为x的预测值。

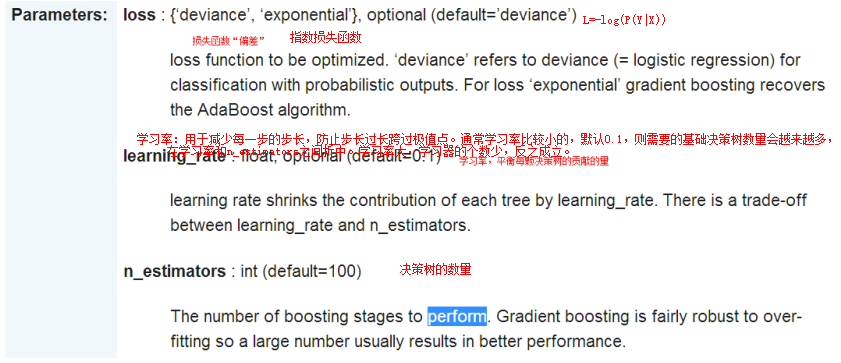
1. 得到回归树



上述梯度提升树算法，在回归问题中，这称之为梯度提升回归树GBRT，在分类问题中，称之为梯度提升决策树GBDT。

## 7.1API的详解





**注意：**

**损失函数：偏差，是用于回归问题的平方误差损失函数L(y,y1)=(y-y1)^2**

**指数损失，用于分类问题L(y,y1)=e^(-y\*y1)**

## 7.2GradientBoosting实战（与随机森林对比）

该案例是在随机森林的基础上修改的，可以对比讲解。

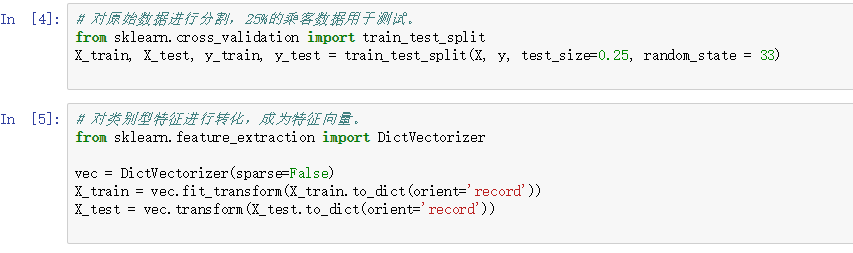
数据地址：

<http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.txt>

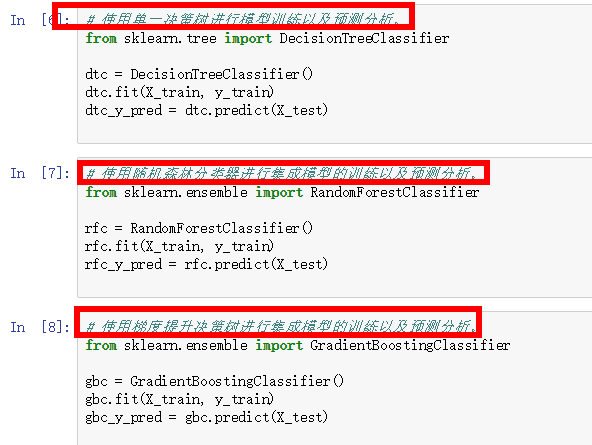
### 7.2.1导包并选取特征



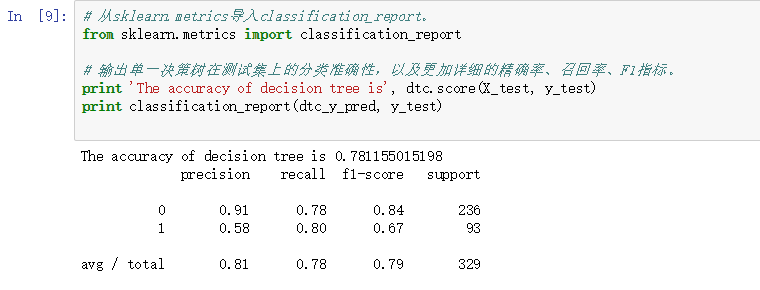
### 7.2.2切分数据及特征处理

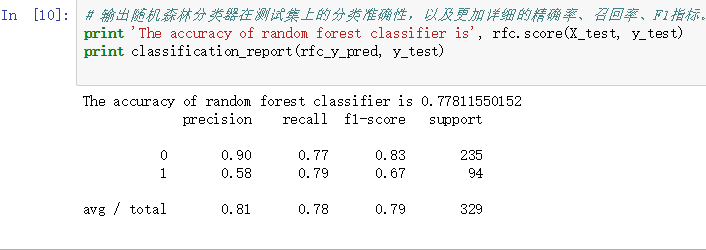


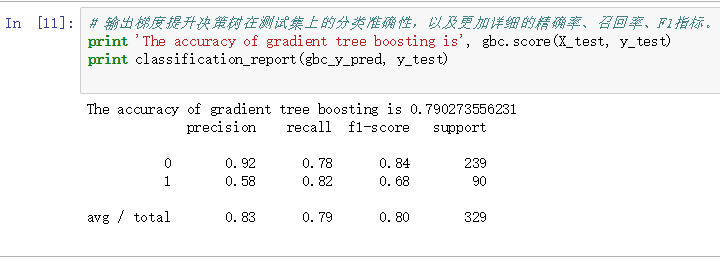
### 7.2.3三种分类器训练及预测



### 7.2.3三种分类器性能评估





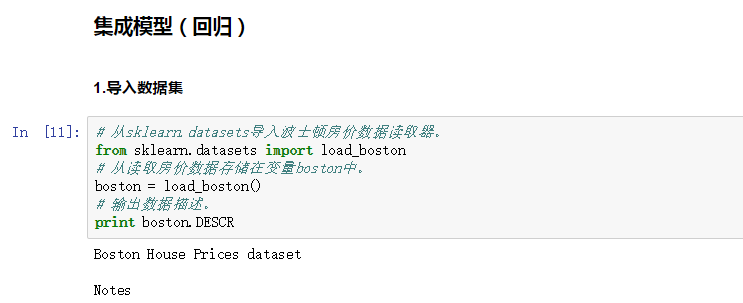


# 8集成模型的回归分析(参考回归课程)

API：

<http://scikit-learn.org/0.17/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.GradientBoostingRegressor>

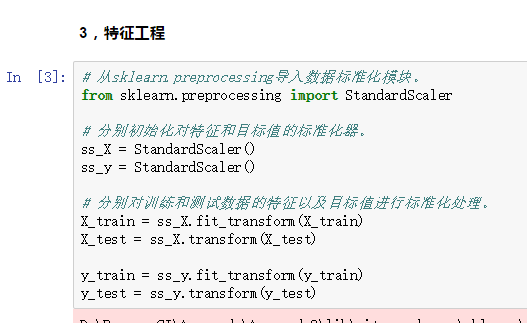
## 8.1导入数据集



## 8.2数据集切分

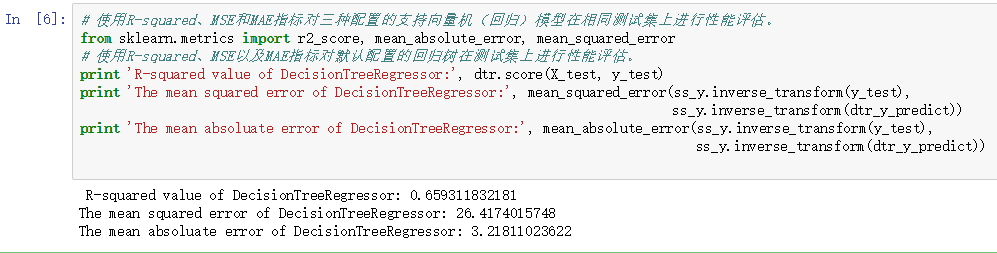


## 8.3特征工程



## 8.4决策树回归器





## 8.5随机森林和极端回归森林回归器



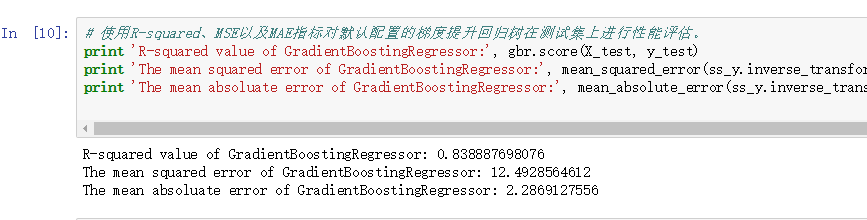
## 8.6随机森林和极端回归森林性能评估



## 8.7GradientBoostingRegressionTree模型（GBDT）



## 8.8GBDT性能评估



# 9.集成学习方法分类总结

目前集成学习方法大概分为两类：

1. 个体学习器之间存在强依赖关系、必须串行生成序列化的方法，代表是Adaboost。
2. 个体学习之间不存在强依赖关系，可同时生成并行化方法，代表是Bagging和随机森林.

**集成学习：**

1. 串行方式：AdaBoost（关注：那些被误分类的点；器重：分类误差比较低的基学习器），GBDT（梯度增强决策树算法）---学习率比较小，n\_estimator（决策树的数量）学习器数量需要增多，同理，学习率较大，n\_estimator学习器数量相应减少。
2. （2）并行方式：Bagging（Bootstrap抽样、基学习器进行学习和预测）、随机森林（基于样本抽样Bagging+基于特征（属性）抽样，+基学习器训练和预测）

# 补充10.集成学习的多样性增强

集成学习中，个体学习器多样性越大越好。通常为了增大个体学习器的多样性，在学习过程中引入随机性。常用的方法包括：对数据样本进行扰动、对输入属性进行扰动、对算法参数进行扰动。

## 10.1数据样本扰动

给定数据集，可以使用采样法从中产生出不同的数据子集。然后在利用不同的数据子集训练出不同的个体学习器。

该方法简单有效，使用广泛。

1. 数据样本扰动对于“不稳定学习器”很有效。“不稳定学习器”是这样一类学习器：训练样本稍加变化就会导致学习器有显著的变动，如决策树和神经网络等。
2. 数据样本扰动对于“稳定学习器”无效。“稳定学习器”是这样一类学习器：学习器对于数据样本的扰动不敏感，如线性学习器、支持向量机、朴素贝叶斯、K近邻学习器等。

如Bagging算法就是利用Bootstrip抽样完成对数据样本的自助采样。

## 10.2输入属性的扰动

训练样本通常由一组属性描述，可以基于这些属性的不同组合产生不同的数据子集，然后在利用这些数据子集训练出不同的个体学习器。

1. 若数据包含了大量冗余的属性，则输入属性扰动效果较好。此时不仅训练出了多样性大的个体，还会因为属性数量的减少而大幅节省时间开销。同时由于冗余属性多，即使减少一些属性，训练个体学习器也不会很差。
2. 若数据值包含少量属性，则不宜采用输入属性扰动法。

## 10.3算法参数的扰动

通常可以通过随机设置不用的参数，比如对模型参数加入小范围的随机扰动，从而产生差别较大的个体学习器。

在使用**交叉验证法（GridSearch网格搜索）来确定基学习器的参数**时，实际上就是用不同的参数训练出来了多个学习器，然后从中挑选出效果最好的学习器。集成学习相当于将所有这些学习器利用起来了。

随机森林学习器就结合了数据样本的扰动及输入属性的扰动。