**分析一个机器学习问题：**  
1.首先搞清楚数据集的内容  
2.数据集： 行--样本(15)、 列---特征个数(4)  
3.特征： (4)  
年龄---青年、 中年、 老年  
收入----高、 中、 低  
学生----是 or 否  
信誉----优 良  
4.类标签： (1)  
最后一列： 买 or 不买 （ 二分类问题）  
5.训练集(训练模型) + 测试集(测试模型的优劣

1. blending 和 bagging

adaboost算法更新数据权重的时候使用归一化。归一化因子: 每一个都要除以所有的总和,这样 是 归一化有了归一化因子，才是概率。总和是1，否则就只是一个数。

因此如果在第m 次迭代过程中，发现基学习器的误差率2 1 m e ，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。

留一法和交叉验证法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。

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。

所有的g是已知的，即blending。另外一种情况是所有g未知，只能通过手上的资料重构g，即learning。aggregation将不同H结合起来得到更好的预测模型，起到了特征转换的效果；对各种效果较好的可能性进行组合，所以，aggregation也起到了正则化（regularization）的效果，让预测模型更具有代表性；

aggregation的两个优势：feature transform和regularization，通常feature transform和regularization是对立的，如果进行feature transform，那么regularization的效果通常很差，反之亦然。也就是说，单一模型通常只能倾向于feature transform和regularization之一，在两者之间做个权衡，aggregation却能将feature transform和regularization各自的优势结合起来。

那对于我们已经选择的性能较好的一些矩gt，如何将它们进行整合、合并，来得到最佳的预测模型呢？这个过程称为blending。

Bias反映的是模型在样本上的输出与真实值之间的误差，即模型本身的精准度，我们知道Bias是受算法模型的复杂度决定的。在概率论和统计学中方差是衡量随机变量或一组数据时离散程度的度量。Variance（方差）反映的是模型每一次输出结果与模型输出期望之间的误差，即模型的稳定性。不同gt的平均误差共识，用偏差bias表示；不同gtgt与共识的差距是多少，反映gt之间的偏差，用方差variance表示；uniform blending的操作时求平均的过程，这样就削减弱化了上式第一项variance的值，从而演算法的表现就更好了，能得到更加稳定的表现。

linear blending，每个gt赋予的权重αt并不相同，其中αt≥0αt≥0。我们最终得到的预测结果等于所有gtgt的线性组合。如何确定αtαt的值，方法是利用误差最小化的思想，找出最佳的αtαt，使Ein(α取最小值。

blending的优点是模型复杂度提高，更容易获得更好的预测模型；缺点是复杂模型也容易带来过拟合的危险。

如何利用已有的一份数据集来构造出不同的gtgt呢？bias-variance，即一个演算法的平均表现可以被拆成两项，一个是所有gtgt的共识（bias），一个是不同gtgt之间的差距是多少（variance）。其中每个gtgt都是需要新的数据集的。只有一份数据集的情况下，如何构造新的数据集？做法就是bootstrapping。bootstrapping的做法是，假设有N笔资料，先从中选出一个样本，再放回去，再选择一个样本，再放回去，共重复N次。这样我们就得到了一个新的N笔资料，这个新的Dt˘Dt˘中可能包含原D里的重复样本点，也可能没有原D里的某些样本，Dt˘Dt˘与D类似但又不完全相同。值得一提的是，抽取-放回的操作不一定非要是N，次数可以任意设定。例如原始样本有10000个，我们可以抽取-放回3000次，得到包含3000个样本的Dt˘Dt˘也是完全可以的。利用bootstrap进行aggragation的操作就被称为bagging。

2、 bagging和boosting

先介绍Bagging方法：

Bagging即套袋法，其算法过程如下：

1. 从原始样本集中抽取训练集。每轮从原始样本集中使用Bootstraping的方法抽取n个训练样本（在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）。共进行k轮抽取，得到k个训练集。（k个训练集之间是相互独立的）

2. 每次使用一个训练集得到一个模型，k个训练集共得到k个模型。（注：这里并没有具体的分类算法或回归方法，我们可以根据具体问题采用不同的分类或回归方法，如决策树、感知器等）

3. 对分类问题：将上步得到的k个模型采用投票的方式得到分类结果；对回归问题，计算上述模型的均值作为最后的结果。（所有模型的重要性相同）

Boosting：

在分类问题中它通过改变训练样本的权重，学习多个分类器，并将分类器进行线性组合。

AdaBoosting方式每次使用的是全部的样本，每轮训练改变样本的权重。下一轮训练的目标是找到一个函数f 来拟合上一轮的残差。当残差足够小或者达到设置的最大迭代次数则停止。Boosting会减小在上一轮训练正确的样本的权重，增大错误样本的权重。（对的残差小，错的残差大）梯度提升的Boosting方式是使用代价函数对上一轮训练出的模型函数f的偏导来拟合残差。

**Bagging，Boosting二者之间的区别**

Bagging和Boosting的区别：

1）样本选择上：

Bagging：训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。

Boosting：每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。

2）样例权重：

Bagging：使用均匀取样，每个样例的权重相等

Boosting：根据错误率不断调整样例的权值，错误率越大则权重越大。

3）预测函数权重：

Bagging：所有预测函数的权重相等。

Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

4）是否可以并行计算：

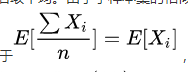
Bagging：各个预测函数可以并行生成

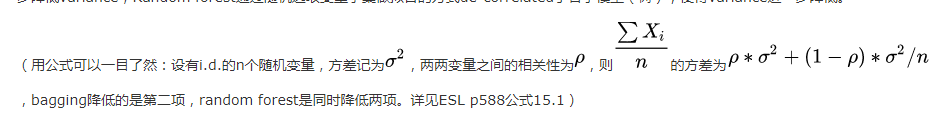
Boosting：各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。

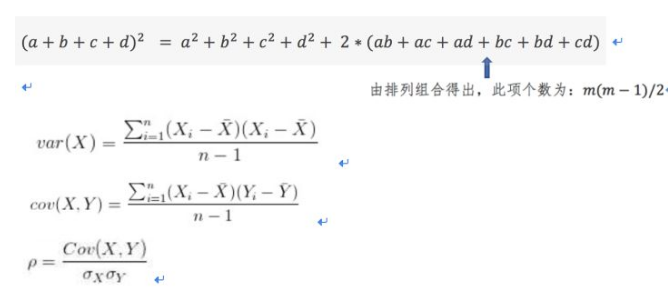
5）为什么说bagging是减少variance，而boosting是减少bias？（重要）

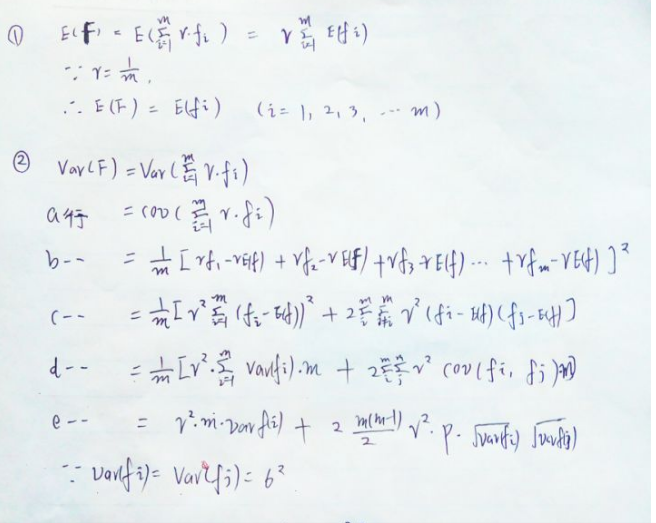
https://www.cnblogs.com/earendil/p/8872001.html

https://www.zhihu.com/question/26760839

Bagging对样本重采样，对每一重采样得到的子样本集训练一个模型，最后取平均。由于子样本集的相似性以及使用的是同种模型，因此各模型有近似相等的bias和variance（事实上，各模型的分布也近似相同，但不独立）。由于 ，所以bagging后的bias和单个子模型的接近，一般来说不能显著降低bias。另一方面，若各子模型独立，则有 ，此时可以显著降低variance。若各子模型完全相同，则 

，此时不会降低variance。bagging方法得到的各子模型是有一定相关性的，属于上面两个极端状况的中间态，因此可以一定程度降低variance。为了进一步降低variance，Random forest通过随机选取变量子集做拟合的方式de-correlated了各子模型（树），使得variance 





boosting从优化角度来看，是用forward-stagewise这种贪心法去最小化损失函数 。例如，常见的AdaBoost即等价于用这种方法最小化exponential loss： 。所谓forward-stagewise，就是在迭代的第n步，求解新的子模型f(x)及步长a（或者叫组合系数），来最小化 ，这里 是前n-1步得到的子模型的和。因此boosting是在sequential地最小化损失函数，其bias自然逐步下降。但由于是采取这种sequential、adaptive的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低variance。所以说boosting主要还是靠降低bias来提升预测精度。

通常来说boosting是在优化loss function，在降低loss，那么很显然，这在很大程度上是减少bias。而bagging，之所以进行bagging，是希望模型能够具有更好的鲁棒性，也就是稳定性，希望避免过拟合，显然这就是在减少variance。

bagging的方法主要是随机取样，每个模型的目标函数跟单模型并没有很大的区别，所以bias变化不大，而多个模型的集成降低异常点的影响，这两个方法都可以降低variance

boosting的方法通过残差学习，最后得预测值发生了改变，是由多个模型预测值与每个模型权重重新表示，纠错之后的boosting预测值自然是更接近真实值，因此bais偏离真实值的误差会降低

下面是将决策树与这些算法框架进行结合所得到的新的算法：

1. Bagging + 决策树 = 随机森林

2. AdaBoost + 决策树 = 提升树

3. Gradient Boosting + 决策树 = GBDT

**Adaboost 算法**

1、 每一轮如何改变训练样本的权重？

提高哪些被前一轮弱分类器错误分类样本权重，降低正确分类的权重

2、 如何将弱分类器组成强分类器？

加权多数表决的方法：加大分类误差率小的弱分类器的权重，减小分类误差率大的弱分类器的权重。

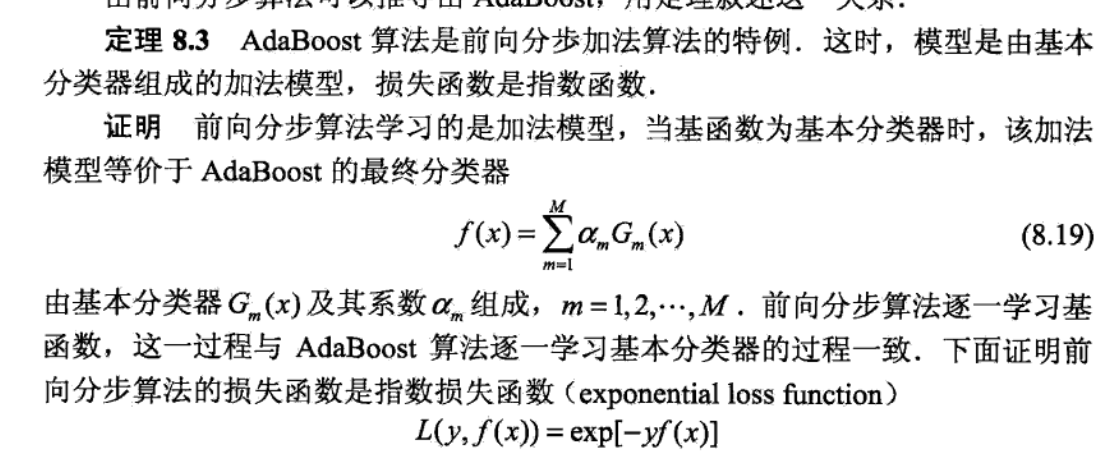
优缺点：

优点：泛化错误率低，易编码，可以应用在大部分分类器上，无参数可调整。

缺点：对离群数据点敏感。

适用数据类型：数值型和标称型数据。

最基本的性质是在学习过程中不断减小训练误差。



提升树：采用加法模型与前向分布算法，以决策树为基函数数的提升方法。

前向分布算法求解经验风险极小化即损失函数极小化问题：因为学习是加法模型，从前向后每一步只学习一个基函数及其系数，逐步逼近优化目标函数。

对于二类分类问题;提升树算法只需将Adaboost的基本分类器限制为二类树就行。

对于回归：采用平方误差损失函数，简单拟合当前模型残差即可

https://blog.csdn.net/qq\_34896915/article/details/73771287

**决策树模型：**

其基本思想是以信息熵为度量构造一棵熵值下降最快的树，到叶子节点处的熵值为零。建立决策树的关键，即在当前状态下选择哪个属性作为分类依据。

决策树学习是由训练数据集估计条件概率模型。决策树学习的损失函数通常是正则化的的极大然函数。策略是以损失函数为目标函数的最小化。从所有可能的决策树中选取最优决策树是NP问题。通常采用启发式学习算法。

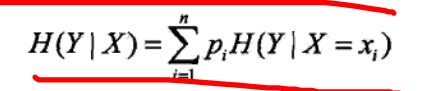
决策树通过自下而上进行减枝，就是去掉过于细分的叶节点，使其回退到父节点，甚至更高的节点，然后将父节点或更高的节点改为新的叶节点。

决策树包括特征选择、生成、剪纸过程。生成只考虑局部最优，剪纸则考虑全局最优。

特征选择准则是信息增益或增益比，选择信息增益最大的特征。

熵表示对随机变量不确定性的度量，熵只依赖X的分布于X取值无关。熵越大，随机不确定性就越大。

条件熵：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。



信息增益：表示得知特征X的信息而使得Y的信息的不确定性减少的程度。

决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

在分类问题困难时，可以使用信息增益比，因为训练数据集的经验熵大的时候，信息增益值会偏大，反之亦然，使用信息增益比可以纠正。信息增益比等于信息增益与熵的比值。

决策的生成：

ID3算法是各个节点用信息增益选择特征递归构建决策树。相当于使用极大似然法进行概率模型的选择。此算法只有树的生成，容易过拟合。

ID3生成算法：

A． 如果所有种类的属性都处理完毕，则返回。否则，执行步骤B.

B． 计算出信息增益最大的属性a，将该属性作为一个节点。如果仅凭借a就可以对样本分类，则返回。否则执行步骤C。

C． 对属性a 的每个可能取值v 执行以下操作：

1.将v 的样本作为s 的一个子集Sv

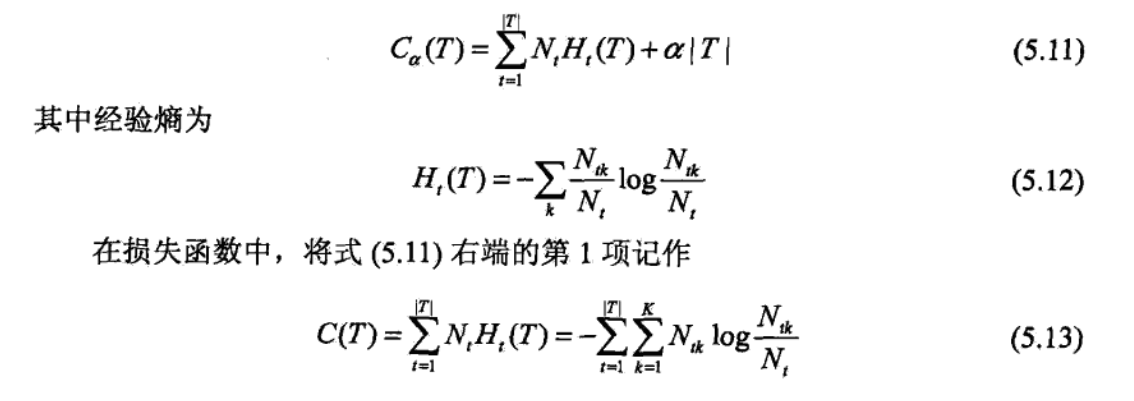
2.生成剩余属性集合AT=A-{a}

3.以样本集合Sv 和属性集合AT 为输入，递归执行ID3 算法。

C4,5与ID3一样，只是采用信息增益比。

决策树的减枝：

通过极小化决策树整体的损失函数或代价函数来实现。（将损失函数背过）。



利用损失函数最小化原则进行减枝等价于正则化的最大似然估计进行模型选择。也可以

说将叶节点的熵求和，值越小说明分类越精确。

如果一组叶节点回缩到其父节点之前整体树的损失函数大于回缩后的损失函数，则进行剪枝，即将父节点变更为新的叶节点。这种缺点是在局部进行。

**决策树的CART（分类回归树）算法：**

在给定输入随机变量X条件下输出随机变量Y的条件概率分布的学习方法。一个属性的信息增益(率)/gini指数越大，表明属性对样本的熵减少的能力更强，这个属性使得数据由不确定性变成确定性的能力越强

决策树算法的特点

（1） 决策树的优点：

 直观，便于理解，小规模数据集有效

 执行效率高，执行只需要一次构建，可反复使用

（2）决策树的缺点：

 处理连续变量不好，较难预测连续字段

 类别较多时，错误增加的比较快

 对于时间序列数据需要做很多的预处理

 可规模性一般

 实际分类的时候只能根据一个字段进行

**CART生成：**

对回归树用平方误差最小化准则和对分类树用基尼指数最小化准则，进行特征选择，生成二叉树。

基尼指数表示集合D的不确定性，基尼指数越大样本集合的不确定性越大。

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。

**CART剪枝：**

1、 首先从生成算法产生的决策树T0底端不断剪枝直到根节点，形成一个子树序列{T0…Tn}

2、 然后通过交叉验证法在独立的验证集上就行测试，选择最优子树。

那么如何确定修剪多少叶子，修剪哪些叶子呢？假设由C&RT算法得到一棵完全长成树（fully-grown tree），总共10片叶子。首先分别减去其中一片叶子，剩下9片，

将这10种情况比较，取EinEin最小的那个模型；然后再从9片叶子的模型中分别减去一片，剩下8片，将这9种情况比较，取EinEin最小的那个模型。以此类推，继续修建

叶子。这样，最终得到包含不同叶子的几种模型，将这几个使用regularized decision tree的error function来进行选择，确定包含几片叶子的模型误差最小，就选

择该模型。另外，参数λ可以通过validation来确定最佳值。

在决策树中预测中，还会遇到一种问题，就是当某些特征缺失的时候，没有办法进行切割和分支选择。一种常用的方法就是surrogate branch，即寻找与该特征相似的替代feature。如何确定是相似的feature呢？做法是在决策树训练的时候，找出与该特征相似的feature，如果替代的feature与原feature切割的方式和结果是类似的，那么就表明二者是相似的，就把该替代的feature也存储下来。当预测时遇到原feature缺失的情况，就用替代feature进行分支判断和选择。

Bagging和Decison Tree算法各自有一个很重要的特点。Bagging具有减少不同gtgt的方差variance的特点。这是因为Bagging采用投票的形式，将所有gtgtuniform结合起来，起到了求平均的作用，从而降低variance。而Decision Tree具有增大不同gtgt的方差variance的特点。这是因为Decision Tree每次切割的方式不同，而且分支包含的样本数在逐渐减少，所以它对不同的资料D会比较敏感一些，从而不同的D会得到比较大的variance。

cart总结：

CART 每次都选择当前数据集中具有最小Gini 信息增益的特征作为结点划分决策树。C4.5和ID3但其生成的决策树分支、规模较大，CART 算法的二分法可以简化决策树的规模，提

高生成决策树的效率.。对于连续特征，CART也是采取和C4.5 同样的方法处理。为了避免过拟合(Overfitting)，CART 决策树需要剪枝（后剪枝）。预测过程当然也就

十分简单，根据产生的决策树模型，延伸匹配特征值到最后的叶子节点即得到预测的类别。

**Random Forest(随机森林)：**

随机森林在bagging基础上做了修改。

从样本集中用Bootstrap采样选出n个样本；

从所有属性中随机选择k个属性，选择最佳分割属性作为节点建立CART决策树；

重复以上两步m次，即建立了m棵CART决策树

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

随机森林在以决策树为基学习器构建Bagging 集成（样本的随机选取）的基础上，进一步在决策树的训练过程中引入随机属性选择。具体来说，传统决策树

在选择划分属性时是在当前节点的属性集合（假设有d 个属性）中选择一个最优属性；而在RF 随机森林中，对基决策树的每个节点，先从该节点的属性集合中

随机选择一个包含K 个属性的子集，然后在从这个子集中选择一个最优属性用于划分。K=d 就是传统决策树，K=1 则是随机选取一个属性用于划分，一般情况

K d 2  log 。

随机森林的随机性体现在每棵树的训练样本是随机的，树中每个节点的分裂属性也是随机选择的。有了这2 个随机因素，即使每棵决策树没有进行剪

枝，随机森林也不会产生过拟合的现象。

Random Forest算法的优点主要有三个。第一，不同决策树可以由不同主机并行训练生成，效率很高；第二，随机森林算法继承了C&RT的优点；第三，将所有的决策树通过bagging的形式结合起来，避免了单个决策树造成过拟合的问题。

通常来说，需要移除的特征分为两类：一类是冗余特征，即特征出现重复，例如“年龄”和“生日”；另一类是不相关特征，例如疾病预测的时候引入的“保险状况”。

那么，如何对许多维特征进行筛选呢？我们可以通过计算出每个特征的重要性（即权重），然后再根据重要性的排序进行选择即可。然而，对于非线性模型来说

，因为不同特征可能是非线性交叉在一起的，所以计算每个特征的重要性就变得比较复杂和困难。

RF中，特征选择的核心思想是random test。random test的做法是对于某个特征，如果用另外一个随机值替代它之后的表现比之前更差，则表明该特征比较重要，所占的权重应该较大，不能用一个随机值替代。相反，如果随机值替代后的表现没有太大差别，则表明该特征不那么重要，可有可无。所以，通过比较某特征被随机值替代前后的表现，就能推断出该特征的权重和重要性。

那么random test中的随机值如何选择呢？通常有两种方法：一是使用uniform或者gaussian抽取随机值替换原特征；一是通过permutation的方式将原来的所

有N个样本的第i个特征值重新打乱分布（相当于重新洗牌）。

RF中，树的个数越多，模型越稳定越能表现得好。

GBDT:

https://www.cnblogs.com/ModifyRong/p/7744987.html

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。这种在函数域的梯度提升观点对机器学习的很多领域有深刻影响。

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BRT，在分类问题中，称之为梯度提升决策树GBDT。

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）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）

学习率是通过交叉验证来得到的学习率小（步长小），那么学习的时间和速度就长。那么决策树的数量也就是学习器的数量需要增多来加快速度。 相反学习率大

，那么决策树的数量响应减少。

GBDT 本身需要融合多棵决策树，而不是像是adaboost算法仅仅是将若干个异质的分类器联合在一起构成一个强分类器的过程，这里是依靠,全部是决策树的模

型,进行加和的形式构成了一个提升树。

梯度提升是解决全局最优解的一个最好的方式,当然最好是随机梯度提升。无论哪个损失函数,希望损失函数的期望越小越好.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。

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的很多领域有深刻影响。

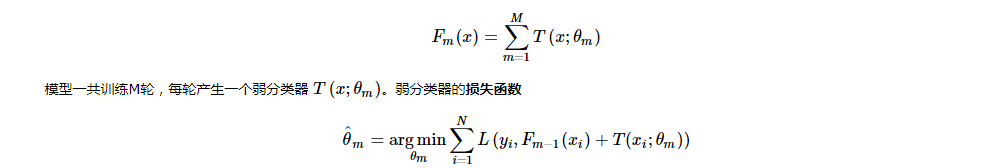
首先每轮bootstrap得到的D’中每个样本会赋予不同的权重u(t)u(t；然后在每个decision tree中，利用这些权重训练得到最好的gtg；最后得出每个gtgt所占的权重，线性组合得到G。这种模型称为AdaBoost-D Tree。

原因大概有几个，一是效果确实挺不错。二是即可以用于分类也可以用于回归。三是可以筛选特征。

训练过程（看网站公式）：

首先gbdt 是通过采用加法模型（即基函数的线性组合），以及不断减小训练过程产生的残差来达到将数据分类或者回归的算法。 gbdt通过多轮迭代,每轮迭代产生一个弱分类器，每个分类器在上一轮分类器的**残差**基础上进行训练。对弱分类器的要求一般是足够简单，并且是低方差和高偏差的。因为训练的过程是通过降低偏差来不断提高最终分类器的精度。 弱分类器一般会选择为CART TREE（也就是分类回归树）。由于上述高偏差和简单的要求 每个分类回归树的深度不会很深。最终的总分类器 是将每轮训练得到的弱分类器加权求和得到的（也就是加法模型）。

        模型最终可以描述为：



gbdt 通过经验风险极小化来确定下一个弱分类器的参数。具体到损失函数本身的选择也就是L的选择，有平方损失函数，0-1损失函数，对数损失函数等等。如果我们选择平方损失函数，那么这个差值其实就是我们平常所说的残差。但是其实我们真正关注的，1.是希望损失函数能够不断的减小，2.是希望损失函数能够尽可能快的减小。

所以如何尽可能快的减小呢？

让损失函数沿着梯度方向的下降。这个就是gbdt 的 gb的核心了。 利用损失函数的负梯度在当前模型的值作为回归问题提升树算法中的残差的近似值去拟合一个回归树。gbdt 每轮迭代的时候，都去拟合损失函数在当前模型下的负梯度。这样每轮训练的时候都能够让损失函数尽可能快的减小，尽快的收敛达到局部最优解或者全局最优解。

gbdt如何选择特征？

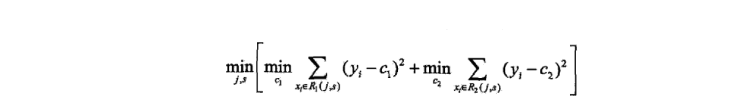
gbdt选择特征的细节其实是想问你CART Tree生成的过程。这里有一个前提，gbdt的弱分类器默认选择的是CART TREE。其实也可以选择其他弱分类器的，选择的前提是低方差和高偏差。框架服从boosting 框架即可。

下面我们具体来说CART TREE(是一种二叉树) 如何生成。CART TREE 生成的过程其实就是一个选择特征的过程。假设我们目前总共有 M 个特征。第一步我们需要从中选择出一个特征 j，做为二叉树的第一个节点。然后对特征 j 的值选择一个切分点 m. 一个 样本的特征j的值 如果小于m，则分为一类，如果大于m,则分为另外一类。如此便构建了CART 树的一个节点。其他节点的生成过程和这个是一样的。

现在的问题是在每轮迭代的时候，如何选择这个特征 j,以及如何选择特征 j 的切分点 m:

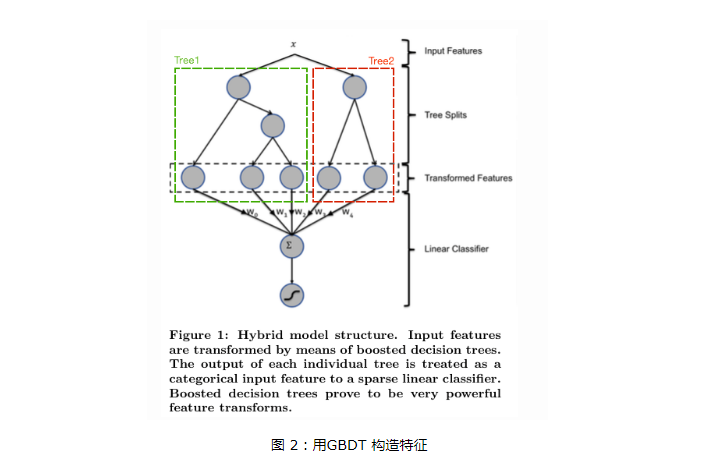
原始的gbdt的做法非常的暴力，首先遍历每个特征，然后对每个特征遍历它所有可能的切分点，找到最优特征 m 的最优切分点 j。

如何衡量我们找到的特征 m和切分点 j 是最优的呢？如果对这段代码不是很了解的，可以先去看看李航第五章中对CART TREE 算法的叙述。在这里，我们先遍历训练样本的所有的特征，对于特征 j，我们遍历特征 j 所有特征值的切分点 c。找到可以让下面这个式子最小的特征 j 以及切分点c.



gbdt 如何构建特征 ?

其实说gbdt 能够构建特征并非很准确，gbdt 本身是不能产生特征的，但是我们可以利用gbdt去产生特征的组合。在CTR预估中，工业界一般会采用逻辑回归去进行处理,逻辑回归本身是适合处理线性可分的数据，如果我们想让逻辑回归处理非线性的数据，其中一种方式便是组合不同特征，增强逻辑回归对非线性分布的拟合能力。利用gbdt去产生有效的特征组合，以便用于逻辑回归的训练，提升模型最终的效果。



如图 2所示，我们 使用 GBDT 生成了两棵树，两颗树一共有五个叶子节点。我们将样本 X 输入到两颗树当中去，样本X 落在了第一棵树的第二个叶子节点，第二颗树的第一个叶子节点，于是我们便可以依次构建一个五纬的特征向量，每一个纬度代表了一个叶子节点，样本落在这个叶子节点上面的话那么值为1，没有落在该叶子节点的话，那么值为 0.

于是对于该样本，我们可以得到一个向量[0,1,0,1,0] 作为该样本的组合特征，和原来的特征一起输入到逻辑回归当中进行训练。实验证明这样会得到比较

显著的效果提升。



GBDT 如何用于分类 ？（看博客）

首先明确一点，gbdt 无论用于分类还是回归一直都是使用的CART 回归树。不会因为我们所选择的任务是分类任务就选用分类树，这里面的核心是因为gbdt 每轮的训练是在上一轮的训练的残差基础之上进行训练的。这里的残差就是当前模型的负梯度值 。这个要求每轮迭代的时候，弱分类器的输出的结果相减是有意义的。残差相减是有意义的。

        如果选用的弱分类器是分类树，类别相减是没有意义的。上一轮输出的是样本 x 属于 A类，本一轮训练输出的是样本 x 属于 B类。 A 和 B 很多时候甚至都没有比较的意义，A 类- B类是没有意义的。

随机森林的优点：

实现简单，训练速度快，泛化能力强，可以并行实现，因为训练时树与树之间是相互独立的；

相比单一决策树，能学习到特征之间的相互影响，且不容易过拟合；

能处理高维数据（即特征很多），并且不用做特征选择，因为特征子集是随机选取的；

对于不平衡的数据集，可以平衡误差；

相比SVM，不是很怕特征缺失，因为待选特征也是随机选取；

训练完成后可以给出哪些特征比较重要。

随机森林的缺点：

在噪声过大的分类和回归问题还是容易过拟合；

相比于单一决策树，它的随机性让我们难以对模型进行解释。

随机森林和GBDT的区别：

1、随机森林采用的bagging思想，而GBDT采用的boosting思想。这两种方法都是Bootstrap思想的应用，Bootstrap是一种有放回的抽样方法思想。虽然都是有放回的抽样，但二者的区别在于：Bagging采用有放回的均匀取样，而Boosting根据错误率来取样（Boosting初始化时对每一个训练样例赋相等的权重1／n，然后用该算法对训练集训练t轮，每次训练后，对训练失败的样例赋以较大的权重），因此Boosting的分类精度要优于Bagging。Bagging的训练集的选择是随机的，各训练集之间相互独立，弱分类器可并行，而Boosting的训练集的选择与前一轮的学习结果有关，是串行的。

2、组成随机森林的树可以是分类树，也可以是回归树；而GBDT只能由回归树组成。

3、组成随机森林的树可以并行生成；而GBDT只能是串行生成。

4、对于最终的输出结果而言，随机森林采用多数投票等；而GBDT则是将所有结果累加起来，或者加权累加起来。

5、随机森林对异常值不敏感；GBDT对异常值非常敏感。

6、随机森林对训练集一视同仁；GBDT是基于权值的弱分类器的集成。

7、随机森林是通过减少模型方差提高性能；GBDT是通过减少模型偏差提高性能。

---------------------

**逻辑斯特回归：**

<https://www.cnblogs.com/ModifyRong/p/7739955.html>

  如何凸显你是一个对逻辑回归已经非常了解的人呢。那就是用一句话概括它！**逻辑回归假设数据服从伯努利分布,通过极大化似然函数的方法，运用梯度下降来求解参数，来达到将数据二分类的目的。**

     这里面其实包含了5个点 1：逻辑回归的假设，2：逻辑回归的损失函数，3：逻辑回归的求解方法，4：逻辑回归的目的，5:逻辑回归如何分类。这些问题是考核你对逻辑回归的基本了解。

* **逻辑回归在训练的过程当中，如果有很多的特征高度相关或者说有一个特征重复了100遍，会造成怎样的影响？**
* 先说结论，如果在损失函数最终收敛的情况下，其实就算有很多特征高度相关也不会影响分类器的效果。
* 但是对特征本身来说的话，假设只有一个特征，在不考虑采样的情况下，你现在将它重复100遍。训练以后完以后，数据还是这么多，但是这个特征本身重复了100遍，实质上将原来的特征分成了100份，每一个特征都是原来特征权重值的百分之一。
* 如果在随机采样的情况下，其实训练收敛完以后，还是可以认为这100个特征和原来那一个特征扮演的效果一样，只是可能中间很多特征的值正负相消了。
* **为什么我们还是会在训练的过程当中将高度相关的特征去掉？**
  + 去掉高度相关的特征会让模型的可解释性更好
  + 可以大大提高训练的速度。如果模型当中有很多特征高度相关的话，就算损失函数本身收敛了，但实际上参数是没有收敛的，这样会拉低训练的速度。其次是特征多了，本身就会增大训练的时间。

**4.逻辑回归的优缺点总结**

    面试的时候，别人也经常会问到，你在使用逻辑回归的时候有哪些感受。觉得它有哪些优缺点。

**在这里我们总结了逻辑回归应用到工业界当中一些优点：**

* 形式简单，模型的可解释性非常好。从特征的权重可以看到不同的特征对最后结果的影响，某个特征的权重值比较高，那么这个特征最后对结果的影响会比较大。
* 模型效果不错。在工程上是可以接受的（作为baseline)，如果特征工程做的好，效果不会太差，并且特征工程可以大家并行开发，大大加快开发的速度。
* 训练速度较快。分类的时候，计算量仅仅只和特征的数目相关。并且逻辑回归的分布式优化sgd发展比较成熟，训练的速度可以通过堆机器进一步提高，这样我们可以在短时间内迭代好几个版本的模型。
* 资源占用小,尤其是内存。因为只需要存储各个维度的特征值，。
* 方便输出结果调整。逻辑回归可以很方便的得到最后的分类结果，因为输出的是每个样本的概率分数，我们可以很容易的对这些概率分数进行cutoff，也就是划分阈值(大于某个阈值的是一类，小于某个阈值的是一类)。

**但是逻辑回归本身也有许多的缺点:**

* 准确率并不是很高。因为形式非常的简单(非常类似线性模型)，很难去拟合数据的真实分布。
* 很难处理数据不平衡的问题。举个例子：如果我们对于一个正负样本非常不平衡的问题比如正负样本比 10000:1.我们把所有样本都预测为正也能使损失函数的值比较小。但是作为一个分类器，它对正负样本的区分能力不会很好。
* 处理非线性数据较麻烦。逻辑回归在不引入其他方法的情况下，只能处理线性可分的数据，或者进一步说，处理二分类的问题 。
* 逻辑回归本身无法筛选特征。有时候，我们会用gbdt来筛选特征，然后再上逻辑回归。

**三种梯度下降法：**

<https://www.cnblogs.com/lliuye/p/9451903.html>

线性回归：

偏差又称为表观误差，是指个别测定值与测定的平均值之差；误差通常指预测值与实际结果的偏差。统计学中的误差也称残差；预测误差是指预测结果与预测对象发展变

化的真实结果的差距。分为预测的相对误差与绝对误差。绝对误差是预测值与实际观测值的绝对差距，相对误差是绝对差距相对于观测值的百分比。

四分位差主要用于测度顺序数据的离散程度。四分位差反映了中间50%数据的离散程度，其数值越小，说明中间的数据越集中；其数值越大，说明中间的数据越分

散。四分位差不受极值的影响。此外，由于中位数处于数据的中间位置。

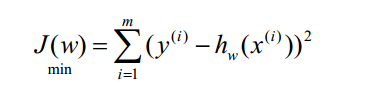
离散系数：离散系数又称变异系数，CV(Coefficient of Variance)表示。CV(Coefficient of Variance)：标准差与均值的比值。离散系数越小，数据的离散程度就越小，反之，亦然。

根据计算过程的可逆性，如若对数分布能够写成随机变量的二次形式，则该分布必然是正态分布。

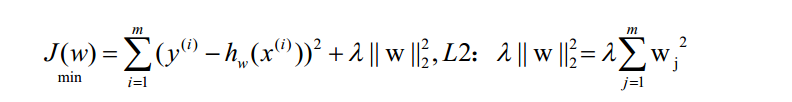
局部加权线性回归（ Locally Weight Linear Regression， WLR） 是一种非参数学习方法，  
为什么局部加权回归又叫非参数学习方法呢？  
参数学习方法是这样一种方法： 在训练完成数据之后得到一系列的训练参数， 然后根据  
训练参数来预测新样本的值， 这是不在依赖之前的训练数据了， 参数值是确定的。 而非参数  
学习方法是这样的一种算法： 在预测新样本的时候每次都会重新训练数据得到最新的参数  
值， 也就是每次预测新样本都会依赖训练数据集合， 所以每次得到的参数是不确定的。

**岭回归：**

岭回归是基于 **L2** 罚项的模型， 只是在最小二乘代价函数中加入权重的平方和。  
线性回归的目标函数为：

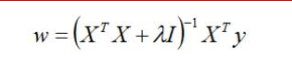


岭回归的目标函数：



通过增加超参数 λ 的值， 我们可以增加正则化的强度， 同时， 也就降低了权重对模型的影响。 一定要注意： 正则化项不影响截距 *w*0、

更改为补充的：  
如果数据的特征比样本点多的场景我们怎么办？ 是否还可以用线性回归和局部加权回归方法呢？ 答案是否定的， 因为线性回归和局部加权回归是需要 (*X T X* )-1存在的前提下。而当数据的特征数比样本点多的时候我们无法保证 (*X T X* )-1存在， 因此， 为了达到进行回归预测的目的， 我们引入岭回归， 即在原目标函数的基础上加入一个 λ*I* ， 使得矩阵变成非奇异矩阵， 进而能对 (*X T X* + λ*I*) 求逆。 其中 I 是一个 m\*m 的单位矩阵， 对角上元素全为 1，其他元素全为 0， 而λ 是自定义的数值， 在这种情况下回归系数的计算公式将变成：



为了实现岭回归和缩减技术， 首先需要对特征进行标准化处理， 在使用上述公式进行岭回归求解。岭回归最先是用来处理特征数大于样本数得情况， 现在也用于在估计中加入偏差，从而得到更好的估计， 即控制模型参数， 结构风险最小化。

**岭回归优点：**  
1.在特征数 M 大于样本数 N 时， *X T X* 不可逆， 故不能直接用 LR， 而岭回归就可  
以。  
2.通过引入 λ 惩罚项， 防止过拟合

在实际案例学习之前， 一定要搞清楚回归问题和分类问题的差别： 待预测的目标是否为连续变量。

熵：

见10天决策树文档

回归树及随机森林：见10天文档

1、回归树原理

2、分类树原理

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。

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

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。

3 算法参数的扰动（学习率，最大深度树，最小深度树）

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。