****ROC曲线****（receiver operating characteristic curve），又称为感受性曲线（sensitivity curve）。得此名的原因在于曲线上各点反映着相同的感受性，它们都是对同一信号刺激的反应，只不过是在几种不同的判定标准下所得的结果而已。接受者操作特性曲线就是以虚惊概率为横轴，击中概率为纵轴所组成的坐标图，和被试在特定刺激条件下由于采用不同的判断标准得出的不同结果画出的曲线。

****ROC曲线的属性****

（1）β值的改变独立于d’的变化，考察β值变化对P(y/SN)和P(y/N)的影响时发现：当β接近0时，击中概率几乎为0，即信号全当成噪音接受；当β接近无穷大时，虚惊概率几乎为0，即噪音全当成信号接受；而当β从接近0向无穷大渐变的过程中，将形成一条完整地ROC曲线，曲线在某一处达到最佳的标准βOPT。（2）ROC曲线的曲率反应敏感性指标d’：对角线，代表P(y/SN)=P(y/N)，即被试者的辨别力d’为0，ROC曲线离这条线愈远，表示被试者辨别力愈强，d’的值当然就愈大。由上可知，d’的变化使ROC曲线形成一个曲线簇，而β的变化体现在这一曲线簇中的某一条曲线上不同点的变化。此外，如果将ROC曲线的坐标轴变为Z分数坐标，我们将看到ROC曲线从曲线形态变为直线形态。这种坐标变换可以用来验证信号检测论一个重要假设，即方差齐性假设。

ROC曲线是显示Classification模型真正率和假正率之间折中的一种图形化方法。

解读ROC图的一些概念定义:：

****真正（True Positive , TP）****被模型预测为正的正样本；  
****假负（False Negative , FN）****被模型预测为负的正样本；  
****假正（False Positive , FP）****被模型预测为正的负样本；  
****真负（True Negative , TN）****被模型预测为负的负样本。

****真正率****（True Positive Rate , TPR）或灵敏度（sensitivity）  
TPR = TP /（TP + FN）  （正样本预测结果数 / 正样本实际数）  
****假负率****（False Negative Rate , FNR）  
FNR = FN /（TP + FN） （被预测为负的正样本结果数 / 正样本实际数 ）

****假正率****（False Positive Rate , FPR）  
FPR = FP /（FP + TN） （被预测为正的负样本结果数 /负样本实际数）  
****真负率****（True Negative Rate , TNR）或特指度（specificity）  
TNR = TN /（TN + FP） （负样本预测结果数 / 负样本实际数）

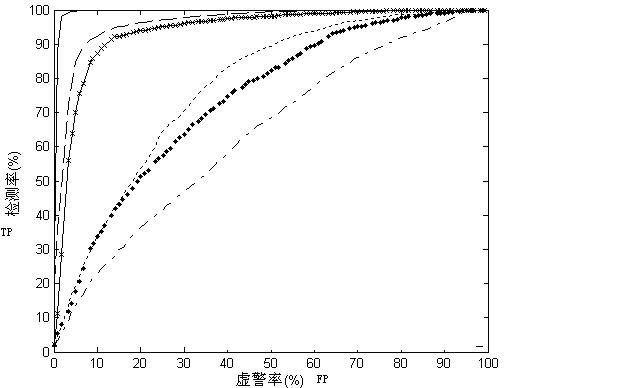
目标属性的被选中的那个期望值称作是“正”（positive）

ROC曲线上几个关键点的解释：

( TPR=0,FPR=0 ) 把每个实例都预测为负类的模型  
( TPR=1,FPR=1 ) 把每个实例都预测为正类的模型  
( TPR=1,FPR=0 ) 理想模型

一个好的分类模型应该尽可能靠近图形的左上角，而一个随机猜测模型应位于连接点（TPR=0,FPR=0）和（TPR=1,FPR=1）的主对角线上。

ROC曲线下方的面积（Area Under the ROC Curve, AUC）提供了评价模型平均性能的另一种方法。如果模型是完美的，那么它的AUC = 1，如果模型是个简单的随机猜测模型，那么它的AUC = 0.5，如果一个模型好于另一个，则它的曲线下方面积相对较大

[](http://photo.blog.sina.com.cn/showpic.html#blogid=493b40e10100jps5&url=http://s13.sinaimg.cn/orignal/493b40e1t8b0049b73f9c)

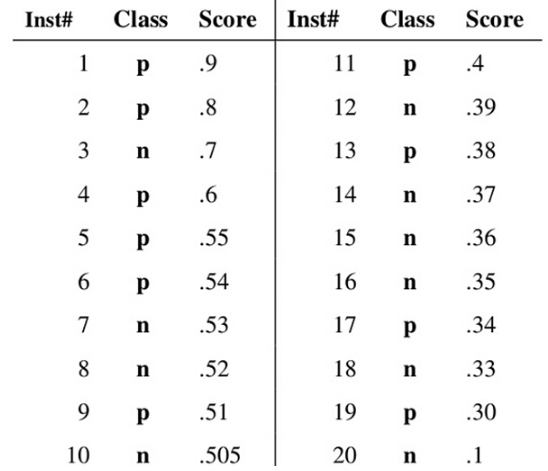
正样本：属于某一类的样本

负样本：不属于某一类的样本

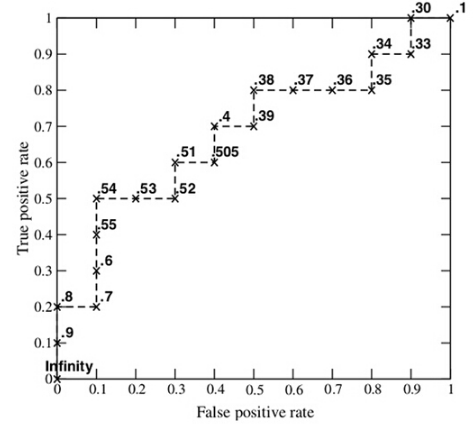
   假设采用逻辑回归分类器，其给出针对每个实例为正类的概率，那么通过设定一个阈值如0.6，概率大于等于0.6的为正类，小于0.6的为负类。对应的就可以算出一组(FPR,TPR),在平面中得到对应坐标点。随着阈值的逐渐减小，越来越多的实例被划分为正类，但是这些正类中同样也掺杂着真正的负实例，即TPR和FPR会同时增大。阈值最大时，对应坐标点为(0,0),阈值最小时，对应坐标点(1,1)。

**二 如何画**roc**曲线**

假设已经得出一系列样本被划分为正类的概率，然后按照大小排序，下图是一个示例，图中共有20个测试样本，“Class”一栏表示每个测试样本真正的标签（p表示正样本，n表示负样本），“Score”表示每个测试样本属于正样本的概率。



接下来，我们从高到低，依次将“Score”值作为阈值threshold，当测试样本属于正样本的概率大于或等于这个threshold时，我们认为它为正样本，否则为负样本。举例来说，对于图中的第4个样本，其“Score”值为0.6，那么样本1，2，3，4都被认为是正样本，因为它们的“Score”值都大于等于0.6，而其他样本则都认为是负样本。每次选取一个不同的threshold，我们就可以得到一组FPR和TPR，即ROC曲线上的一点。这样一来，我们一共得到了20组FPR和TPR的值，将它们画在ROC曲线的结果如下图：



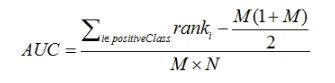
AUC(Area under Curve)：Roc曲线下的面积，介于0.1和1之间。Auc作为数值可以直观的评价分类器的好坏，值越大越好。

**首先AUC值是一个概率值，当你随机挑选一个正样本以及负样本，当前的分类算法根据计算得到的Score值将这个正样本排在负样本前面的概率就是AUC值，AUC值越大，当前分类算法越有可能将正样本排在负样本前面，从而能够更好地分类。**

**二、AUC计算**

 1.  最直观的，根据AUC这个名称，我们知道，计算出ROC曲线下面的面积，就是AUC的值。事实上，这也是在早期 Machine Learning文献中常见的AUC计算方法。由于我们的测试样本是有限的。我们得到的AUC曲线必然是一个阶梯状的。因此，计算的AUC也就是这些阶梯 下面的面积之和。这样，我们先把score排序(假设score越大，此样本属于正类的概率越大)，然后一边扫描就可以得到我们想要的AUC。但是，这么 做有个缺点，就是当多个测试样本的score相等的时候，我们调整一下阈值，得到的不是曲线一个阶梯往上或者往右的延展，而是斜着向上形成一个梯形。此 时，我们就需要计算这个梯形的面积。由此，我们可以看到，用这种方法计算AUC实际上是比较麻烦的。

   2. 一个关于AUC的很有趣的性质是，它和Wilcoxon-Mann-Witney Test是等价的。这个等价关系的证明留在下篇帖子中给出。而Wilcoxon-Mann-Witney Test就是测试任意给一个正类样本和一个负类样本，正类样本的score有多大的概率大于负类样本的score。有了这个定义，我们就得到了另外一中计 算AUC的办法：得到这个概率。我们知道，在有限样本中我们常用的得到概率的办法就是通过频率来估计之。这种估计随着样本规模的扩大而逐渐逼近真实值。这和上面的方法中，样本数越多，计算的AUC越准确类似，也和计算积分的时候，小区间划分的越细，计算的越准确是同样的道理。具体来说就是统计一下所有的 M×N(M为正类样本的数目，N为负类样本的数目)个正负样本对中，有多少个组中的正样本的score大于负样本的score。当二元组中正负样本的 score相等的时候，按照0.5计算。然后除以MN。实现这个方法的复杂度为O(n^2)。n为样本数（即n=M+N）   
   3.  第三种方法实际上和上述第二种方法是一样的，但是复杂度减小了。它也是首先对score从大到小排序，然后令最大score对应的sample 的rank为n，第二大score对应sample的rank为n-1，以此类推。然后把所有的正类样本的rank相加，再减去M-1种两个正样本组合的情况。得到的就是所有的样本中有多少对正类样本的score大于负类样本的score。然后再除以M×N。即



      公式解释：

        1、为了求的组合中正样本的score值大于负样本，如果所有的正样本score值都是大于负样本的，那么第一位与任意的进行组合score值都要大，我们取它的rank值为n，但是n-1中有M-1是正样例和正样例的组合这种是不在统计范围内的（为计算方便我们取n组，相应的不符合的有M个），所以要减掉，那么同理排在第二位的n-1，会有M-1个是不满足的，依次类推，故得到后面的公式M\*(M+1)/2，我们可以验证在正样本score都大于负样本的假设下，AUC的值为1

      2、根据上面的解释，不难得出，rank的值代表的是能够产生score前大后小的这样的组合数，但是这里包含了（正，正）的情况，所以要减去这样的组（即排在它后面正例的个数），即可得到上面的公式

      另外，特别需要注意的是，再存在score相等的情况时，对相等score的样本，需要 赋予相同的rank(无论这个相等的score是出现在同类样本还是不同类的样本之间，都需要这样处理)。具体操作就是再把所有这些score相等的样本 的rank取平均。然后再使用上述公式。

   一个分类器最主要的评测指标就是查准率（正确率）和查全率（召回率）。为了评价二分分类问题的性能，先做以下约定：

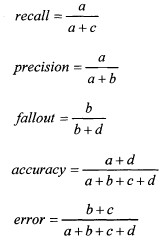
       a:正例测试文档被正确分类为该类的数量；

       b:负例测试文档被错误分类为属于该类的数量；

       c:正例测试文档被错误分类为不属于该类的数量；

       d：负例测试文档被正确分类为不属于该类的数量；

      基于上面四个值，就可以定义下面的一些评价指标：

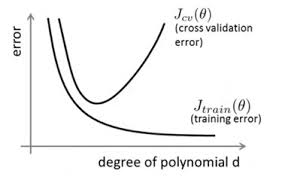


其中，查准率和查全率反映了分类器分类性能的两个方面。如果综合考虑查准率与查全率，可以得到新的评价指标F1测试值，也称为综合分类率：

IMG_256

## 正则化方法：防止过拟合，提高泛化能力

在训练数据不够多时，或者overtraining时，常常会导致overfitting（过拟合）。其直观的表现如下图所示，随着训练过程，网络在training data上的error渐渐减小，但是在验证集上的error却反而渐渐增大——因为训练出来的网络过拟合了训练集，对训练集外的数据却不work。

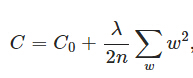


为了防止overfitting，可以用的方法有很多，下文就将以此展开。有一个概念需要先说明，在机器学习算法中，我们常常将原始数据集分为三部分：training data、validation data，testing data。这个validation data是什么？它其实就是用来避免过拟合的，在训练过程中，我们通常用它来确定一些超参数（比如根据validation data上的accuracy来确定early stopping的epoch大小、根据validation data确定learning rate等等）。那为啥不直接在testing data上做这些呢？因为如果在testing data做这些，那么随着训练的进行，我们的网络实际上就是在一点一点地overfitting我们的testing data，导致最后得到的testing accuracy没有任何参考意义。因此，training data的作用是计算梯度更新权重，validation data如上所述，testing data则给出一个accuracy以判断网络的好坏。

避免过拟合的方法有很多：early stopping、数据集扩增（Data augmentation）、正则化（Regularization）包括L1、L2（L2 regularization也叫weight decay），dropout。

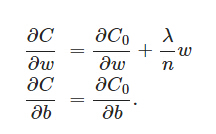
## L2 regularization（权重衰减）(L2范数)

L2正则化就是在代价函数后面再加上一个正则化项：

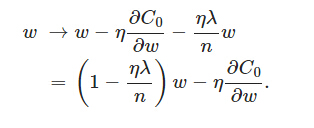


C0代表原始的代价函数，后面那一项就是L2正则化项，它是这样来的：所有参数w的平方的和，除以训练集的样本大小n。λ就是正则项系数，权衡正则项与C0项的比重。另外还有一个系数1/2，1/2经常会看到，主要是为了后面求导的结果方便，后面那一项求导会产生一个2，与1/2相乘刚好凑整。

L2正则化项是怎么避免overfitting的呢？我们推导一下看看，先求导：

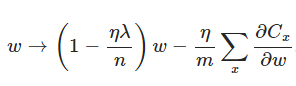


可以发现L2正则化项对b的更新没有影响，但是对于w的更新有影响:



在不使用L2正则化时，求导结果中w前系数为1，现在w前面系数为 1?ηλ/n ，因为η、λ、n都是正的，所以 1?ηλ/n小于1，它的效果是减小w，这也就是权重衰减（weight decay）的由来。当然考虑到后面的导数项，w最终的值可能增大也可能减小。

另外，需要提一下，对于基于mini-batch的随机梯度下降，w和b更新的公式跟上面给出的有点不同：



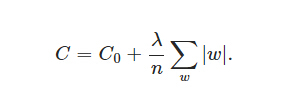
IMG_260

对比上面w的更新公式，可以发现后面那一项变了，变成所有导数加和，乘以η再除以m，m是一个mini-batch中样本的个数。

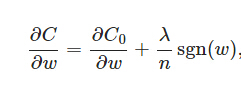
到目前为止，我们只是解释了L2正则化项有让w“变小”的效果，但是还没解释为什么w“变小”可以防止overfitting？人们普遍认为：更小的权值w，从某种意义上说，表示网络的复杂度更低，对数据的拟合刚刚好（这个法则也叫做奥卡姆剃刀）。而在实际应用中，也验证了这一点，L2正则化的效果往往好于未经正则化的效果。

## L1 regularization(L1范数)

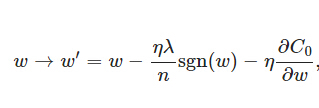
在原始的代价函数后面加上一个L1正则化项，即所有权重w的绝对值的和，乘以λ/n（这里不像L2正则化项那样，需要再乘以1/2，具体原因上面已经说过。）



同样先计算导数：



上式中sgn(w)表示w的符号。那么权重w的更新规则为：



比原始的更新规则多出了η \* λ \* sgn(w)/n这一项。当w为正时，更新后的w变小。当w为负时，更新后的w变大——因此它的效果就是让w往0靠，使网络中的权重尽可能为0，也就相当于减小了网络复杂度，防止过拟合。

另外，上面没有提到一个问题，当w为0时怎么办？当w等于0时，|W|是不可导的，所以我们只能按照原始的未经正则化的方法去更新w，这就相当于去掉η\*λ\*sgn(w)/n这一项，所以我们可以规定sgn(0)=0，这样就把w=0的情况也统一进来了。（在编程的时候，令sgn(0)=0,sgn(w>0)=1,sgn(w<0)=-1）

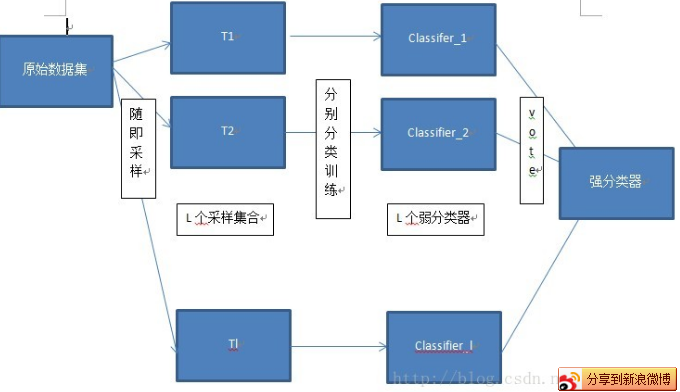
**集成学习：bagging and boosting**

bagging，boosting都是一种思想，Gradient Boosting是一种实现Boosting的方法，random forest是一种实现bagging的方法

**bagging ：**

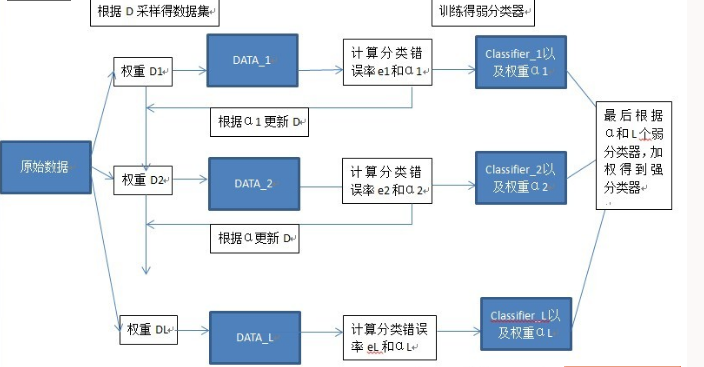
原始数据集通过T次有放回的随机采样，得到T个与原始数据集相同大小的子数据集，分别训练得到T个弱分类器Classifier，然后结合为一个强分类器。采用的是概率论里面的booststrap思想，由于小样本估计的不准确性，再加上现代计算性能的提升，可以用重复的计算提升小样本的精度。原始小样本不能正确反映数据的真实分布，用T次随机采样拟合真实分布。如果原始数据为真实分布的前提下，用bagging集成分类器，始终是能提升效果的，提升的效果取决于分类器的稳定性，稳定性越差，提升的效果越高。如神经网络这样的不稳定分类器。当然，上面假设是数据接近真实分布，然后在概率[1/N,1/N,.....1/N]下重采样。如果训练数据不是真实分布，那么bagging的效果也可能比非bagging更差。

接下来是如何把L个弱分类器集成为强分类器：最简单的方法就是投票法（vote）。对于一个测试样本，通过L个弱分类器得到L个类别信息，这些信息投票产生最后的类别。如L=10，分类结果分别为：[3,3,3,3，5,5,6,7,1,8.]那么这个样本就属于3.



**Boosting**

boosting也是通过重采样得到多个弱分类器，最后得到一个强分类器。区别是boosting是基于权值的弱分类器集成。更新采样权值D和计算分类器权重α，前者使得原来分错的样本再下一个分类器中能够有较大的几率出现，从而提高原来分错样本之后分对的概率；后者根据分类器的表现，赋予不同弱分类器不同权值，最后得到一个加权的强分类器。不断地迭代更新能使得最终的结果无限接近最优分类，不过boosting会倾向于一直分错的样本，如果样本中有离群的错误样本，boosting就会出现效果不好的情况。



**Bootstraping思想**

名字来自成语“pull up by your own bootstraps”，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法，它是非参数统计中一种重要的估计统计量方差进而进行区间估计的统计方法。其核心思想和基本步骤如下：

（1）采用重抽样技术从原始样本中抽取一定数量（自己给定）的样本，此过程允许重复抽样。

（2）根据抽出的样本计算给定的统计量T。

（3）重复上述N次（一般大于1000），得到N个统计量T。

（4）计算上述N个统计量T的样本方差，得到统计量的方差。

应该说Bootstrap是现代统计学较为流行的一种统计方法，在小样本时效果很好。通过方差的估计可以构造置信区间等，其运用范围得到进一步延伸。

**Bagging与Boosting的区别**：

二者的主要区别是取样方式不同。Bagging采用均匀取样，而Boosting根据错误率来取样，因此Boosting的分类精度要优于Bagging。Bagging的训练集的选择是随机的，各轮训练集之间相互独立，而Boostlng的各轮训练集的选择与前面各轮的学习结果有关；Bagging的各个预测函数没有权重，而Boosting是有权重的；Bagging的各个预测函数可以并行生成，而Boosting的各个预测函数只能顺序生成。对于象神经网络这样极为耗 时的学习方法。Bagging可通过并行训练节省大量时间开销。

bagging和boosting都可以有效地提高分类的准确性。在大多数数据集中，boosting的准确性比bagging高。在有些数据集中，boosting会引起退化--- Overfit。

Boosting思想的一种改进型AdaBoost方法在邮件过滤、文本分类方面都有很好的性能

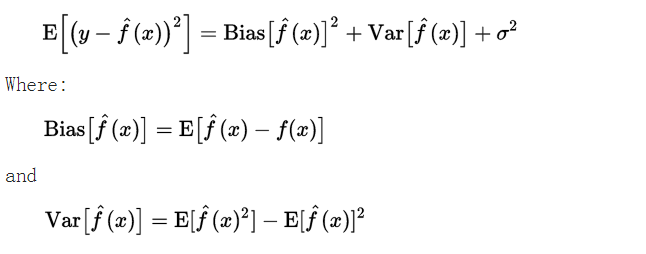
**Validation curve&&Learning curve**

**bias–variance**

**Bias（偏差）是由于对于学习算法的错误假设而导致的错误。高偏差会导致算法对于特征和目标输出之间的相关性丢失，也就是欠拟合**

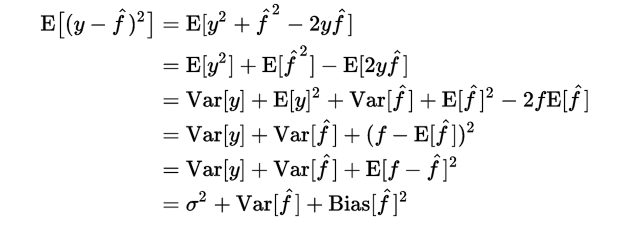
**Variance（方差）是由于对训练集的微小变化敏感造成的误差，高方差会造成过拟合，**

**基于均方误差（squared error）的bias–variance分解**



：irreducible error {\displaystyle \sigma ^{2}}IMG_256，比如噪声等

公式推导；



一个模型的偏差如果很小，一般来说都会很复杂（例如多项式回归模型），它可以使模型对于训练集的表示很精确，但是这样在处理的是时候也会把大量的噪声表示出来，引入了大量的噪声，这样对于预测就很不准确，而如果偏差很大，它虽然使得模型相对很简单，但是这样会欠拟合，这样使得预测的结果的方差很低

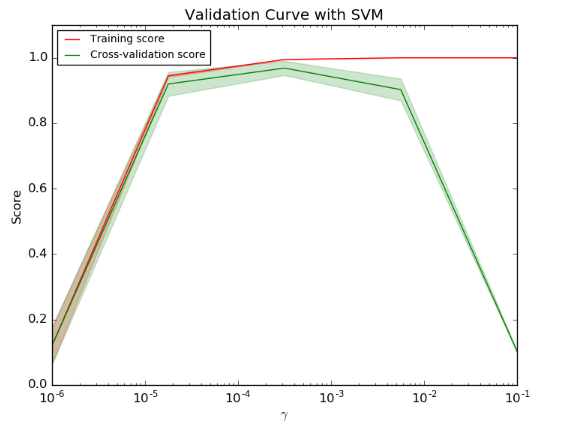
The ****bias**** **of an estimator is its average error for different training sets. The **variance** of an estimator indicates how sensitive it is to varying training sets. Noise is a property** of the data

偏差和方差是模型的特有属性，所以我们通常会去选择偏差和方差都尽可能小的学习算法模型或者模型的超参，降低方法的一个方法是使用更多的训练数据。在简单的低维度（比如二维）的问题上，我们很容易通过可视化的方法画出来这样就可以看出模型好坏相对于偏差和方差的关系，但是对于高维度的问题，就不好可视化了，基于这个原因，就可以用**Validation curves&&Learning curve和观察模型的相关情况**

**Validation curve：判断模型是否欠拟合或者过拟合**

**评估一个模型，我们需要一个评估函数，例如对于分类器的准确性，选择模型多个超参的适当的方法一般是**grid search或者其他类似的，选择基于一个或多个评估数据集上的最大分数的那个超参，但是有时还是需要看看这些超参在训练分数和评估分数的影响，从而发现模型是否欠拟合或者过拟合

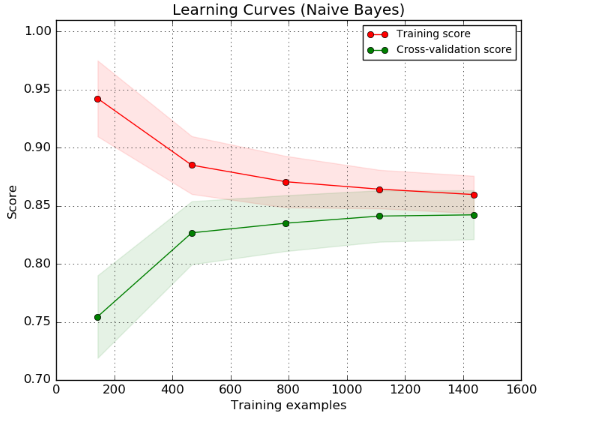
如果训练分数和评估分数都很低，这个模型就是欠拟合，如果这个训练分数很高但是评估分数很低，模型就是过拟合，训练分数很低，评估分数很高通常是不可能的



SVM模型的训练分数和评估分数随着参数 IMG_256的改变的变化情况

## **Learning curve**

Learning curve描述了模型的评估分数和训练分数对于训练样本的数目的变化情况。它可以帮助我们弄清我们增加更多的数据模型所能获得的好处。如果评估分数和训练分数随着增大训练集的样本大小趋于一直的那个值较小，那么我们就认为模型即使增加样本也没有改善，就不用再增加样本，下图趋于一个较低值



如果训练分数在一个很大的训练数据数目上还比评估分数大很多，增加更多的训练样本就很有可能增加模型的泛化能力，下图这个模型就可以增加训练样本来提高泛化能力。

