本篇主要是对第二章剩余知识的理解，包括：性能度量、比较检验和偏差与方差。在上一篇中，我们解决了评估学习器泛化性能的方法，即用测试集的“测试误差”作为“泛化误差”的近似，当我们划分好训练/测试集后，那如何计算“测试误差”呢？这就是性能度量，例如：均方差，错误率等，即“测试误差”的一个评价标准。有了评估方法和性能度量，就可以计算出学习器的“测试误差”，但由于“测试误差”受到很多因素的影响，例如：算法随机性或测试集本身的选择，那如何对两个或多个学习器的性能度量结果做比较呢？这就是比较检验。最后

偏差与方差是解释学习器泛化性能的一种重要工具。写到后面发现冗长之后读

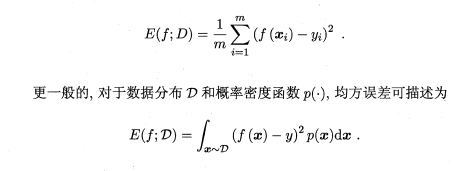
起来十分没有快感，故本篇主要知识点为性能度量。

**2.5 性能度量**

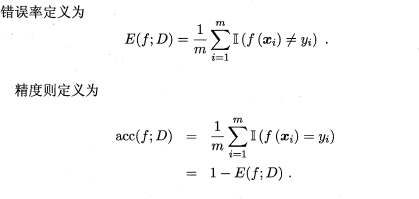
性能度量（performance measure）是衡量模型泛化能力的评价标准，在对比不同模型的能力时，使用不同的性能度量往往会导致不同的评判结果。本节除2.5.1外，其它主要介绍分类模型的性能度量。

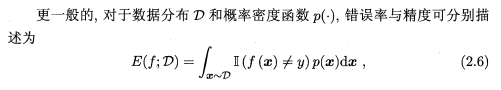
**2.5.1 最常见的性能度量**

在回归任务中，即预测连续值的问题，最常用的性能度量是“均方误差”（mean squared error）,很多的经典算法都是采用了MSE作为评价函数，想必大家都十分熟悉。

[](https://camo.githubusercontent.com/396be8e7f6ffff8ef97f1e959a05d361e133704a/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616637363237362e706e67)

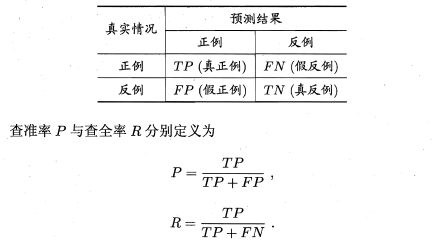
在分类任务中，即预测离散值的问题，最常用的是错误率和精度，错误率是分类错误的样本数占样本总数的比例，精度则是分类正确的样本数占样本总数的比例，易知：错误率+精度=1。

[](https://camo.githubusercontent.com/2fb6a10488aeb059f6341a3ed98945015100ff53/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616634633730342e706e67)

[](https://camo.githubusercontent.com/e1500fb19a55666d2ab66dc5d5d307d12d892cf7/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616636666238342e706e67)

**2.5.2 查准率/查全率/F1**

错误率和精度虽然常用，但不能满足所有的需求，例如：在推荐系统中，我们只关心推送给用户的内容用户是否感兴趣（即查准率），或者说所有用户感兴趣的内容我们推送出来了多少（即查全率）。因此，使用查准/查全率更适合描述这类问题。对于二分类问题，分类结果混淆矩阵与查准/查全率定义如下：

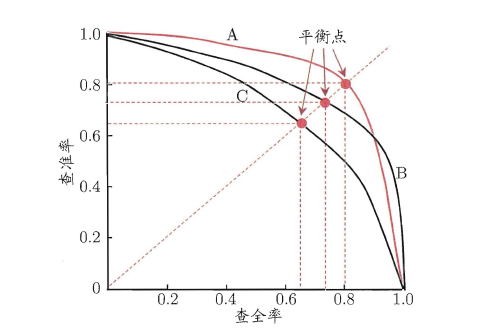
[](https://camo.githubusercontent.com/488d729ecaf09364100d7ac8e3d817459bb25beb/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616638383561342e706e67)

初次接触时，FN与FP很难正确的理解，按照惯性思维容易把FN理解成：False->Negtive，即将错的预测为错的，这样FN和TN就反了，后来找到一张图，描述得很详细，为方便理解，把这张图也贴在了下边：

[](https://camo.githubusercontent.com/10cded0a192b054829de87e70b389d0986bff49e/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616638373161362e706e67)

正如天下没有免费的午餐，查准率和查全率是一对矛盾的度量。例如我们想让推送的内容尽可能用户全都感兴趣，那只能推送我们把握高的内容，这样就漏掉了一些用户感兴趣的内容，查全率就低了；如果想让用户感兴趣的内容都被推送，那只有将所有内容都推送上，宁可错杀一千，不可放过一个，这样查准率就很低了。

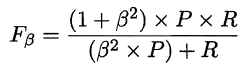
“P-R曲线”正是描述查准/查全率变化的曲线，P-R曲线定义如下：根据学习器的预测结果（一般为一个实值或概率）对测试样本进行排序，将最可能是“正例”的样本排在前面，最不可能是“正例”的排在后面，按此顺序逐个把样本作为“正例”进行预测，每次计算出当前的P值和R值，如下图所示：

[](https://camo.githubusercontent.com/89c709b82bf20efde039fe32fff5724ec6bbe6d9/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616663343431312e706e67)

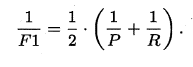
P-R曲线如何评估呢？若一个学习器A的P-R曲线被另一个学习器B的P-R曲线完全包住，则称：B的性能优于A。若A和B的曲线发生了交叉，则谁的曲线下的面积大，谁的性能更优。但一般来说，曲线下的面积是很难进行估算的，所以衍生出了“平衡点”（Break-Event Point，简称BEP），即当P=R时的取值，平衡点的取值越高，性能更优。

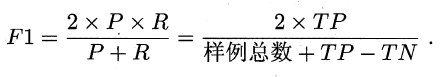
P和R指标有时会出现矛盾的情况，这样就需要综合考虑他们，最常见的方法就是F-Measure，又称F-Score。F-Measure是P和R的加权调和平均，即：

[7.png](https://camo.githubusercontent.com/ebcd21123a6cd4260a5161280bda54e3072a8dea/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616634306666362e706e67)

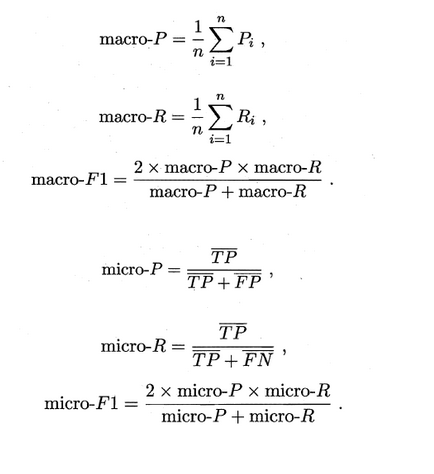
[](https://camo.githubusercontent.com/11dcfb277e54d35f6f1be05451f8d01f5f6461e7/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616637353430372e706e67)

特别地，当β=1时，也就是常见的F1度量，是P和R的调和平均，当F1较高时，模型的性能越好。

[](https://camo.githubusercontent.com/148459f6da9bf321c626e8846b4e96d977ca5904/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616632303838352e706e67)

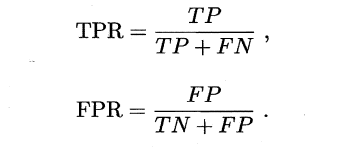
[](https://camo.githubusercontent.com/78656c0dfecbb6adf99be21a54caaba189d1fc96/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373164616634623930612e706e67)

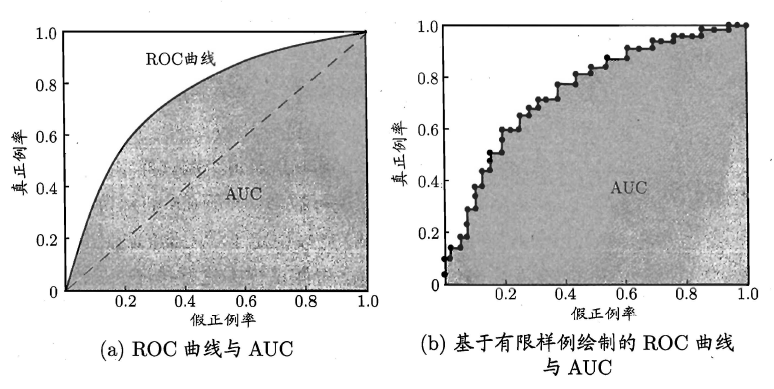
有时候我们会有多个二分类混淆矩阵，例如：多次训练或者在多个数据集上训练，那么估算全局性能的方法有两种，分为宏观和微观。简单理解，宏观就是先算出每个混淆矩阵的P值和R值，然后取得平均P值macro-P和平均R值macro-R，在算出Fβ或F1，而微观则是计算出混淆矩阵的平均TP、FP、TN、FN，接着进行计算P、R，进而求出Fβ或F1。

[](https://camo.githubusercontent.com/3b742456151a6db3a64e1a0bf7b60613030edf08/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643730323330652e706e67)

**2.5.3 ROC与AUC**

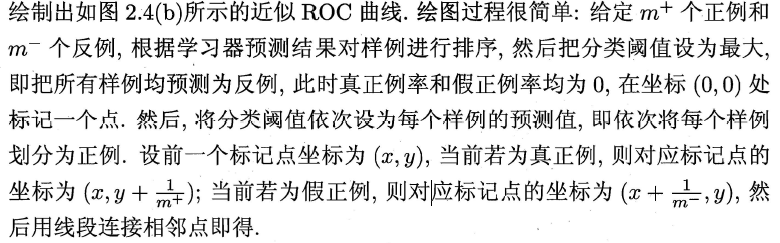
如上所述：学习器对测试样本的评估结果一般为一个实值或概率，设定一个阈值，大于阈值为正例，小于阈值为负例，因此这个实值的好坏直接决定了学习器的泛化性能，若将这些实值排序，则排序的好坏决定了学习器的性能高低。ROC曲线正是从这个角度出发来研究学习器的泛化性能，ROC曲线与P-R曲线十分类似，都是按照排序的顺序逐一按照正例预测，不同的是ROC曲线以“真正例率”（True Positive Rate，简称TPR）为横轴，纵轴为“假正例率”（False Positive Rate，简称FPR），ROC偏重研究基于测试样本评估值的排序好坏。

[](https://camo.githubusercontent.com/67b8c5b50d27078f759c30b80274affb580f6baf/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643662656539312e706e67)

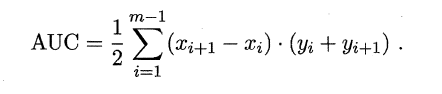
[](https://camo.githubusercontent.com/fd910cedcf687728efd0d67d938504c5ee7f3cf3/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643735636566652e706e67)

简单分析图像，可以得知：当FN=0时，TN也必须0，反之也成立，我们可以画一个队列，试着使用不同的截断点（即阈值）去分割队列，来分析曲线的形状，（0,0）表示将所有的样本预测为负例，（1,1）则表示将所有的样本预测为正例，（0,1）表示正例全部出现在负例之前的理想情况，（1,0）则表示负例全部出现在正例之前的最差情况。限于篇幅，这里不再论述。

现实中的任务通常都是有限个测试样本，因此只能绘制出近似ROC曲线。绘制方法：首先根据测试样本的评估值对测试样本排序，接着按照以下规则进行绘制。

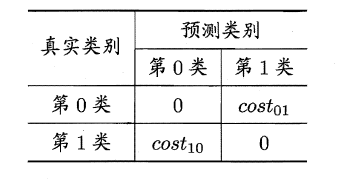
[](https://camo.githubusercontent.com/148c31d4ac16e006ed1062eb323d456a6e73c476/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643734306132342e706e67)

同样地，进行模型的性能比较时，若一个学习器A的ROC曲线被另一个学习器B的ROC曲线完全包住，则称B的性能优于A。若A和B的曲线发生了交叉，则谁的曲线下的面积大，谁的性能更优。ROC曲线下的面积定义为AUC（Area Uder ROC Curve），不同于P-R的是，这里的AUC是可估算的，即AOC曲线下每一个小矩形的面积之和。易知：AUC越大，证明排序的质量越好，AUC为1时，证明所有正例排在了负例的前面，AUC为0时，所有的负例排在了正例的前面。

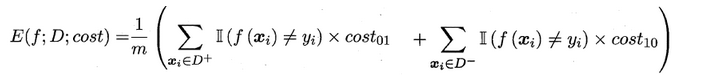
[](https://camo.githubusercontent.com/d160f4cf436d37a3de656bc4ae6a639096853528/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643665326335372e706e67)

**2.5.4 代价敏感错误率与代价曲线**

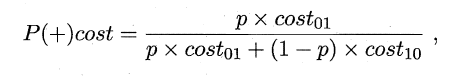
上面的方法中，将学习器的犯错同等对待，但在现实生活中，将正例预测成假例与将假例预测成正例的代价常常是不一样的，例如：将无疾病-->有疾病只是增多了检查，但有疾病-->无疾病却是增加了生命危险。以二分类为例，由此引入了“代价矩阵”（cost matrix）。

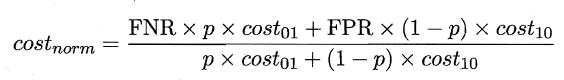
[](https://camo.githubusercontent.com/1fcdc99b6a2b6c8b56e4188034dd4111bfc4f96b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643665643538322e706e67)

在非均等错误代价下，我们希望的是最小化“总体代价”，这样“代价敏感”的错误率（2.5.1节介绍）为：

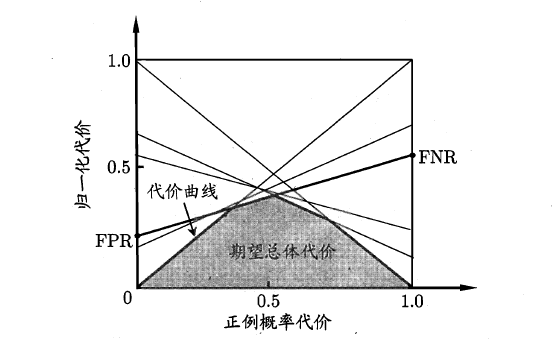
[](https://camo.githubusercontent.com/25522a5499d3d77d2e3d3a0068f75a8d8d5da86b/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643730626562652e706e67)

同样对于ROC曲线，在非均等错误代价下，演变成了“代价曲线”，代价曲线横轴是取值在[0,1]之间的正例概率代价，式中p表示正例的概率，纵轴是取值为[0,1]的归一化代价。

[](https://camo.githubusercontent.com/ef3452e60b00af400ffe8ece4a78c8b02f70968c/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643665393532652e706e67)

[](https://camo.githubusercontent.com/c3c4a4cfb9eceb9014963daf10b4dc9940422803/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643665656537622e706e67)

代价曲线的绘制很简单：设ROC曲线上一点的坐标为(TPR，FPR) ，则可相应计算出FNR，然后在代价平面上绘制一条从(0，FPR) 到(1，FNR) 的线段，线段下的面积即表示了该条件下的期望总体代价；如此将ROC 曲线土的每个点转化为代价平面上的一条线段，然后取所有线段的下界，围成的面积即为在所有条件下学习器的期望总体代价，如图所示：

[](https://camo.githubusercontent.com/181fd50db67c159c2e84a1a45c8d8a8e7480cc5c/68747470733a2f2f692e6c6f6c692e6e65742f323031382f31302f31372f356263373165643731366530642e706e67)

在此模型的性能度量方法就介绍完了，以前一直以为均方误差和精准度就可以了，现在才发现天空如此广阔~