第二章：模型评估与选择

目 录

目 录

经验误差与过拟合

- **错误率&误差：** m 个样本中有 a 个样本分类错误
  - **错误率：** 错分样本的占比：$E = a/m$

    精度：1-a/m, 即　精度=1-错误率
  - **误差：** 样本真实输出与预测输出之间的差异
    - **训练(经验)误差**：训练集上

      training error或empirical error
    - **测试误差**：测试集
    - **泛化误差**：除训练集外所有样本，新样本上的误差

经验误差与过拟合

- 我们希望泛化误差小，但由于事先并不知道新样本的特征，我们只能努力使经验误差最小化；
- 很多时候虽然能在训练集上做到分类错误率为零，精度100%，但多数情况下这样的学习器并不好
- 为了达到新样本上表现好的学习器，要学出所有潜在样本的“普遍规律”，才能遇到新样本的时候判断正确。
- 然而，当学习器把训练样本学的太好的时候，就可能把训练样本本身的一些特点当做了潜在样本都会具有的一般性质，就会导致泛化能力下降，这种现象就叫“过拟合”overfitting

经验误差与过拟合

- **过拟合：**

  学习器把训练样本学习的“太好”，将训练样本本身的特点当做所有样本的一般性质，导致泛化性能下降。

  机器学习面临的关键障碍，无法彻底避免，只能缓解

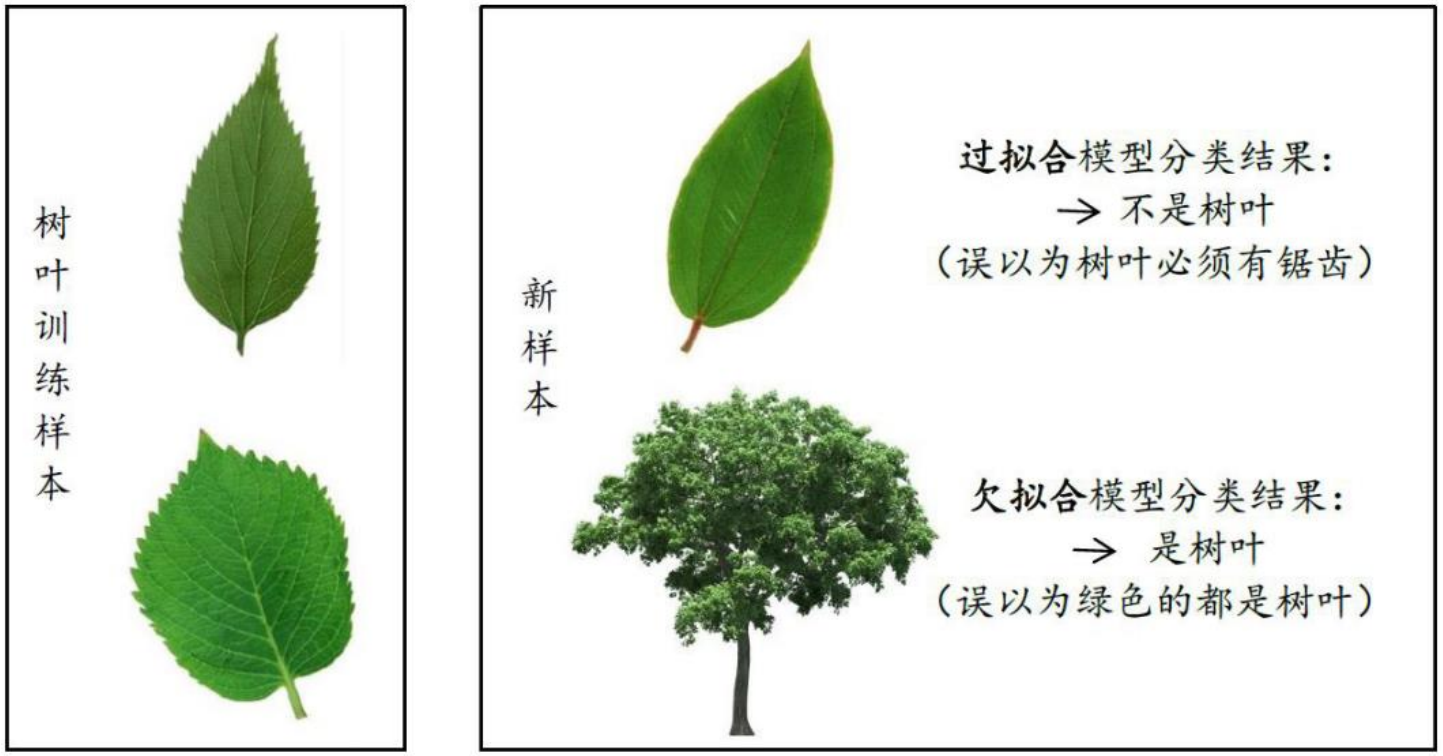
  - **优化目标加正则项**
  - early stop

- **欠拟合：**

  对训练样本的一般性质尚未学好,学习能力低下造成的，容易客服，克服方法例如

  - **决策树:拓展分支**
  - **神经网络:增加训练轮数**

经验误差与过拟合



过拟合、欠拟合的直观类比

过拟合：学习器把训练样本本身特点当做所有潜在样本都会具有的一般性质.

欠拟合：训练样本的一般性质尚未被学习器学好.

目 录

评估方法

现实任务中，往往有多种算法可选择，甚至对同一个学习算法，不同的参数配置，也会有不同的模型。如何选择？

理想方案：对候选模型的泛化误差进行评估，选择**泛化误差最小**的。

实际情况：无法直接获得泛化误差，而训练误差又由于过拟合现象的存在而不适合作为标准。

如何评估选择？

评估方法

- 现实任务中往往会对学习器的泛化性能、时间开销、存储开销、可解释性等方面的因素进行评估并做出选择。

- 我们假设：测试集是从样本真实分布中独立采样获得，

  将测试集上的“测试误差”作为泛化误差的近似，

  所以测试集要和训练集中的样本尽量互斥。

举例：

测试样本为什么要尽可能不出现在训练集中呢?

场景:老师出了10道习题练习,考试时老师又用同样的这10道题考试,这个考试成绩能否有效反映出同学们学得好不好呢?

评估方法

希望得到泛化性能强的模型，好比是希望同学们对课程学得很好，

获得了对所学知识“举一反三”的能力；

训练样本，相当于给同学们练习的习题，**测试过程**相当于是考试，

显然，若**测试样本**被用作训练了，得到的将是过于“乐观”的估计结果。

通常将包含个m样本的数据集$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$
拆分成训练集S和测试集T：

常见做法：

- 留出法：
  - 直接将数据集划分为两个互斥集合，在S上训练

训练集S,另一个作为测试集T, 即 $D = S \cup T, S \cap T = \varnothing$

用T来评估其测试误差,作为对泛化误差的估计

评估方法

- **留出法：**
  - 直接将数据集划分为两个互斥集合
  - 训练/测试集划分要**尽可能保持数据分布的一致性**
  - 一般若干次随机划分、重复实验取平均值

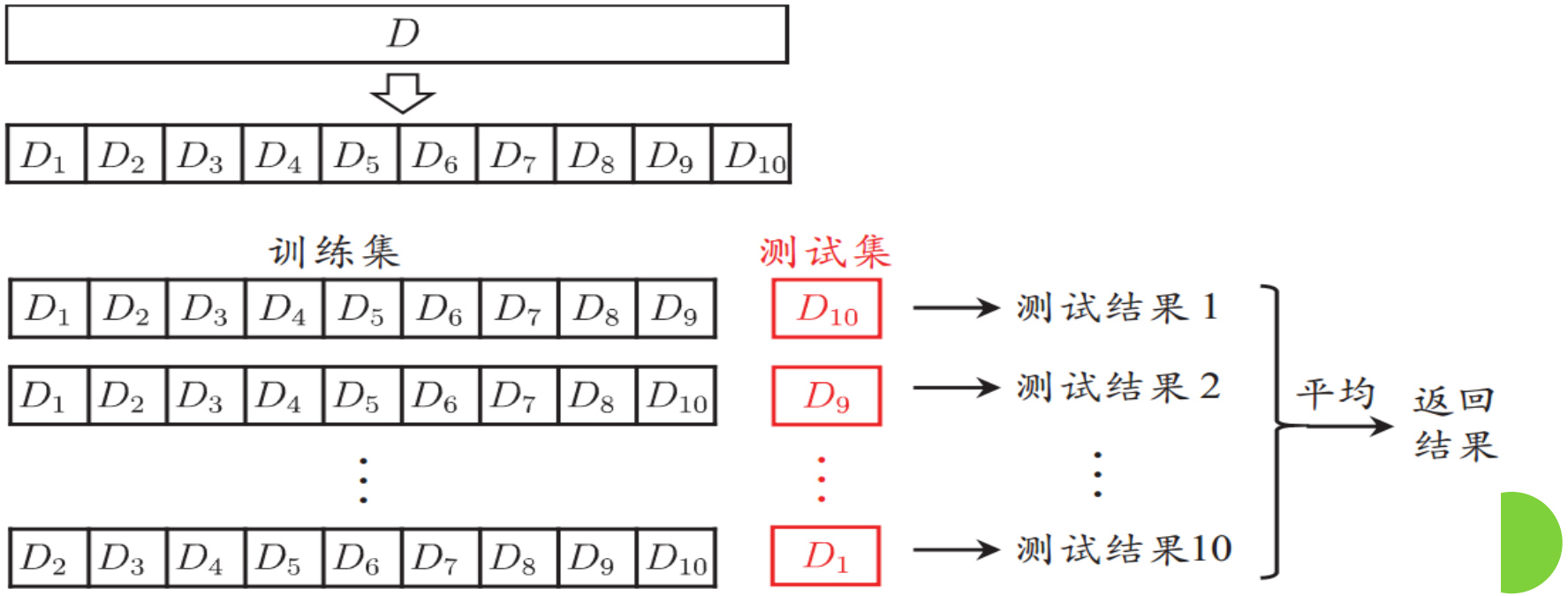
    不同分割导致不同的训练集合测试集，单次划分不准确，
  - 训练/测试样本比例通常为2:1~4:1

评估方法

- **交叉验证法：**

将数据集分层采样划分为k个大小相似的互斥子集，每次用k-1个子集的并集作为训练集，余下的子集作为测试集，最终返回k个测试结果的均值，k最常用的取值是10.

$$D = D_1 \cup D_2 \cup \ldots \cup D_k,\ D_i \cap D_j = \varnothing\ (i \neq j)$$

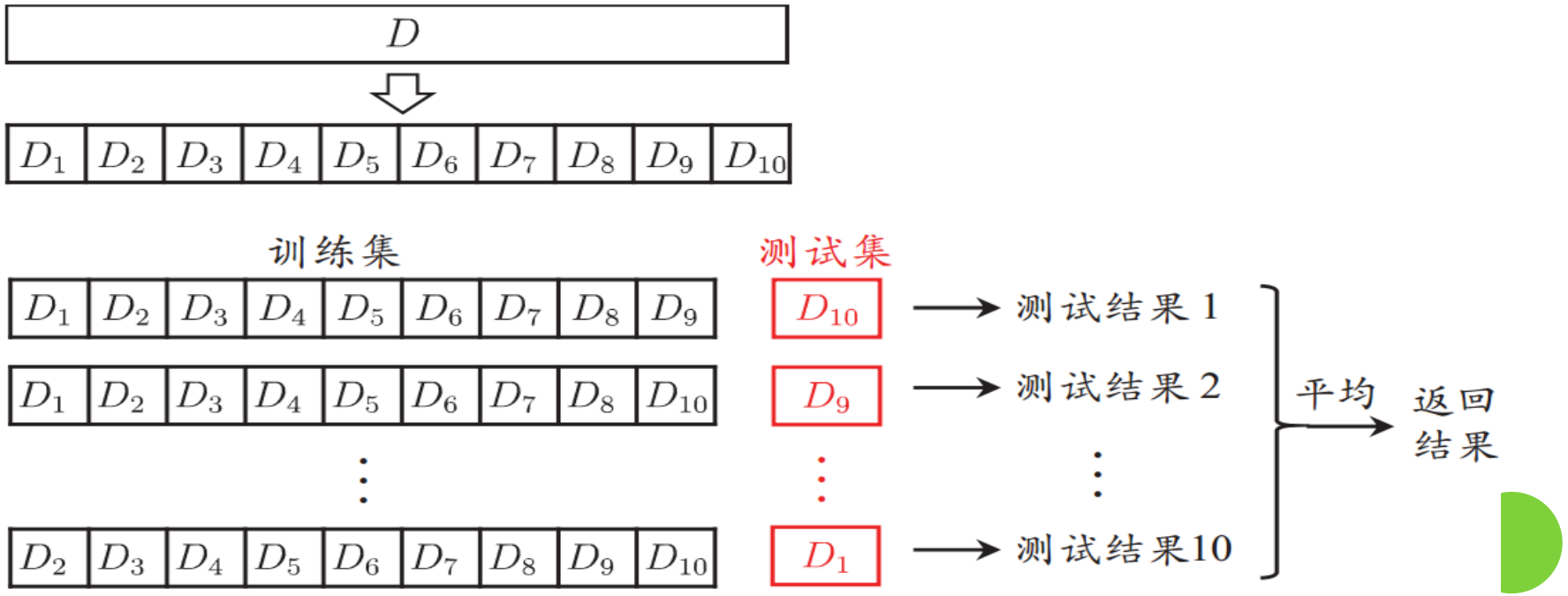


10 折交叉验证示意图

评估方法

- **交叉验证法：**

显然，交叉验证法**评估结果的稳定性和保真性在很大程度上取决于*k*的取值**，为了强调这一点，通常把交叉验证法称为“*k*折交叉验证”(*k*- fold cross validation)，*k*最常用的取值是10,此时称为10折交叉验证，其他常用的*k*值有5，20，图2.2给出了10折交叉验证的示意图



10 折交叉验证示意图

评估方法

留一法：

与留出法类似，将数据集D划分为k个子集同样存在多种划分方式，为了减小因样本划分不同而引入的差别，k折交叉验证通常随机使用不同的划分重复p次，最终的评估结果是这p次k折交叉验证结果的均值，例如常见的“10次10折交叉验证”

假设数据集D包含m个样本，若令$k = m$，则得到留一法：

- 不受随机样本划分方式的影响,m个样本只有唯一的划分方式，每个子集只有一个样本，
- 结果往往比较准确，因为训练集和初始数据集只少了一个样本
- 当数据集比较大时，计算开销难以忍受

评估方法

我们**希望评估: 用D训练出的模型**，**但是,**

留出法和交叉验证法: 由于保留了一部分样本用于测试，因此实际评估的模型所使用的训练级比D小，这必然会引入一些因训练样本规模不同而导致的估计偏差。

留一法：受训练样本规模变化的影响较小，但计算复杂度太高了。

有没有什么办法可以减少训练样本规模不同造成的影响，同时还能比较高效的进行实验估计呢？

“自助法”（bootstrapping)是一个比较好的解决方案

评估方法

“自助法”(bootstrapping)是一个比较好的解决方案, 它直接以自助采样法(bootstrap sampling)为基础 [Efron and Tibshirani, 1993]. 给定包含 m 个样本的数据集 D , 我们对它进行采样产生数据集 D': 每次随机从 D 中挑选一个样本, 将其拷贝放入 D' , 然后再将该样本放回初始数据集 D 中, 使得该样本在下次采样时仍有可能被采到; 这个过程重复执行 m 次后, 我们就得到了包含 m 个样本的数据集 D', 这就是自助采样的结果. 显然, D 中有一部分样本会在 D' 中多次出现, 而另一部分样本不出现. 可以做一个简单的估计, 样本在 m 次采样中始终不被采到的概率是$\left(1-\frac{1}{m}\right)^m$, 取极限得到

$$\lim_{m\mapsto\infty}\left(1-\frac{1}{m}\right)^m \mapsto \frac{1}{e}\approx 0.368\ , \tag{2.1}$$

即通过自助采样, 初始数据集 D 中约有 36.8% 的样本未出现在采样数据集 D' 中. 于是我们可将 D' 用作训练集, $D\setminus D'$ 用作测试集; 这样, 实际评估的模型与期望评估的模型都使用 m 个训练样本, 而我们仍有数据总量约 1/3 的、没在训练集中出现的样本用于测试. 这样的测试结果, 亦称“包外估计”(out-of-bag estimate).

评估方法

- **自助法：**

  以自助采样法为基础，对数据集D有放回采样m次得到训练集D'，$D \setminus D'$用做测试集。

  - 实际模型与预期模型都使用m个训练样本
  - 约有1/3的样本没在训练集中出现
  - 从初始数据集中产生多个不同的训练集，对集成学习有很大的好处
  - 自助法在数据集较小、难以有效划分训练/测试集时很有用；由于改变了数据集分布可能引入估计偏差，在数据量足够时，留出法和交叉验证法更常用。

目 录

性能度量

- 性能度量是衡量模型泛化能力的评价标准，
- 性能度量反映了任务需求，对比不同模型的能力时候，使用不同的性能度量往往会导致不同的评判结果，这意味着模型好坏是相对的，什么样的模型是好的坏的**不仅仅取决于算法和数据，还取决于任务需求**

在预测任务中，给定样例集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$

评估学习器的性能 f 也即把预测结果 $f(\boldsymbol{x})$ 和真实标记比较.

回归任务最常用的性能度量是“均方误差”：

$$E(f; D) = \frac{1}{m} \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2$$

性能度量

对于分类任务,**错误率和精度**是最常用的两种性能度量:

- 错误率: 分错样本 占 样本总数的比例
- 精度: 分对样本 占 样本总数的比率

分类错误率

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)$$

精度

$$\begin{aligned}\operatorname{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

性能度量

- 信息检索、Web搜索等场景中经常需要衡量：

正例被预测出来的比率或者预测出来的正例中正确的比率，

此时查准率和查全率比错误率和精度更适合。

- 统计真实标记和预测结果的组合可以得到“混淆矩阵”

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

查准率 $P = \frac{TP}{TP + FP}$

查全率 $R = \frac{TP}{TP + FN}$

$TP + FP + TN + FN$ =样例总数

性能度量

查准率 P 与查全率 R 分别定义为

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

$$P = \frac{TP}{TP + FP},$$

$$R = \frac{TP}{TP + FN}.$$

查准率和查全率是一对矛盾的度量.

一般，**查准率高时,查全率一般偏低，**

查全率高时,查准率一般偏低。

性能度量

查准率 P 与查全率 R 分别定义为

$$P = \frac{TP}{TP + FP},$$

$$R = \frac{TP}{TP + FN}.$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

查准率和查全率是一对矛盾的度量. 一般来说,

一般，**查准率高时查全率一般偏低，**

查全率高时查准率一般偏低。

例如：若希望将好瓜尽可能多地选出来，则可通过增加选瓜的数量来实现；

如果将所有西瓜都选上，那么所有的好瓜也必然都被选上了，

但这样查准率就会较低;

若希望选出的瓜中好瓜比例尽可能高，则可只挑选最有把握的瓜，

但这样就难免会漏掉不少好瓜，使得查全率较低.

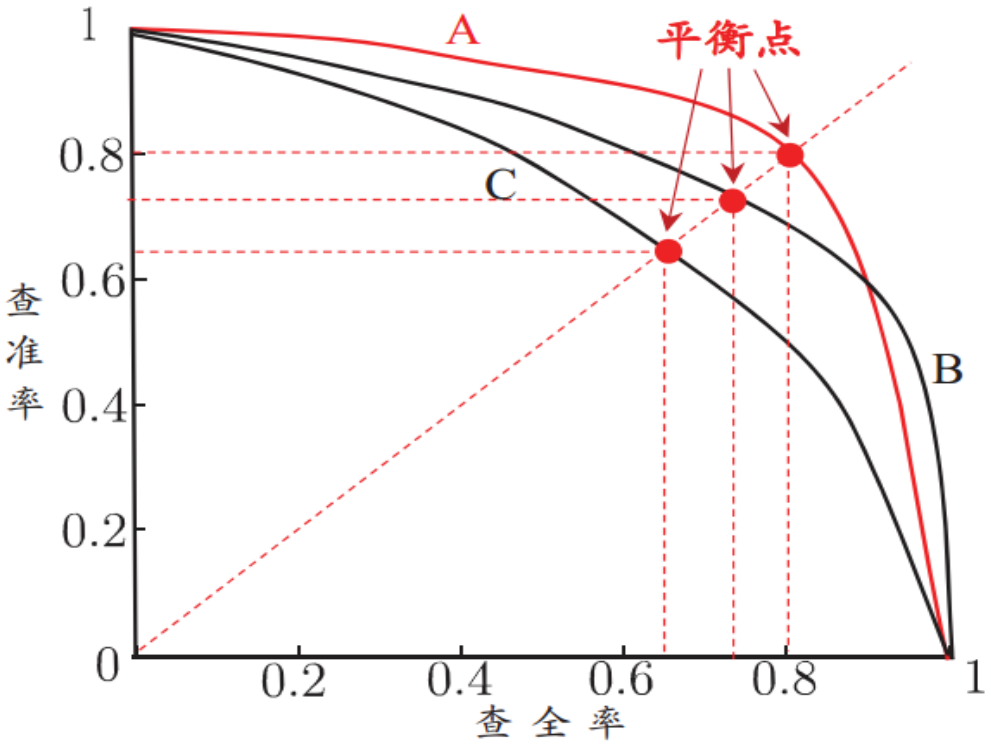
通常只有在一些简单任务中,才可能使查全率和查准率都很高.

性能度量

根据学习器的预测结果，按正例可能性大小对样例进行排序，

并逐个把样本作为正例进行预测，

则可以得到查准率-查全率曲线，简称“P-R曲线”



P-R曲线与平衡点示意图

性能度量

根据学习器的**预测结果**按**正例可能性大小**对样例进行**排序**，排在**最前面**是学习器认为“**最可能**”是正例的样本，排在**最后的**则是学习器认为“**最不可能**”是正例的样本。并逐个**把样本作为正例**进行预测，

排序

编号	Label	预测指标	预测结果	是否正确
1	真	0.98		
2	真	0.95		
3	真	0.91		
4	假	0.87		
5	真	0.85		
6	假	0.70		
7	假	0.55		
8	假	0.22		
9	假	0.13		
10	假	0.07		

性能度量

然后，按此顺序 **逐个把样本** 作为 **正例** 进行 **预测**，我们把第一个样本作为正例：

编号	Label	预测指标	预测结果	是否正确
1	真	0.98	正例	正确
2	真	0.95	反例	错误
3	真	0.91	反例	错误
4	假	0.87	反例	正确
5	真	0.85	反例	错误
6	假	0.70	反例	正确
7	假	0.55	反例	正确
8	假	0.22	反例	正确
9	假	0.13	反例	正确
10	假	0.07	反例	正确

不难看出，当我们假设 **第一个为正例** 的时候，**预测指标 > 0.95**，此时的 **查准率很高**，但 **查全率很低**。

此时的混淆矩阵为：

真实情况	预测结果	
	正例	反例
正例	1	3
反例	0	6

此时的 **查准率 P** 与 **查全率 R** 分别为：

$$P = \frac{TP}{TP + FP} = \frac{1}{1+0} = 1$$

$$R = \frac{TP}{TP + FN} = \frac{1}{3+1} = \frac{1}{4}$$

性能度量

接着，我们把前三个当作正例：

编号	Label	预测指标	预测结果	是否正确
1	真	0.98	正例	正确
2	真	0.95	正例	正确
3	真	0.91	正例	正确
4	假	0.87	反例	正确
5	真	0.85	反例	错误
6	假	0.70	反例	正确
7	假	0.55	反例	正确
8	假	0.22	反例	正确
9	假	0.13	反例	正确
10	假	0.07	反例	正确

显然，当我们假设 **前三个为正例** 的时候，

预测指标 > 0.90，此时的 **查准率很高**，

而且 **查全率也有提升**。

此时的混淆矩阵为：

真实情况	预测结果	
	正例	反例
正例	3	1
反例	0	6

此时的 **查准率 P** 与 **查全率 R** 分别为：

$$P = \frac{TP}{TP + FP} = \frac{3}{3+0} = 1$$

$$R = \frac{TP}{TP + FN} = \frac{3}{3+1} = \frac{3}{4}$$

我们 **按顺序逐个把样本** 作为 **正例** 进行 **预测**，就可以计算出当前的 **查全率、查准率**。

通过上述过程我们就可以得到 **查准率 - 查全率 的 关系曲线**：

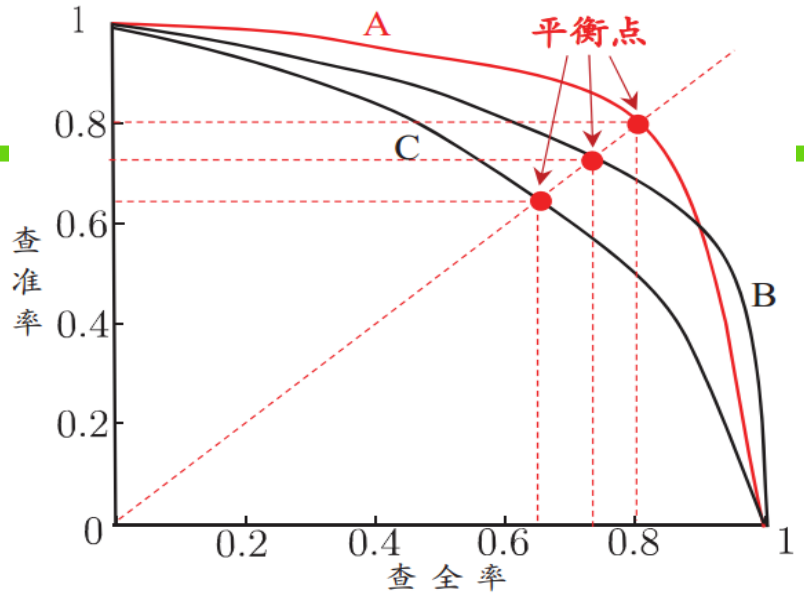
性能度量

对于 一个 学习器来说，

我们可以 选择最佳参数

对于 多个 学习器来说，

我们可以 比较学习器的优劣



P-R曲线与平衡点示意图

P-R图直观地显示出学习器在样本总体上的查全率查准率.

在进行比较时，

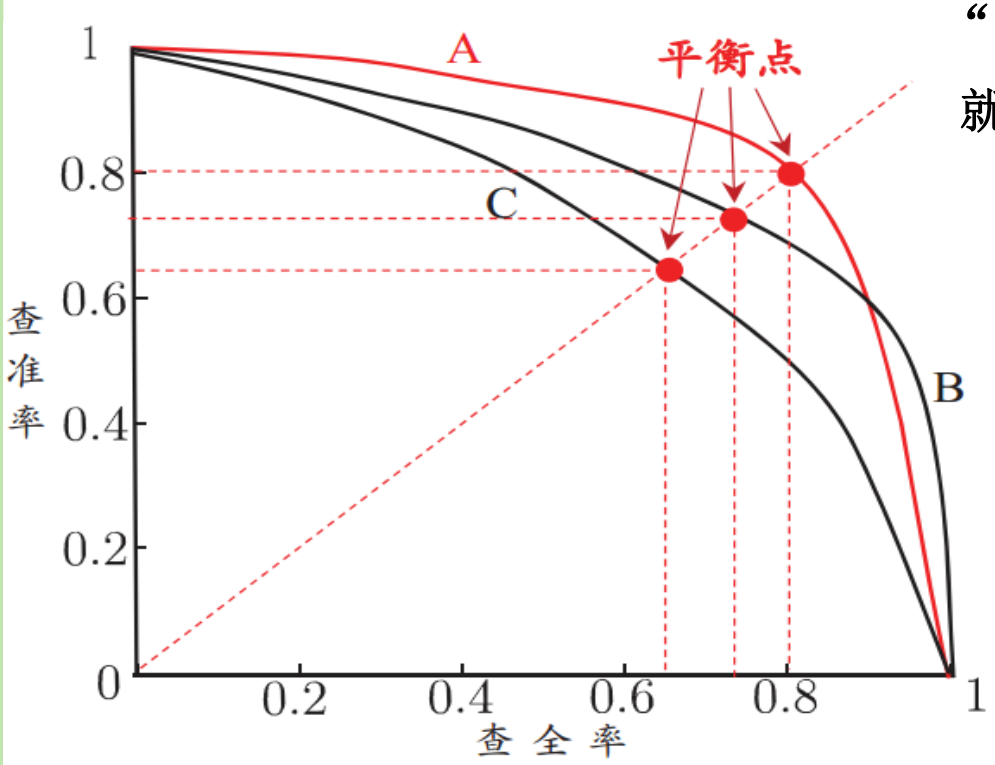
- 若一个学习器的P-R曲线被另外一个的P-R曲线完全包住，

则可断言后者的性能优于前者，如图2.3, 学习器A的性能优于学习器C;

- 如果两个学习器的P-R曲线发生了交叉，如图A和B，则难以一般性的断言两者孰优孰劣，只能在具体的查准率和查全率条件下进行比较。

然而在很多情形下，人们往往仍希望把A与B比出个高低，这时一个比较合理的依据就是：**比较P-R曲线下面积的大小**，他在一定程度上表征了学习器在查准率和查全率上取得相对双高的比例，**但这个值不太容易评估算**，因此人们设计的一套综合查考虑查准率，查询率的性能度量 ⇨ 平衡点

性能度量



P-R曲线与平衡点示意图

“平衡点”（Break-Event-Point简称BEP）就是这样一个度量

平衡点是曲线上“查准率=查全率”时的取值，可用来用于度量P-R曲线有交叉的分类器性能高低

例如图2.3中学习器C的BEP是0.64，可认为学习器A优于B

性能度量

比**P-R**曲线平衡点更用常用的是**F1**度量：

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{样例总数 + TP - TN}$$

在一些应用中，**对查准率和查全率**的重视程度有所不同。例如：

- 在商品推荐中，为了尽可能少打扰用户，更希望推荐内容确是用户感兴趣的，**此时查准率很重要；**
- **而在逃犯信息检索系统中，更希望尽可能的减少漏掉的逃犯，此时查全率很重要；**

$F1$ 是基于查准率与查全率的调和平均(harmonic mean)定义的：

$$\frac{1}{F1} = \frac{1}{2} \cdot \left(\frac{1}{P} + \frac{1}{R} \right).$$

F_β 则是加权调和平均：

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2} \cdot \left(\frac{1}{P} + \frac{\beta^2}{R} \right).$$

与算术平均($\frac{P+R}{2}$)和几何平均($\sqrt{P \times R}$)相比，调和平均更重视较小值.

$F1$ 度量的一般形式——F_β，

能让我们表达出对查准率/查询率的**不同偏好，定义为：**

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

性能度量

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

其中 $\beta > 0$ 度量了查全率对查准率的相对重要性

比F1更一般的形式 F_β ，

$$F_\beta = \frac{(1+\beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

$\beta = 1$ ： 退化为标准F1

$\beta > 1$ ： 偏重查全率(逃犯信息检索)

$\beta < 1$ ： 偏重查准率(商品推荐系统)

$F1$ 是基于查准率与查全率的调和平均(harmonic mean)定义的:

$$\frac{1}{F1} = \frac{1}{2} \cdot \left(\frac{1}{P} + \frac{1}{R}\right).$$

F_β 则是加权调和平均:

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2} \cdot \left(\frac{1}{P} + \frac{\beta^2}{R}\right).$$

与算术平均($\frac{P+R}{2}$)和几何平均($\sqrt{P \times R}$)相比, 调和平均更重视较小值.

性能度量

很多时候我们有多个二分类的混淆矩阵，例如:

- 进行多次训练或测试，每次得到一个混淆矩阵
- 或者是在多个数据集上进行测试或训练，希望估计算法的全局性能
- 或者是执行多分类任务，每两两类别的组合都对应一个混淆矩阵…

总之我们希望在n个二分类混淆矩阵上综合考察查准率和查全率

一种直接的做法是：先在各混淆矩阵上分别计算出查准率和查全率，

记为： $(P1, R1),(P2, R2),\ldots,(Pn,Rn),$

再计算平均值，这样就得到：

“宏查准率”(macro-P)、“宏查全率”(macro-R),

以及相应的“宏F1”(macro-F1):

性能度量

一种直接的做法是：先在各混淆矩阵上分别计算出查准率和查全率，

记为： *(P1, R1),(P2, R2),...,(Pn,Rn),*

再计算平均值，这样就得到：

“宏查准率”(macro-P)：

$$\text{macro-}P = \frac{1}{n}\sum_{i=1}^{n} P_i \ ,$$

“宏查全率”(macro-R)：

$$\text{macro-}R = \frac{1}{n}\sum_{i=1}^{n} R_i \ ,$$

“宏F1”(macro-F1):

$$\text{macro-}F1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R}$$

性能度量

还可先将各混淆矩阵的对应元素进行平均，得到：

TP、FP、TN、FN的平均值，分别记为 $\overline{TP}$、$\overline{FP}$、$\overline{TN}$、$\overline{FN}$

再基于这些平均值计算出：

“微查准率”(micro-P)、“微查全率”(micro-R)和“微F1”(micro-F1):

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}$$

$$\text{micro-}F1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R}$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

对应元素平均

性能度量 ROC 与 AUC

很多学习器是为测试样本产生一个实值或概率预测：

将预测值和分类值进行比较，**预测值>阈值，　正类**

预测值<阈值，　负类

如神经网络：在一般情形下对每个测试样本预测出一个[0.0,1.0]之间的实值，然后将这个值与0.5进行比较大于0.5，判断为正例，否则为反例；

预测结果的好坏直接决定了学习器的泛化能力。

实际上根据这个实值或者概率预测结果，我们可以将测试样本进行排序，“最可能”的正例排在最前面，“最不可能”是正例的排在最后面。

这样，分类过程相当于就在这个排序中以某个**“截断点”(cut point)**将样本分为两个部分：**前一部分判断是正例，后一部分则判断是反例**

性能度量 ROC 与 AUC

在不同的应用任务中，我们可根据任务需求来采用不同的截断点，例如：

若更重视“查准率”，则可选择排序中靠前的位置进行截断；

若更重视“查全率”，则可选择靠后的位置来进行截断；

因此，**排序本身的质量好坏，体现了综合考虑学习器在不同任务下的“期望泛化性能”的好坏，或者说：一般情况下泛化性能的好坏。**

ROC曲线则是从这个角度出发来研究学习器泛化性能的有利工具

性能度量 ROC与AUC

类似P-R曲线，根据学习器的预测结果**对样例排序**，**并逐个作为正例进行预测，**

以“假正例率”为横轴，“真正例率”为纵轴

可得到ROC曲线，全称“受试者工作特征”.

与P-R曲线使用查准率、查全率 为 纵、横轴不同，

ROC曲线：

纵轴是：“真正例率”(True Positive Rate,简称TPR)，

横轴是：“假正例率”(False Positive Rate,简称FPR)，

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

查准率 $P = \frac{TP}{TP + FP}$

查全率 $R = \frac{TP}{TP + FN}$

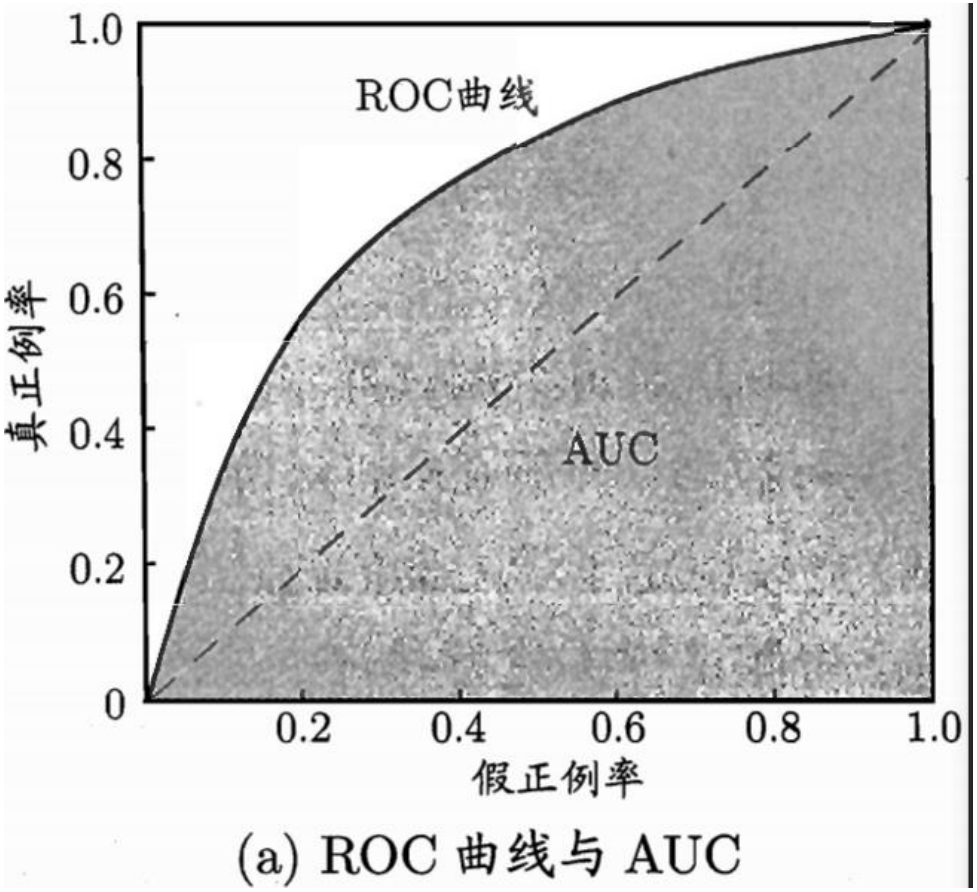
基于表2.1中的符号,两者分别定义为

$$\mathrm{TPR} = \frac{TP}{TP + FN}$$

$$\mathrm{FPR} = \frac{FP}{TN + FP}$$

性能度量

显示 ROC 曲线的图称为“ROC 图”



(a) ROC 曲线与 AUC

对角线对应于“随机猜测”模型，

而点(0,1)则对应于:

将所有正例排在所有反例之前的“理想模型”:

假正例率：0

真正例率：1

$$\mathrm{TPR} = \frac{TP}{TP + FN},$$

$$\mathrm{FPR} = \frac{FP}{TN + FP}.$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

现实任务中通常是： 利用有限个测试样例来绘制ROC图，

此时仅能获得有限个(真正例率, 假正例率)坐标对，

无法产生图2.4(a)中的光滑ROC曲线,只能**绘制出近似曲线**

性能度量

显示 ROC 曲线的图称为“ROC 图”

ROC图的绘制：给定 m^+ 个正例和 m^- 个负例，

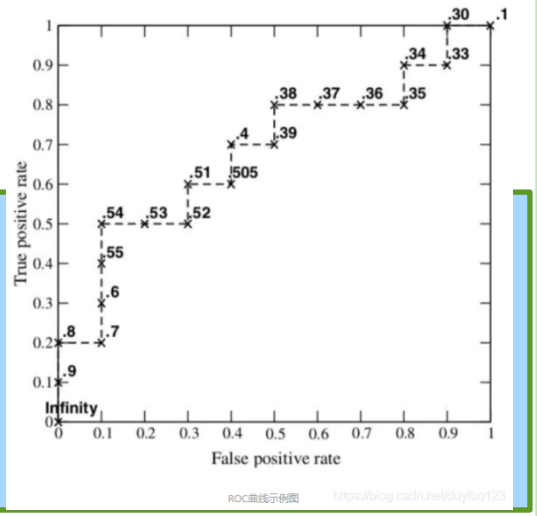
根据学习器预测结果对样例进行**排序**，

将分类阈值设为每个样例的预测值，当前标记点坐标为 (x, y)，

当前若为**真正例**，则对应标记点的坐标为 $(x, y + \frac{1}{m^+})$

当前若为**假正例**，则对应标记点的坐标为 $(x + \frac{1}{m^-}, y)$

然后用线段连接相邻点.



分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

$$\mathrm{TPR} = \frac{TP}{TP + FN},$$

$$\mathrm{FPR} = \frac{FP}{TN + FP}.$$

性能度量

显示 ROC 曲线的图称为“ROC 图”

ROC图的绘制：给定 m^+ 个正例和 m^- 个负例，

假设我们已经训练得到一个学习器 f(s)，

8 个测试样本（4 个正例，4 个反例，也即 m+ = m− = 4）进行预测，

假设预测结果为：

$$(s_1, 0.77, +), (s_2, 0.62, -), (s_3, 0.58, +), (s_4, 0.47, +), (s_5, 0.47, -), (s_6, 0.33, -), (s_7, 0.23, +), (s_8, 0.15, -)$$

其中，+ 和 − 分别表示为正例和为反例，

里面的数字表示学习器 f(s) 预测该样本为正例的概率

例如对于反例 s2 来说，当前学习器 f(s) 预测它是正例的概率为 0.62

性能度量

$$(s_1, 0.77, +), (s_2, 0.62, -), (s_3, 0.58, +), (s_4, 0.47, +), (s_5, 0.47, -), (s_6, 0.33, -), (s_7, 0.23, +), (s_8, 0.15, -)$$

ROC图的绘制方法：

- 首先：需要对所有测试样本按照学习器给出的预测结果进行排序

（上面给出的预测结果已经按照预测值从大到小排好）

- 接着将分类阈值设为一个不可能取到的最大值，

显然这时候所有样本预测为正例的概率都一定小于分类阈值，

那么预测为正例的样本个数为 0，相应的真正例率和假正例率也都为 0，

所以此时我们可以在坐标 (0,0) 处打一个点。

- 接下来我们需要把**分类阈值从大到小依次设为每个样本的预测值**，

也就是依次设为 0.770.620.580.470.330.230.15，

然后每次计算真正例率和假正例率，再在相应的坐标上打一个点

- 最后再将各个点用直线串连起来即可得到 ROC 曲线。

性能度量

$$(s_1, 0.77, +), (s_2, 0.62, -), (s_3, 0.58, +), (s_4, 0.47, +), (s_5, 0.47, -), (s_6, 0.33, -), (s_7, 0.23, +), (s_8, 0.15, -)$$

ROC图的绘制方法：

注意：在统计预测结果时，预测值等于分类阈值的样本也算作预测为正例。

例如，当分类阈值为 0.77 时，测试样本 s1 被预测为正例，

由于它的真实标记也是正例，所以此时 s1 是一个真正例。

为了便于绘图，我们将:

x 轴（假正例率轴）的“步长”定为 $\frac{1}{m^-}$

y 轴（真正例率轴）的“步长”定为 $\frac{1}{m^+}$，

这样的话，根据真正例率和假正例率的定义可知，

每次变动分类阈值时，

若新增 i 个假正例，那么相应的 x 轴坐标也就增加 $\frac{i}{m^-}$；

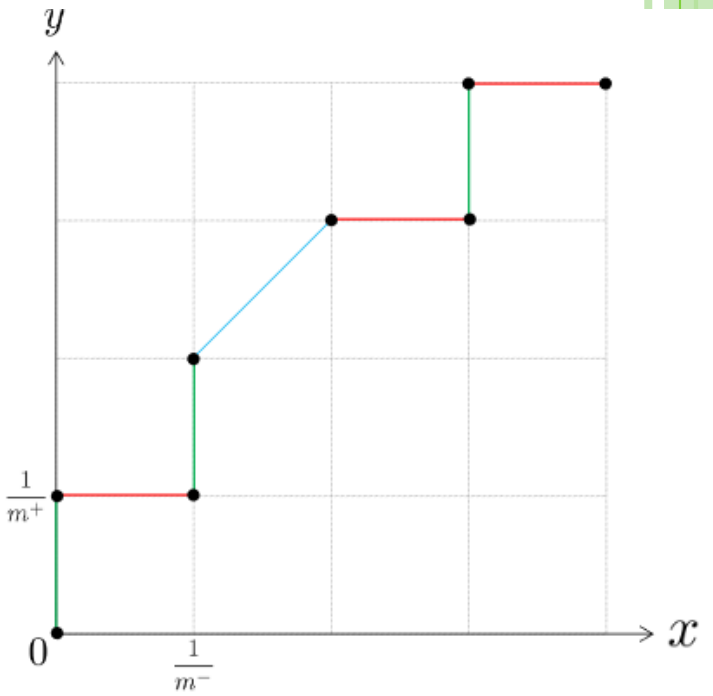
同理，若新增 j 个真正例，那么相应的 y 轴坐标也就增加 $\frac{j}{m^+}$ 。

按照以上讲述的绘制流程，最终我们可以绘制出如下图所示的 ROC 曲线

性能度量

$$(s_1, 0.77, +), (s_2, 0.62, -), (s_3, 0.58, +), (s_4, 0.47, +), (s_5, 0.47, -), (s_6, 0.33, -), (s_7, 0.23, +), (s_8, 0.15, -)$$

ROC图的绘制方法：



在这里:我们为了能在解析公式(2.21)时复用此图所以没有写上具体地数值，转而用其数学符号代替。其中:

绿色线段： 表示在分类阈值变动的过程中只新增了真正例，

红色线段： 表示只新增了假正例，

蓝色线段： 表示既新增了真正例也新增了假正例。

根据AUC值的定义可知，此时的AUC值:

其实就是: **所有红色线段和蓝色线段与x轴围成的面积之和。**

观察上图可知，红色线段与x轴围成的图形恒为矩形，

蓝色线段与x轴围成的图形恒为梯形，

但是由于梯形面积公式既能算梯形面积，也能算矩形面积，

所以无论是红色线段还是蓝色线段，其与x轴围成的面积都能用梯形公式来计算，也即

$$\frac{1}{2} \cdot (x_{i+1} - x_i) \cdot (y_i + y_{i+1})$$

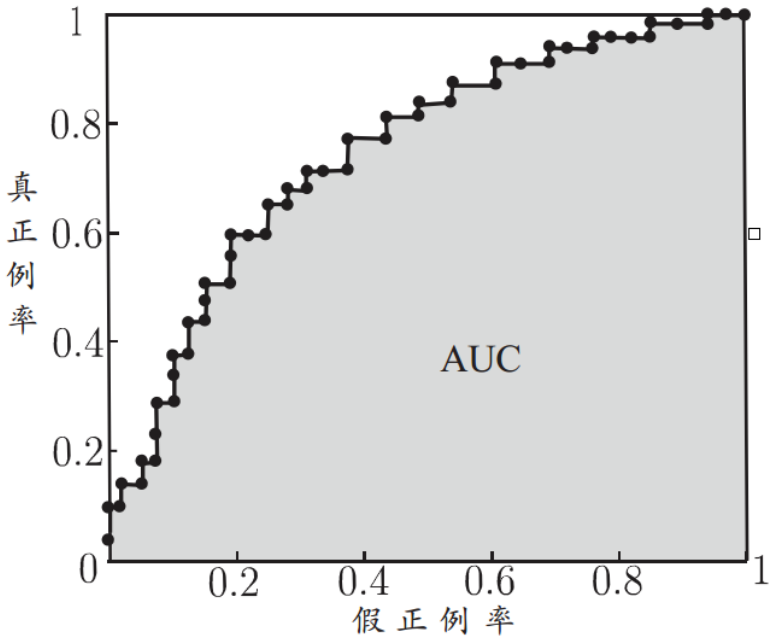
性能度量

若某个学习器的ROC曲线被另一个学习器的曲线“包住”，

则后者性能优于前者；

否则如果曲线交叉，可以根据ROC曲线下面积大小进行比较，

也即AUC值.



基于有限样例绘制的 ROC 曲线与 AUC

假设ROC曲线由 $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$

的点按序连接而形成 $(x_1 = 0, x_m = 1)$ ，

则：

AUC可估算为：

$$\mathrm{AUC} = \frac{1}{2} \sum_{i=1}^{m-1} (x_{i+1} - x_i) \cdot (y_i + y_{i+1})$$

AUC衡量了样本预测的排序质量。

代价敏感错误率

现实任务中不同类型的错误所造成的后果很可能不同，

为了权衡不同类型错误所造成的不同损失，

可为错误赋予**“非均等代价”**。

以二分类为例，可**根据领域知识设定“代价矩阵”，**

如下表所示，其中$cost_{ij}$ 表示将第i类样本预测为第j类样本的代价。

损失程度越大，$cost_{01}$与 $cost_{10}$值的差别越大。

表 2.2　二分类代价矩阵

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

代价敏感错误率

前面介绍的性能度量：它们大都**隐式地假设了均等代价**，

例如式(2.4)所定义的**错误率是：直接计算“错误次数”，并没有考虑不同错误会造成不同的后果.**

在非均等代价下，

我们希望不再是简单地最小化错误次数，而是**希望最小化“总体代价”(total cost).**

若将表2.2中的第0类作为正类、第1类作为反类,令***D+*与*D-***分别代表样例集**D**的正例子集和反例子集，则“代价敏感”(cost-sensitive)错误率为

在非均等代价下，不再最小化错误次数，**而是最小化“总体代价”**，则**“代价敏感”错误率**相应的为：

表 2.2　二分类代价矩阵

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

$$E(f;D;cost)=\frac{1}{m}\left(\sum_{\boldsymbol{x}_i\in D^+}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)\times cost_{01}+\sum_{\boldsymbol{x}_i\in D^-}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)\times cost_{10}\right).$$

性能度量

类似的，可给出:

基于分布定义的代价敏感错误率，以及其他一些性能度量如精度的代价敏感版本.

若令 $cost_{ij}$中的i、j取值不限于0、1，则可定义出多分类任务的代价敏感性能度量.

在非均等代价下，ROC曲线不能直接反映出学习器的期望总体代价，

而“代价曲线”可以。

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

$$\mathrm{TPR} = \frac{TP}{TP+FN},$$

$$\mathrm{FPR} = \frac{FP}{TN+FP}.$$

代价曲线的横轴是取值为[0, 1]的正例概率代价

$$P(+)cost = \frac{p \times cost_{01}}{p \times cost_{01} + (1-p) \times cost_{10}}$$

纵轴是取值为[0, 1]的归一化代价

$$cost_{norm} = \frac{\mathrm{FNR} \times p \times cost_{01} + \mathrm{FPR} \times (1-p) \times cost_{10}}{p \times cost_{01} + (1-p) \times cost_{10}}$$

其中 FPR 是式(2.19)定义的假正例率, $\mathrm{FNR} = 1 - \mathrm{TPR}$ 是假反例率.

性能度量

代价曲线图的绘制：ROC曲线上每个点对应了代价曲线上的一条线段，

设ROC曲线上点的坐标为(TPR, FPR)，则可相应计算出FNR,

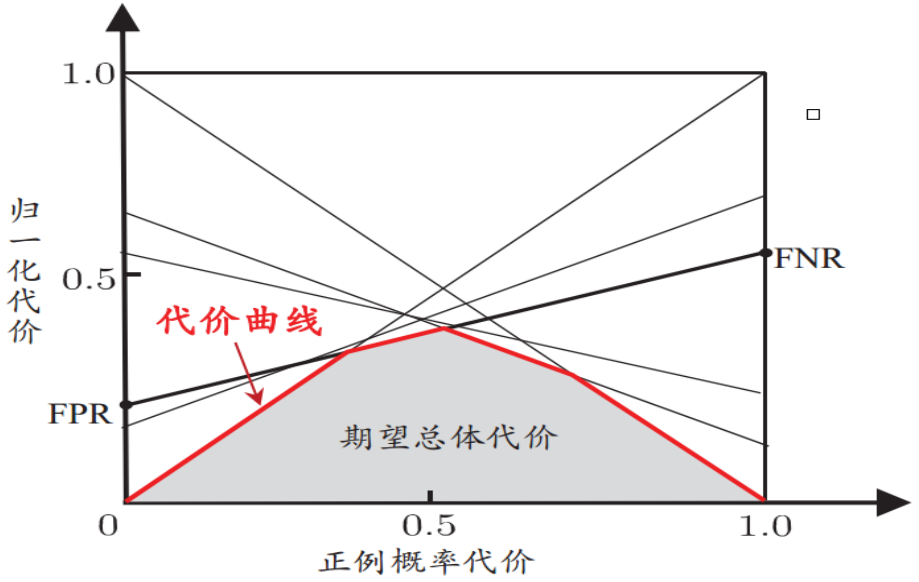
然后在代价平面上绘制一条从(0, FPR)到(1, FNR)的线段，

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曲线上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:所有条件下学习器的期望总体代价。



代价曲线与期望总体代价

目 录

比较检验

- 关于性能比较：
  - **测试性能并不等于泛化性能**
  - **测试性能随着测试集的变化而变化**
  - **很多机器学习算法本身有一定的随机性**

直接选取相应评估方法在相应度量下比大小的方法不可取！

假设检验为学习器性能比较提供了重要依据，基于其结果我们可以推断出：**若在测试集上观察到学习器A比B好，**

则A的泛化性能是否在统计意义上优于B，

以及这个结论的把握有多大。

二项检验

记泛化错误率为 ϵ，测试错误率为 $\hat{\epsilon}$，假定测试样本从样本总体分布中独立采样而来，我们可以使用“二项检验”对 $\epsilon \leq \epsilon_0$ 进行假设检验。

假设 $\epsilon \leq \epsilon_0$，若测试错误率小于

$$\bar{\epsilon} = \max \epsilon \quad \text{s.t.} \quad \sum_{i=\epsilon_0 \times m+1}^{m} \binom{m}{i} \epsilon^i (1-\epsilon)^{m-i} < \alpha$$

则在 α 的显著度下，假设不能被拒绝，也即能以 $1-\alpha$ 的置信度认为，模型的泛化错误率不大于 ϵ_0。

T检验

对应的，面对多次重复留出法或者交叉验证法进行多次训练/测试时可使用“t检验”。

假定得到了k个测试错误率，$\hat{\epsilon}_1, \hat{\epsilon}_2, \ldots, \hat{\epsilon}_k$，假设，$\epsilon = \epsilon_0$

对于显著度α，若$[t_{-\alpha/2}, t_{\alpha/2}]$位于临界范围$|\mu - \epsilon_0|$内，则假设不能被拒绝，即可认为**泛化错误率** $\epsilon = \epsilon_0$

其置信度为$1 - \alpha$.

交叉验证T检验

现实任务中，更多时候需要对**不同学习器的性能**进行比较

对两个学习器A和B，若k折交叉验证得到的测试错误率分别为 $\epsilon_1^A, \ldots, \epsilon_k^A$ 和 $\epsilon_1^B, \ldots, \epsilon_k^B$ ，可用**k折交叉验证“成对t检验”**进行比较检验。

若两个学习器的性能相同，

则他们使用相同的训练/测试集得到的测试错误率应相同，

即 .

$$\epsilon_i^A = \epsilon_i^B$$

交叉验证T检验

先对每对结果求差，$\Delta_i = \epsilon_i^A - \epsilon_i^B$，

若两个学习器性能相同，则差值应该为0，继而用 $\Delta_1, ..., \Delta_k$ 来对“学习器A与B性能相同”这个假设做t检验。

假设检验的前提是**测试错误率为泛化错误率的独立采样**，

然而由于样本有限，使用交叉验证导致训练集重叠，

测试错误率并不独立，从而　过高估计假设成立的概率，

为缓解这一问题，可采用“5*2交叉验证”法.

5*2交叉验证法

所谓5*2折交叉验证就是**做5次二折交叉验证**，每次二折交叉验证之前将数据打乱，使得5次交叉验证中的数据划分不重复。

为缓解测试数据错误率的非独立性，仅计算第1次2折交叉验证结果的**平均值** $\mu = 0.5(\Delta_1^1 + \Delta_1^2)$

和每次二折实验计算得到的**方差** $\sigma_i^2 = \left(\Delta_i^1 - \frac{\Delta_i^1+\Delta_i^2}{2}\right)^2 + \left(\Delta_i^2 - \frac{\Delta_i^1+\Delta_i^2}{2}\right)^2$

则变量

$$\tau_t = \frac{\mu}{\sqrt{0.2\sum_{i=1}^{5}\sigma_i^2}}$$

服从自由度为5的t分布。

McNemar检验

对于二分类问题，**留出法**不仅可以估计出学习器A和B的测试错误率，还能获得两学习器分类结果的差别，如下表所示

两学习器分类差别列联表

算法 B	算法 A	
	正确	错误
正确	e_{00}	e_{01}
错误	e_{10}	e_{11}

假设两学习器性能相同 $e_{01} = e_{10}$

则 $|e_{01} - e_{10}|$ 应服从正态分布，

且均值为1，方差为 $e_{01} + e_{10}$ ，

则

$$\tau_{\chi^2} = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$$

服从自由度为1的 χ^2 分布。

FRIEDMAN检验

交叉验证t检验和McNemar检验都是在一个数据集上比较两个算法的性能，可以用Friedman检验在一组数据集上对多个算法进行比较。

假定用 D_1, D_2, D_3, D_4 四个数据集对算法 A, B, C 进行比较。

先使用留出法或者交叉验证法得到每个算法在每个数据集上的测试结果，然后在每个数据集上根据性能好坏排序，并赋序值1, 2, …;若算法性能相同则平分序值, 继而得到每个算法的平均序值 r_i .

FRIEDMAN检验

得到表格如下所示，由平均序值进行Friedman检验来判断这些算法是否性能都相同。

算法比较序值表

数据集	算法 A	算法 B	算法 C
D_1	1	2	3
D_2	1	2.5	2.5
D_3	1	2	3
D_4	1	2	3
平均序值	1	2.125	2.875

则变量：

$$\tau_{\chi^2} = \frac{12N}{k(k+1)}\left(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4}\right)$$

服从自由度为k-1的 χ^2 分布

其中N，k表示数据集和算法数目

NEMENYI后续检验

若“所有算法的性能相同”这个假设被拒绝，说明算法的性能显著不同，此时可用Nemenyi后续检验进一步区分算法。

Nemenyi检验计算平均序值差别的临界阈值

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

如果两个算法的平均序值之差超出了临界阈值CD，则以相应的置信度拒绝“两个算法性能相同”这一假设。

FRIEDMAN检验图

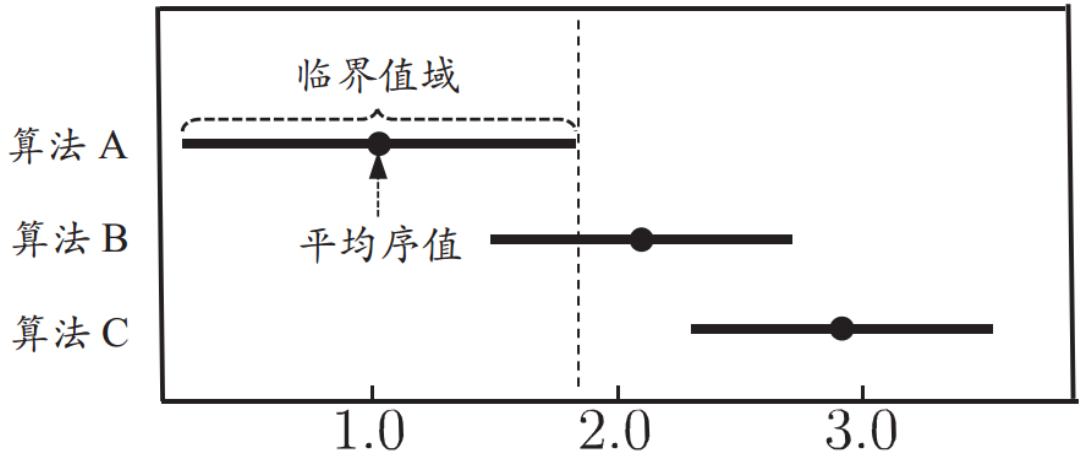
根据上例的序值结果可绘制如下Friedman检验图，

横轴为平均序值，每个算法圆点为其平均序值，

线段为临界阈值的大小。

若两个算法有交叠(A和B)，则说明没有显著差别;

否则有显著差别(A和C),算法A明显优于算法C.



目 录

偏差与方差

- **偏差**度量了学习算法期望预测与真实结果的偏离程度；

  即刻画了学习算法本身的拟合能力；

- **方差**度量了同样大小训练集的变动所导致的学习性能的变化；

  即刻画了数据扰动所造成的影响；

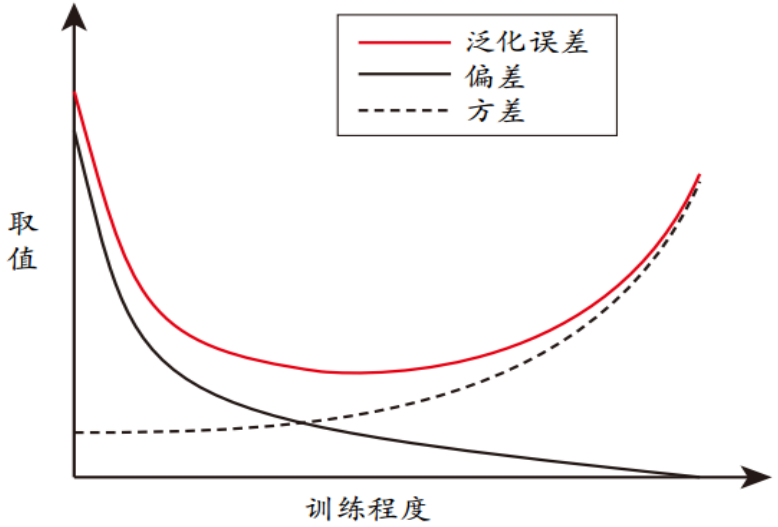
- **噪声**表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界；**即刻画了学习问题本身的难度。**

泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度所共同决定的。给定学习任务为了取得好的泛化性能，需要使**偏差小(充分拟合数据)而且方差较小(减少数据扰动产生的影响)**。

偏差与方差

一般来说，偏差与方差是有冲突的，称为**偏差-方差窘境**。

如右图所示，假如我们能控制算法的训练程度：



泛化误差与偏差、方差的关系示意图

- 在训练不足时，学习器拟合能力不强，训练数据的扰动不足以使学习器的拟合能力产生显著变化，此时**偏差主导泛化错误率**；
- 随着训练程度加深，学习器拟合能力逐渐增强，**方差逐渐主导泛化错误率**；
- 训练充足后，学习器的拟合能力非常强，训练数据的轻微扰动都会导致学习器的显著变化，若训练数据自身非全局特性被学到则会发生过拟合。

目 录

阅读材料

- 自助采样法在机器学习中有重要用途,[Efron and Tibshirani, 1993]对此有详细讨论。
- ROC曲线在二十世纪八十年代后期被引入机器学习[Spackman, 1989],AUC则是从九十年代中期起在机器学习领域广为使用[Bradley,1997].[Hand and Till,2001]将ROC曲线从二分类任务推广到多分类任务.[Fawcett,2006]综述了ROC曲线的用途.
- [Drummond and Holte,2006]发明了代价曲线.代价敏感学习[Elkan,2001;Zhou and Liu,2006]专门研究非均等代价下的学习。

阅读材料

- [Dietterich,1998]指出了常规k折交叉验证法存在的风险,并提出了5*2折交叉验证法.[Demsar, 2006]讨论了对多个算法进行比较检验的方法.

- [Geman et al.,1992]针对回归任务给出了偏差-方差-协方差分解，后来被简称为偏差-方差分解。但仅基于均方误差的回归任务中推导，对分类任务，由于0/1损失函数的跳变性,理论上推导出偏差-方差分解很困难。已有多种方法可通过试验队偏差和方差进行估计[Kong and Dietterich,1995;Kohavi and Wolpert, 1996; Breiman,1996;Friedman,1997;Domingos,2000].