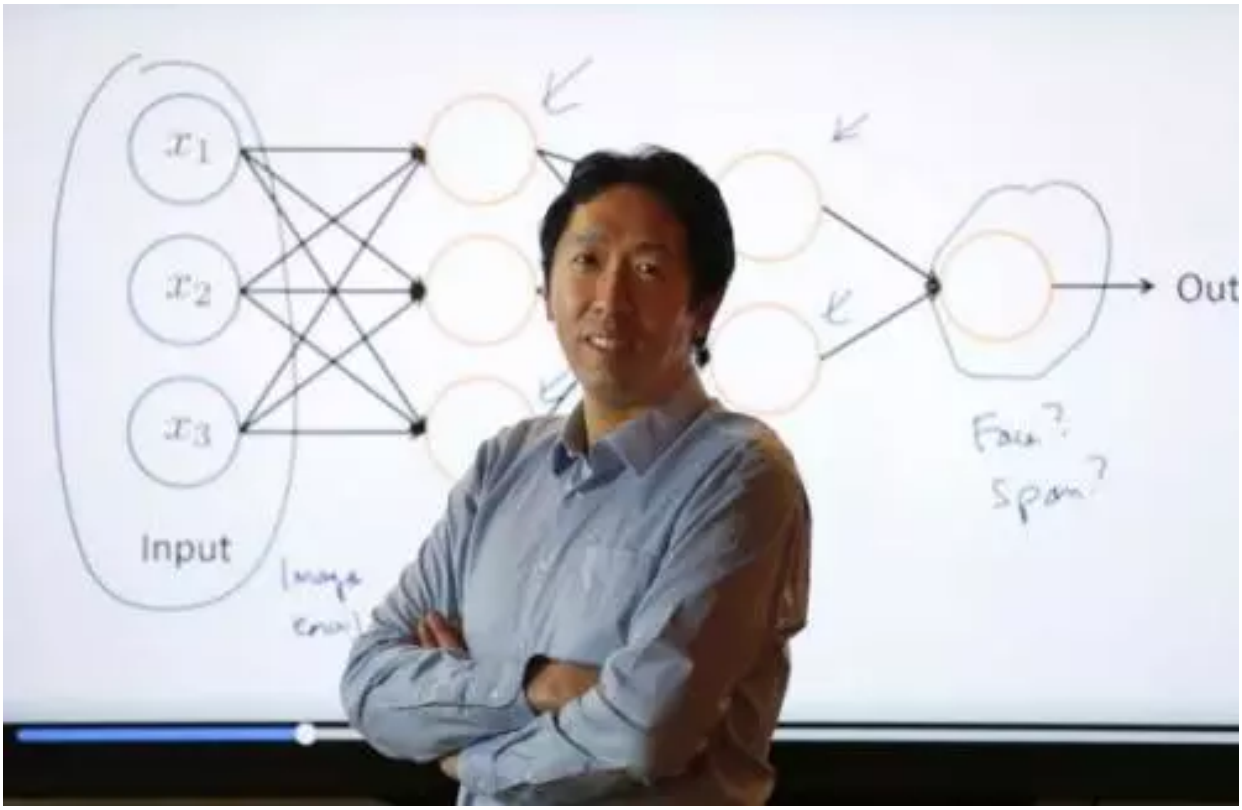
干货|全面理解模型性能评估方法

机器学习算法与自然语言处理 2019-03-22

以下文章来源于AI遇见机器学习 ，作者AI遇见机器学习

AI遇见机器学习

记录机器学习，深度学习，人工智能发展，自然语言处理，机器视觉等相关学习与前沿...



评估模型，不仅需要有效可行的实验估计方法，还需要有衡量模型泛化能力的评价标准，这便是性能度量（performance measure）。

性能度量反映任务需求，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，也即是说，模型的好坏其实也是相对的，什么样的模型是“合适”的，不仅和算法与数据有关，还和任务需求有关，而本章所述的性能度量，便是由任务需求出发，用于衡量模型的方法。

假设有数据集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$，其中$y_i$是示例$x_i$的真实标记（类别标签）。要评估学习器f的性能，我们就要将学习器预测结果与真实标记y进行比较。

一、均方误差

我们先介绍回归任务常用的性能度量方式，**均方误差**，想必它的表达式很多人已经比较熟悉了：

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

更一般的，对于数据分布D和概率密度函数p(·)，均方误差可描述为：

$$E(f;\mathcal{D})=\int_{\boldsymbol{x}\sim\mathcal{D}}\left(f\left(\boldsymbol{x}\right)-y\right)^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

二、错误率与精度

接下来我们主要介绍分类任务的性能度量。
分类任务中最常用的两种性能度量就是**错误率**和**精度**，既适用于二分类，也适用于多分类。
对样本集D，错误率表示分类错误的样本数占样本总数的比例，定义为：

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

（莫忘了 I() 是指示函数）

精度表示分类正确的样本占样本总数的比例，定义为：

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

而更一般的，对于数据分布D和概率密度函数p(·)，错误率与精度可分别描述为：

$$E(f;\mathcal{D}) = \int_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{I}(f(\boldsymbol{x}) \neq y)\, p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\ ,$$

$$\begin{aligned}\mathrm{acc}(f;\mathcal{D}) &= \int_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{I}(f(\boldsymbol{x}) = y)\, p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= 1 - E(f;\mathcal{D})\ .\end{aligned}$$

三、查准率、查全率

错误率和精度尽管常用，但并不代表它能满足所有的任务需求。假设瓜农拉来一车瓜（这个例子来自西瓜书，西瓜书的每个例子果然都离不开瓜），我们用某训练好的模型对这些西瓜进行判别，错误率便衡量了有多少比例的瓜被判别错误。但是，若我们关心的是“挑出的西瓜中好瓜的比例是多少”，或说“挑出的瓜占所有好瓜的比例”，这个时候我们就需要一些其它的性能度量来帮助我们解决这些问题了。

仅看上面这个例子似乎不太能理解为什么会有这样的问题，但事实上，类似的需求往往会在信息检索，web搜索等应用中经常出现：例如，在信息检索中，我们会关注“检索出的信息中用户感兴趣的信息占的比例是多少”，“有多少用户感兴趣的信息被检索出来了”……这个时候，**查准率**（precision）（亦称准确率）和**查全率**（recall）（亦称召回率）就是更为适用于此类需求的性能度量。

对于二分类问题，我们可以将样本根据其真实类别与学习器预测出来的类别的组合划分为**真实例**（true positive,TP），**假正例**（false positive,FP），**真反例**（true negative,TN），**假反例**（false negative,FN），若令TP,FP,TN,FN分别表示其对应的样本数，那么就有TP+FP+TN+FN=样本总数。分类结果的**混淆矩阵**（confusion matrix）如图：

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）

- TP：预测为正，实际为正的样本数量
- FN：预测为负，实际为正的样本数量
- FP：预测为正，实际为负的样本数量
- TN：预测为负，实际为负的样本数量

因此，很显然，预测正确的样本数量就是TP+TN，预测错的样本数量就是FN+FP，positive和negative表示其类别标签，true和false表示预测结果与真实情况是否相同。

由此，我们有查准率P的定义式：

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

查全率R的定义式：

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

查准率和查全率是一对矛盾的度量，换句话说，**在一般情况下，查准率越高查全率就会偏低；查全率越高，查准率就会越低。**

例如，我们若希望尽可能将好瓜都选出来，那么我们可以通过增加选中的瓜的数量来实现，若是将所有的瓜都选了，那必然也会选中所有的好瓜，但这样子查准率就会偏低；若希望选出的瓜中好瓜比例尽可能的高，那么就会只挑选有把握的瓜，但这样难免会漏掉不少好瓜，就使得查全率较低。一般只有在一些简单任务中，才能使查全率和查准率都高。

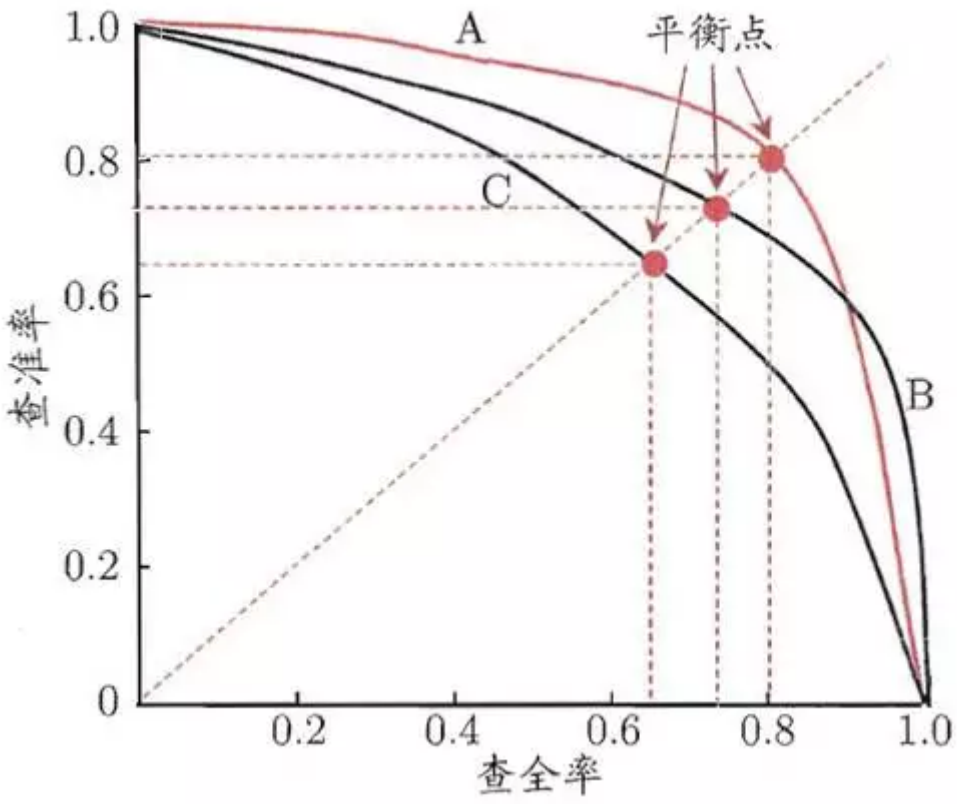
这个例子看起来有点奇怪，我们可以举一个分类垃圾邮件的例子，我们希望选中垃圾邮件准确率尽可能的高，但也绝不能把有用的邮件归为垃圾邮件，因为这样显然会惹恼用户，这个时候，就是“宁

放过，不杀错”，宁可查漏几封垃圾邮件，也不能不小心过滤掉了对于用户有用的邮件，也即是让查准率高，同时查全率也就相对变低。

在很多情况下，我们可以根据学习器的预测结果来对样本进行排序，学习器将人为“最可能”是正例的样本排在前面，“最不可能”是正例的样本排在后面，按照这个顺序我们逐个将样本作为正例进行预测，就可以在每一次计算出当前的查全率与查准率。（就是每添加一个样本计算一次查全率和查准率）

（以信息检索为例，我们逐条向用户反馈其可能感兴趣的信息，就可以计算出查全率，查准率）

我们以查准率为纵轴、查全率为横轴在二维平面上作图，得到查准率-查全率曲线，简称**P-R曲线**（PR曲线），显示该曲线的图称为**P-R图**（PR图），下面就是三个学习器的PR曲线组成的PR图：



PR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PR曲线被另一个学习器的PR曲线完全“包住”，我们就可以断言后者的性能优于前者，像上图中学习器A的性能就优于学习器C；若是两个学习器的PR曲线发生交叉，像A和B，就比较难断言孰优孰劣，只能是在具体的查准率或查全率条件进行比较。

但也有很多时候我们仍希望A和B分个高低，这个时候我们会选择比较PR曲线下面积的大小，这是一个比较合理的判据，它在一定程度上表征了学习器在查准率和查全率取得相对“双高”的比例。但

这个值又不是那么好算，因此，人们又设计了一些综合考虑查准率、查全率的性能度量：

四、平衡点（Break-Even Point，BEP）与F1

平衡点就是那么一个综合考虑查准率和查全率的性能度量，它是“查准率=查全率”时的取值，如上图中C的BEP就是0.64，而基于BEP进行比较，我们可以认为学习器A要比B好。

但BEP还是有些过于简单了，更常用的是**F1**度量：

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

其中F1是基于查准率和查全率的**调和平均**（harmonic mean）定义的：

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

F1度量的一般形式——F_β，能让我们表达出对查准率/查全率的不同偏好，它定义为：

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

实际上就是加权调和平均：

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

（与算数平均 (P+R)/2 和几何平均$\sqrt{(P*R)}^2$相比，调和平均更重视较小值，换句话说，就是受较小值的影响比受较大值的影响要大，举个例子，就好像是两个电阻并联的等效电阻的值受到电阻值较小的电阻的影响更大）

其中 β > 0 度量了查全率对查准率的**相对重要性**。β =1 时退化为标准的 F1 ； β > 1 时查全率具有更大影响； β < 1 时查准率有更大影响。

五、多个二分类混淆矩阵的综合考查

很多时候我们往往有多个二分类混淆矩阵，例如在进行多次训练/测试的时候，每次训练/测试都会得到一个混淆矩阵；或是在多个数据集上进行训练/测试；又或是执行多分类任务，类别的每一个两两组合都可以衍生出一个混淆矩阵……这个时候，我们就希望在 n 个二分类混淆矩阵上综合考察准确率和查全率。

一种方法是直接在各混淆矩阵上分别计算出查准率和查全率，记为$(P_1,R_1),\ldots,(P_n,R_n)$，在计算平均值，这样子得到的额就是**宏查准率**（macro-P）、**宏查全率**（macro-R），以及相应的**宏F1**（macro-F1）：

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

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

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

还有一种方法是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 $\bar{TP}, \bar{FP}, \bar{TN}, \bar{FN}$ 再基于这些平均值计算出微查准率（micro-P），微查全率（micro-R），微F1（micro-F1）：

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

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

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

六、ROC与AUC

一般来说，我们在解决分类问题的时候用的学习器都是为测试样本产生一个实值或者概率预测，然后将这个预测值与预设的一个**分类阈值**（threshold）进行比较，若大于阈值则认为样本为正，反之为负。这样的学习器我们见过很多，逻辑回归（阈值0.5），感知机（阈值0）等等，这个预测值结果的好坏往往直接决定学习器的泛化能力。

但我们并不是只有这一种方法，实际上，我们还可以根据这个预测值，对测试样本进行排序，“最可能”为正例的样本排在最前面，“最不可能”为正例的样本排在最后面。然后，我们规定分类过程就是在这个序列中以某个**截断点**（cut point）为基准将样本集划分为两部分，排在前面的那一部分当作正例，排在后面的那一部分当作反例。

这样子，我们就可以根据不同的任务需求来采用不同的截断点，假设我们更重视查准率，那就可以让截断点靠近序列的前面；若是重视查全率，则可以让截断点排在序列的靠后方。因此，排序本身的质量好坏，就体现了综合考虑学习器在不同任务下**期望泛化性能**的好坏，或者说是“一般情况下”泛化性能的好坏。

ROC曲线就是从这个角度出发来研究学习器泛化性能的工具。

ROC全称**受试者工作特征**（Receiver Operating Characteristic）曲线，它的绘制过程与PR曲线类似，我们首先根据学习器的预测结果对样本进行排序，然后从前往后逐个将样本作为正例进行预

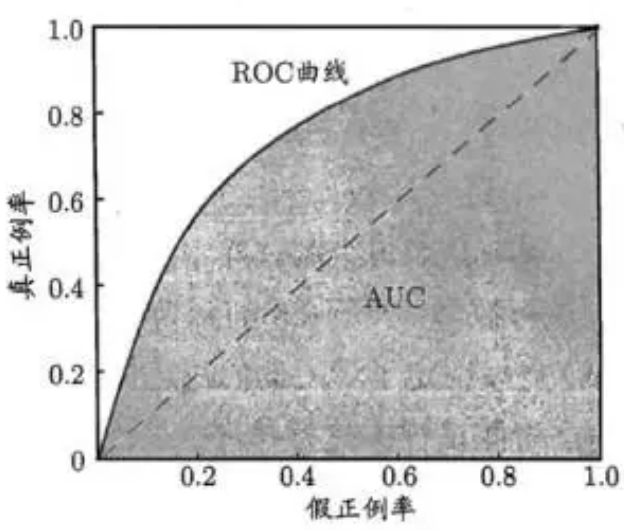
测，每多预测一个样本，就对已预测的所有样本计算两个重要的值。这两个值就是它与PR曲线的区别，这两个值分别是**真正例率**（True Positive Rate ,TPR），作为ROC曲线的纵轴；**假正例率**（False Positive Rate,FPR），作为横轴，两者的定义分别为：

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

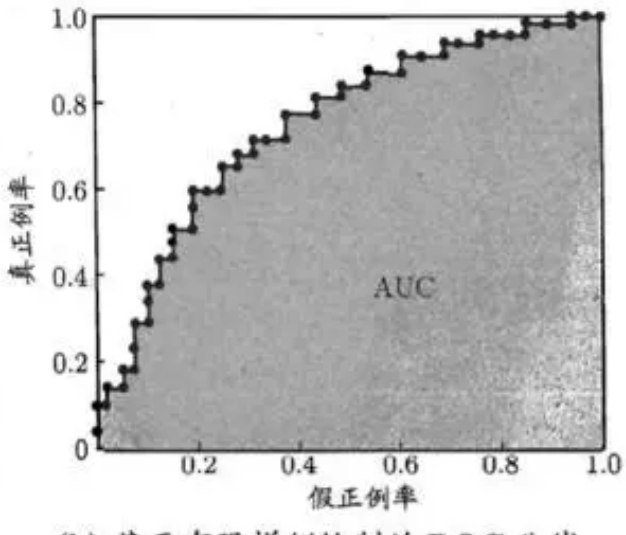
$$\mathrm{FPR} = \frac{FP}{TN + FP}.$$

TPR即预测正确的正例占预测正确的正例和预测错误的反例中比例，FPR即预测错误的正例占预测正确的反例和预测错误的正例的比例。

显示ROC曲线的图为ROC图，下面这幅图中的（a）图就是一个示例：



(a) ROC 曲线与 AUC



(b) 基于有限样例绘制的 ROC 曲线与 AUC

对角线表示的是“随机猜测”模型，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”。

现实中通常利用有限个测试样本来绘制ROC图，此时就仅能获得有限个（真正例率，假正例率）坐标对，结果就像（b）图一样，而无法得到（a）图中那样的光滑曲线。（PR图其实也有一样的问

题）

绘图过程如下：

假设我们有m^+个正例，m^-个反例：

1. 将分类阈值设为最大，即把所有样例均设为反例，此时真正例率和假正例率均为0，在坐标（0，0）处标记一个点
2. 将分类阈值一次设为每个样例的预测值，即截断点依次向后调，假设前一个标记点的坐标为（x,y），则后一个点若为真正例，坐标就是$(x, y+\frac{1}{m^+})$；若为假正例，则对应标记点坐标为$(x+\frac{1}{m^-}, y)$
3. 用线段连接相邻点即可

与PR图相同，若一个学习器的ROC曲线被另一个学习器包住，我们就断言后者性能优于前者；若两者发生交叉，则判断ROC曲线下的面积**AUC**（Area Under ROC Curve）

假设ROC曲线的坐标集合为$\{(x_1,y_1),\ldots,(x_m,y_m)\}$，其中$(x_1=0,x_m=1)$，这些点连成曲线，如图（b），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的计算公式之所以这么写，而不是直接计算对应矩形的面积之和，是因为有时候当多个样本的预测值相等时，我们调整阈值就会使得标记点相对上一个标记点斜向平移，这个时候用求矩形面积之和的公式就不方便计算了，参考AUC计算方法总结-CSDN ）

AUC考虑样本预测的排序质量，因此它与排序误差有紧密联系。给定m^+个正例和m^-个反例，令D^+和D^-分别表示正、反例集合，则排序**损失**（loss）定义为：

$$\ell_{rank}=\frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}\left(\mathbb{I}\left(f(x^+)<f(x^-)\right)+\frac{1}{2}\mathbb{I}\left(f(x^+)=f(x^-)\right)\right),$$

即考虑每一对正、反例，若正例的预测值小于反例，则记一个“罚分”，若相等，则记0.5个“罚分”。容易看出，l_{rank}对应的是ROC曲线上方的面积：若一个正例在ROC曲线上对应标记点的坐标为（x，y），则x恰是排序在这个样本之前的反例所占的比例，即假正例率。因此有：

$$\mathrm{AUC} = 1 - \ell_{rank} \ .$$

七、代价敏感错误率与代价曲线

在某些现实任务中，我们会发现，不同类型的错误造成的后果不同。例如在医疗诊断中，将一名癌症患者错误地诊断成健康人士，将一名健康人士错误地诊断为癌症患者，这两种情况要承担的后果是截然不同的：后者仅是增加了进一步检查的麻烦，而前者若丧失最佳治疗时机，则要付出生命的代价。

为此，我们为错误赋予**非均等代价**（unequal cost）来权衡不同类型错误所造成的不同损失。

以二分类任务为例，我们可以为任务设置一个**代价矩阵**（cost matrix），如下表所示：

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

其中$cost_{ij}$表示第i类样本错误预测为第j类的代价。一般情况下costii=0这是因为预测正确了，就没有代价了。我们规定，若将第0类判别为第一类的所造成的损失更大，那么就有$cost_{01} > cost_{10}$，损失程度越大则两者相差越大。但这个“相差”不是表示两者之间的差值，表示的是比值，举个例子$cost_{01}:cost_{10}=5:1$的效果和 $cost_{01}:cost_{10}=50:10$是相同的。

前面我们考虑的性能度量都是假设在**均等代价**的情况下，也即是说，以减小错误次数为主，但不考虑不同的错误类型造成的后果的严重程度。而在**非均等代价**下，我们希望最小化**总体代价**（total

cost）。若设上表中第0类为正类，第1类为反类，令D^+和D^-分别代表样本集中的正类样本子集和反类样本子集，则**代价敏感**（cost-sensitive）错误率为：

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

由此我们可以推出其它性能度量的代价敏感版本，或者是基于分布定义的代价敏感错误率。若$cost_{ij}$中的值不局限于0，1，还可以定义出多分类任务的代价敏感性能度量。

在非均等代价下，我们用**代价曲线**（cost curve）代替ROC曲线表现学习器的期望总体代价。代价曲线图横轴是取值为[0,1]的**正例概率代价**：

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

其中p表示样例为正例的概率。

纵轴是取值为[0,1]的**归一化代价**：

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

归一化是**规范化**（normalization）的特例，规范化表示将不同变化范围的值映射到某相同、固定的范围当中，常见的固定范围是[0,1]，这个时候就是“归一化”。

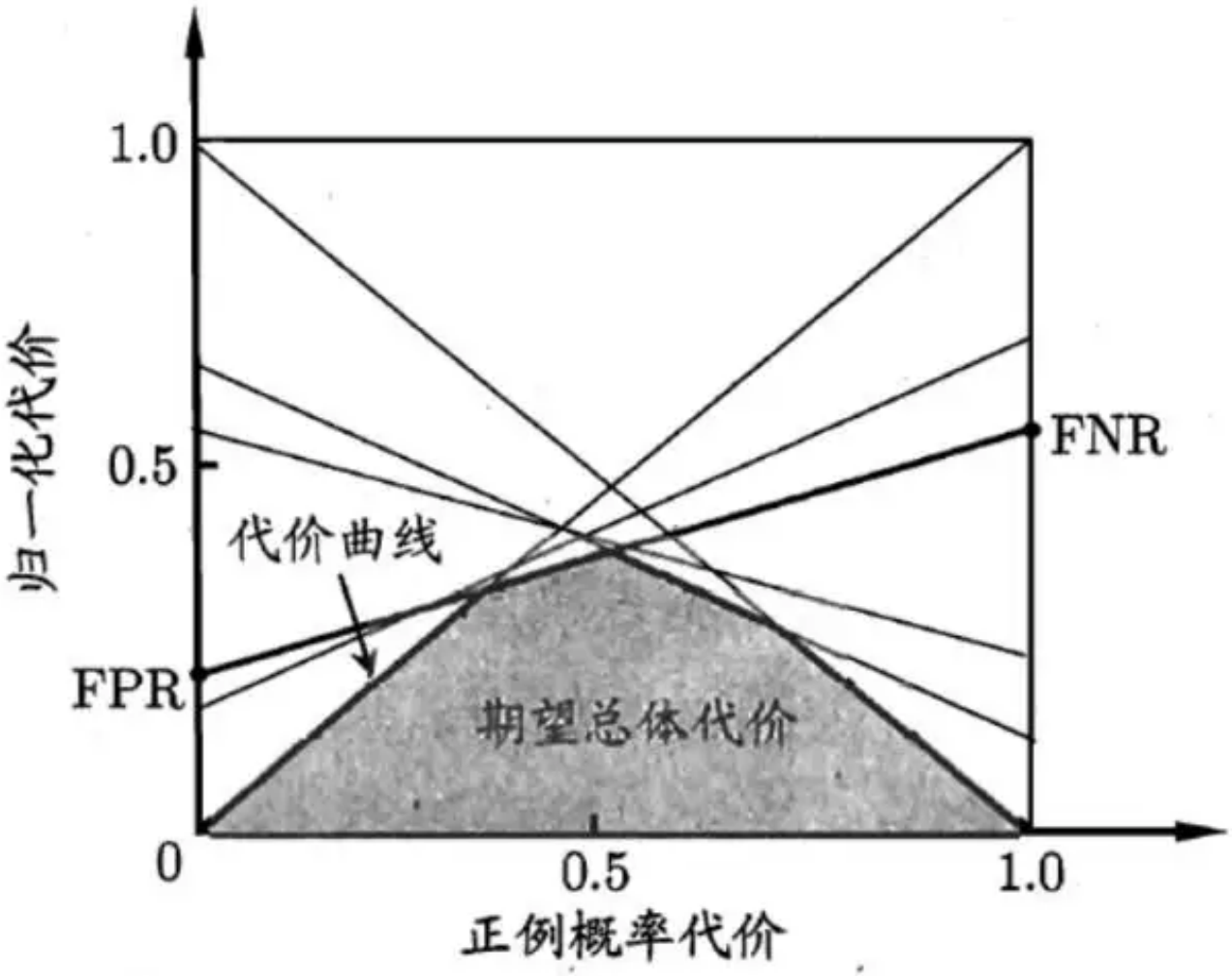
FPR即之前定义的假正例率，FNR=1-TPR 是假反例率。

代价曲线的绘制方式：

1. ROC曲线上每一点对应代价平面上的一条线段，设ROC曲线上的坐标为（FPR，TPR），计算出相应的FNR

2. 在代价平面上绘制一条从（0，FPR）到（1，FNR）的线段，线段下的买诺记就表示了该条件下的期望总体代价
3. 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届，围成的面积即为在所有条件下学习器的期望总体代价

如下面的示例图：



推荐阅读

一起走进自然语言处理的世界

干货|全面理解N-Gram语言模型

你还记得吗？那些最基础的机器学习知识。

干货|学术论文怎么写

欢迎关注我们，看通俗干货！