机器学习中常用评估指标汇总

原创 Alice 机器学习X计划 2017-07-03

评估指标 Evaluation metrics 可以说明模型的性能，辨别模型的结果。

我们建立一个模型后，计算指标，从指标获取反馈，再继续改进模型，直到达到理想的准确度。在预测之前检查模型的准确度至关重要，而不应该建立一个模型后，就直接将模型应用到看不见的数据上。

今天先来简单介绍几种回归和分类常用的评估方法。

回归：

均方误差：

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

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

其中 D 为数据分布，p 为概率密度函数。

```
from sklearn.metrics import mean_squared_error
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
mean_squared_error(y_true, y_pred)

0.375
```

分类：

二分类 and 多分类：

错误率

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(x_i)\neq y_i) \ .$$

精度

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

二分类

混淆矩阵：

表 2.1 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)


```
from sklearn.metrics import confusion_matrix
pipe_svc.fit(X_train, y_train)
y_pred = pipe_svc.predict(X_test)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
print(confmat)

[[71  1]
[ 2 40]]
```

单纯用 错误率，精度 是无法知道下面的问题时：

查准率：

应用场景 - 当你想知道“挑出的西瓜中有多少比例是好瓜”

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


```
from sklearn.metrics import precision_score
from sklearn.metrics  import recall_score, f1_score
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))

Precision: 0.976
```

查全率：

应用场景 - 当你想知道“所有好瓜虫有多少比例被挑出来了”

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

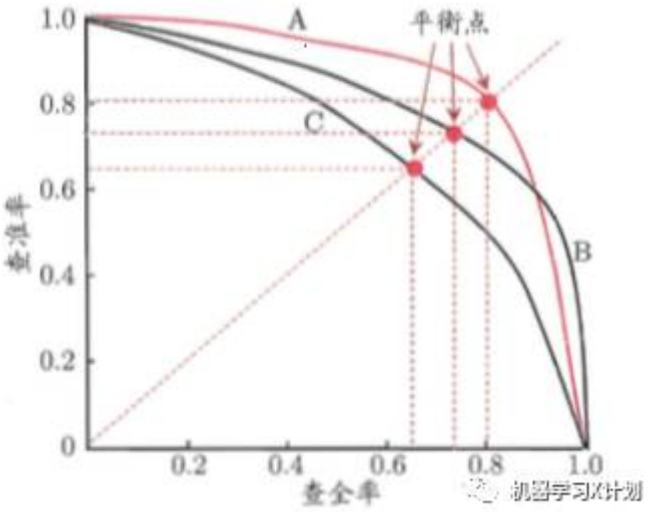

```
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))

Recall: 0.952
```

P - R 图：

当一个学习器的 P - R 曲线被另一个学习器的包住，那么后者性能优于前者。

有交叉时，需要在具体的查准率或者查全率下进行比较。



平衡点 (Break Event Point BEP)：

即上图中三个红点。

综合考虑查准率，查全率的度量

当 查准率 = 查全率 时的点，谁大谁比较优。

F1 度量：

也是综合考虑查准率，查全率的度量，比 BEP 更常用：

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


```
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))

F1: 0.964
```

Fβ：

可以表达对查准率，查全率的不同重视度，

β > 1 则查全率有更大影响，β < 1 则查准率有更大影响，β = 1 则为 F1。

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

One vs. All (OvA) 分类问题

这时会在 n 个二分类问题上综合考虑查准率，查全率。

宏 ~ ：先在每个混淆矩阵上计算率，再求平均

宏查准率

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

宏查全率

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

宏 F1

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

微 ~ ：先将各个混淆矩阵上对应元素求平均，再计算率

微查准率

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

微查全率

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

微 F1

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

ROC：

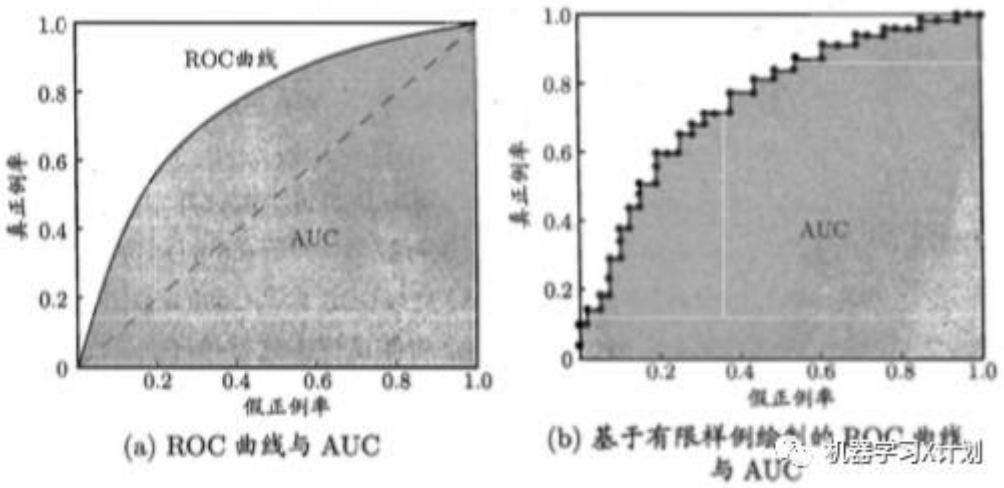
反映敏感性和特异性连续变量的综合指标，roc曲线上每个点反映着对同一信号刺激的感受性。

纵轴为 TPR 真正例率，预测为正且实际为正的样本占所有正例样本的比例
横轴为 FPR 假正例率。预测为正但实际为负的样本占所有负例样本的比例

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

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

对角线对应的是“随机猜想”



(a) ROC 曲线与 AUC　　(b) 基于有限样例绘制的 ROC 曲线与 AUC

当一个学习器的 ROC 曲线被另一个学习器的包住，那么后者性能优于前者。
有交叉时，需要用 AUC 进行比较。

AUC:

ROC 曲线下的面积

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


```
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])
roc_auc_score(y_true, y_scores)

0.75
```

代价敏感

现实任务中，当不同类型的错误具有不同的影响后果时，它们的代价也是不一样的。

此时，可以设定

代价矩阵 cost matrix：

如果将第 0 类预测为 第 1 类造成的损失更大，则 cost01 > cost10，相反将第 1 类预测为 第 0 类造成的损失更大，则 cost01 < cost10 :

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

则带有“代价敏感”的错误率为：

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

其中 0 为正类，1 为反类，D + 为正例子集合，D - 为反例子集合。

代价曲线 cost curve：

非均等代价下，反应学习器的期望总体代价。

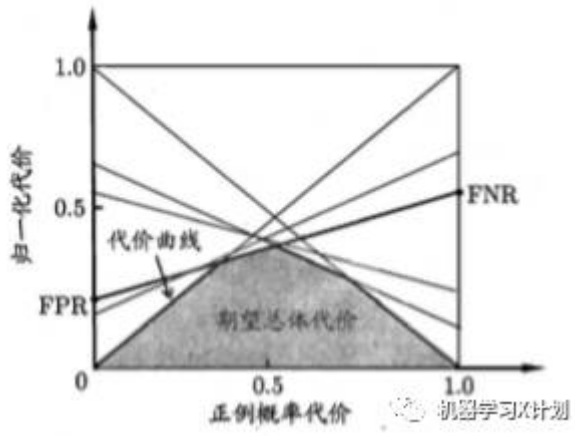
横轴为取值为［0，1］的正例概率代价：

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

纵轴为取值为［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 p 为正例的概率，FPR = 1 - TPR。



资料：

机器学习

Python Machine Learning

推荐 阅读原文

也许可以找到你想要的：

[入门问题][TensorFlow][深度学习][强化学习][神经网络][机器学习][自然语言处理][聊天机器人]