CSDN
不 止 于 代 码

2.5 分类模型的评价

CSDN学院

2017年11月

大纲

- Logistic回归基本原理
- 多类Logistic回归
- Scikit learn 中的Logistic回归实现
- 不均衡样本学习
- 分类模型的评价
- 模型选择与参数调优
- 案例分析

分类模型性能评价

- 损失函数可以作为评价指标(log_loss、 zero_one_loss、 hinge_loss)
- logistic / 负log似然损失（log_loss）：
  - $\text{logloss} = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M} y_{ij} log p_{ij}$,
    - M为类别数，y_{ij}为二值，当第i个样本为第j类时$y_{ij} = 1$，否则取0；p_{ij}为模型预测的第i个样本为第j类的概率
    - 当M=2时，$\text{logloss} = -\frac{1}{N}\sum_{i=0}^{N}\left(y_i log p_i + (1-y_i) log(1-p_i)\right)$
      - y_i为第i个样本类别，p_i为模型预测的第i个样本为第1类的概率

- 0-1损失 (zero_one_loss) （错误率、正确率评价指标均与此有关）
  - $\text{MCE} = -\frac{1}{N}\sum_{\hat{y}_i \neq y_i} 1$

http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

► 两类分类任务中更多评价指标

- ROC / AUC
- PR曲线
- MAP@n

False Positive & False Negative

- 0-1损失：假设两种错误的代价相等
  - False Positive （ FP ） & False Negative (FN)
- 有些任务中可能某一类错误的代价更大
  - 如医疗诊断中将病例误分为正常，错过诊疗时机
  - 因此单独列出每种错误的比例：混淆矩阵
- 混淆矩阵（ confusion matrix ）
  - 真正的正值（ true positives ）
  - 假的正值（ false positives ）
  - 真正的负值（ true negatives ）
  - 假的负值（ false negatives ）

观测值 / 真值 \ 预测值	$\hat{y}=1$	$\hat{y}=0$	Σ
$y=1$	#TP	#FN	N_+
$y=0$	#FP	#TN	N_-
Σ	$\hat{N}_+$	$\hat{N}_-$	

confusion matrix

- Scikit-learn实现了多类分类任务的混淆矩阵
- sklearn.metrics.confusion_matrix(*y_true*, *y_pred*, *labels=None*, *sample_weight=None*)
  - *y_true*：*N*个样本的标签观测值 / 真值
  - *y_pred*：*N*个样本的预测标签值
  - *labels*：*C*个类别在矩阵的索引顺序
    - 缺省为*y_true*或*y_pred*类别出现的顺序
  - *sample_weight*：*N*个样本的权重

手写数字识别的混淆矩阵

```
[[87  0  0  0  1  0  0  0  0  0]
 [ 0 88  1  0  0  0  0  0  1  1]
 [ 0  0 85  1  0  0  0  0  0  0]
 [ 0  0  0 79  0  3  0  4  5  0]
 [ 0  0  0  0 88  0  0  0  0  4]
 [ 0  0  0  0  0 88  1  0  0  2]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  0  0  0  0  1  0 88  0  0]
 [ 0  0  0  0  0  0  0  0 88  0]
 [ 0  0  0  1  0  1  0  0  0 90]]
```

真值

预测值

Receiver Operating Characteristic (ROC)

	$\hat{y}=1$	$\hat{y}=0$	Σ
$y=1$	#TP	#FN	N_+
$y=0$	#FP	#TN	N_-
Σ	$\hat{N}_+$	$\hat{N}_-$	

$$\text{accuracy} = \frac{TP+TN}{N}$$

$$\text{error rate} = \frac{FP+FN}{N}$$

	$y=1$	$y=0$	$y=1$	$y=0$
$\hat{y}=1$	$TP/\hat{N}_+$=precision	$FP/\hat{N}_+$=FDP	TP/N_+=TPR	FP/N_-=FPR
$\hat{y}=0$	$FN/\hat{N}_-$	$TN/\hat{N}_-$=NPV	FN/N_+=FNR	TN/N_-=TNR

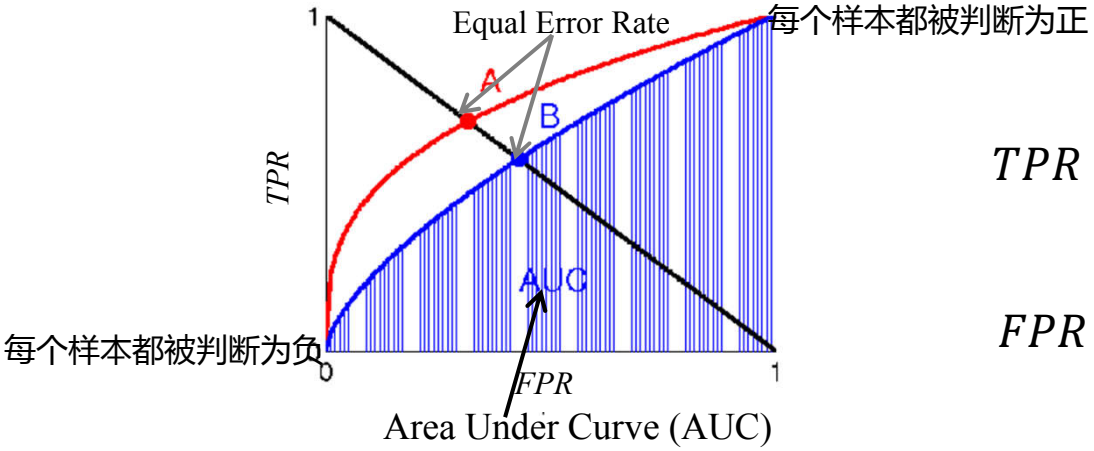
PPV - positive predictive value, **precision** 预测结果为真的样本中真正为真的比例

TPR - true positive rate, **sensitivity, recall**, hit rate 预测结果召回了多少真正的真样本

FPR – False positive rate, **false alarm**, fallout 预测结果将多少假的样本预测预测成了真

Receiver Operating Characteristic (ROC)

- 上面我们讨论给定阈值τ的TPR和FPR
- 如果不是只考虑一个阈值，而是在一些列阈值上运行检测器，并画出TPR和FPR为阈值τ的隐式函数，得到ROC曲线。



$$TPR = \frac{TP}{N_+}$$

$$FPR = \frac{FP}{N_-}$$

PR曲线

- Precision and Recall (PR曲线)：用于稀有事件检测，如目标检测、信息检索
  - 负样本非常多，因此$FPR = FP/N_$很小，比较TPR和FPR不是很有信息（ROC曲线中只有左边很小一部分有意义）→ 只讨论正样本
  - Precision（精度，查准率）：以信息检索为例，对于一个查询，返回了一系列的文档，正确率指的是返回结果中相关文档占的比例
    - Precision=返回结果中相关文档的数目/返回结果的数目
  - Recall（召回率，查全率）：返回结果中相关文档占所有相关文档的比例
    - Recall=返回结果中相关文档的数目/所有相关文档的数目

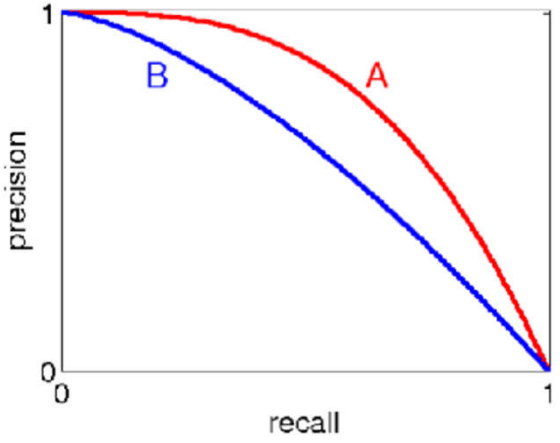
PR曲线(cont.)

- Precision and Recall (PR曲线)
  - 阈值变化时的P和R

Precision: the fraction of our detection are actually positive
Recall: the fraction of the positives we actually detected

$\text{Precsion} = TP/\hat{N}_+$:检测结果真正为正的比例

$\text{Recall} = TP/N_+$:被正确检测到的正样本的比例



AP

- Precision只考虑了返回结果中相关文档的个数，没有考虑文档之间的序。
- 对一个搜索引擎或推荐系统而言，返回的结果是有序的，且越相关的文档越靠前越好，于是有了AP的概念。
- AP: Average Precision，对不同召回率点上的精度进行平均
  - $AP = \int_0^1 p(k)dr = \sum_{k=0}^{n} p(k)\Delta r(k)$
  - 即PR曲线下的面积（Recall： AUC为ROC下的面积）
  - 其中k为返回文档中的序位，n为返回文档的数目，$p(k)$为列表中k截止点的precision， $\Delta r(k)$表示从k-1到k Recall的变化。
- 上述离散求和表示等价于：$AP = \sum_{k=0}^{n} p(k)rel(k)/$相关文档数目，其中$rel(k)$为示性函数，即第$k$个位置为相关文档取1，否则取0.
  - 计算每个位置上的precision，如果该位置的文档是不相关的则该位置 precision=0
  - 然后对所有的位置的precision再求平均

Mean Average Precision

- 多个查询的AP平均：
  - $MAP = \left(\sum_{q=0}^{Q} AP(q)\right)/(Q)$，
  - 其中Q为查询的数目，n为文档数目

MAP@K（MAP_K）

- 在现代web信息检索中，recall其实已经没有意义，因为相关文档有成千上万个，很少有人会关心所有文档。
- Precision@K：在第K个位置上的Precision，
  - 对于搜索引擎，考虑到大部分作者只关注前一、两页的结果，所以Precision @10， Precision @20对大规模搜索引擎非常有效
- MAP@K：多个查询Precision@K的平均

F1 分数

- 亦被称为F1 score, balanced F-score or F-measure
- Precision 和 Recall 调和平均：$F1 = \frac{2*(\text{Precision}*\text{Recall})}{(\text{Precision}+\text{Recall})}$
  - 最好为1，最差为0
  - 多类：每类的F1平均值

模型性能评价

- Scikit learn 提供3 不同的 API，用于评估模型预测的性能：
  - **Estimator score method**: 模型自带的分数方法（score函数）提供一个缺省的评估准则。
  - **Scoring parameter**: 采用交叉验证的模型评估工具（ model_selection.cross_val_score and model_selection.GridSearchCV、以及一些xxxCV类 ）有 *scoring* 参数（最佳参数为最大*scoring*模型对应的参数）
  - **Metric functions**: metrics模块提供评价预测性能的功能
    - Classification metrics,
    - Multilabel ranking metrics
    - Regression metrics
    - Clustering metrics

分类模型性能评价

- 对分类模型，缺省的score函数返回的是正确率（Mean accuracy）

scoring参数

- 交叉验证中可设置scoring参数，规定模型性能的评价指标
- 注意：scoring越大的模型性能越好，所以如果采用损失/误差，需要加neg，如‘neg_log_loss’

Scoring	Function	Comment
Classification		
‘accuracy’	metrics.accuracy_score	
‘average_precision’	metrics.average_precision_score	
‘f1’	metrics.f1_score	for binary targets
‘f1_micro’	metrics.f1_score	micro-averaged
‘f1_macro’	metrics.f1_score	macro-averaged
‘f1_weighted’	metrics.f1_score	weighted average
‘f1_samples’	metrics.f1_score	by multilabel sample
‘neg_log_loss’	metrics.log_loss	requires predict_proba support
‘precision’ etc.	metrics.precision_score	suffixes apply as with ‘f1’
‘recall’ etc.	metrics.recall_score	suffixes apply as with ‘f1’
‘roc_auc’	metrics.roc_auc_score	

▶可以自定义评价函数

- 有些指标还需要额外的参数，而没有在scoring出现，或者某个任务需要特殊的指标，scikit learn支持自定义scoring函数

```
from sklearn.metrics import fbeta_score, make_scorer
ftwo_scorer = make_scorer(fbeta_score, beta=2)

from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC

grid = GridSearchCV(LinearSVC(), param_grid={'C': [1, 10]},
scoring=ftwo_scorer)
```

需要beta参数

可以自定义评价函数

```
# 定义特异性（specificity）计算函数
def specificity(y, y_hat):
#confusion matrix → a numpy.ndarray object
mc = metrics.confusion_matrix(y, y_hat)

#"negative" is the first row (index 0) of the matrix
import numpy
res = mc[0,0]/numpy.sum(mc[0,:])

return res
```

$$\text{Confusion matrix} = \begin{bmatrix} 184 & 17 \\ 45 & 54 \end{bmatrix}$$

```
# 将上述函数作为评价函数
specificite = metrics.make_scorer(specificity, greater_is_better = True)

 # 使用自定义的评价函数
#modele is the classifier fitted on the training set
sp = specificite(modele, X_test, y_test)
print (sp)  # 0.915 = 184 / (184 + 17)
```

分类模型性能评价

- Scikit learn中Classification metrics 模块针对两类分类问题提供的性能评价指标有

precision_recall_curve(y_true, probas_pred)	Compute precision-recall pairs for different probability thresholds
roc_curve(y_true, y_score[, pos_label, …])	Compute Receiver operating characteristic (ROC)

亦可用于multilabel场合

average_precision_score(y_true, y_score[, …])	Compute average precision (AP) from prediction scores
roc_auc_score(y_true, y_score[, average, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

可以适用于Multiclass任务：

cohen_kappa_score(y1, y2[, labels, weights, …])	Cohen’s kappa: a statistic that measures inter-annotator agreement.
confusion_matrix(y_true, y_pred[, labels, …])	Compute confusion matrix to evaluate the accuracy of a classification
hinge_loss(y_true, pred_decision[, labels, …])	Average hinge loss (non-regularized)
matthews_corrcoef(y_true, y_pred[, …])	Compute the Matthews correlation coefficient (MCC)

亦可用于multilabel场合

accuracy_score(y_true, y_pred[, normalize, …])	Accuracy classification score.
classification_report(y_true, y_pred[, …])	Build a text report showing the main classification metrics
f1_score(y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
fbeta_score(y_true, y_pred, beta[, labels, …])	Compute the F-beta score
hamming_loss(y_true, y_pred[, labels, …])	Compute the average Hamming loss.
jaccard_similarity_score(y_true, y_pred[, …])	Jaccard similarity coefficient score
log_loss(y_true, y_pred[, eps, normalize, …])	Log loss, aka logistic loss or cross-entropy loss.
precision_recall_fscore_support(y_true, y_pred)	Compute precision, recall, F-measure and support for each class
precision_score(y_true, y_pred[, labels, …])	Compute the precision
recall_score(y_true, y_pred[, labels, …])	Compute the recall
zero_one_loss(y_true, y_pred[, normalize, …])	Zero-one classification loss.

CSDN
不 止 于 代 码

THANK YOU

AI 100