R 机器学习

第 07 讲　Logistic 回归

张敬信

2022 年 10 月 27 日

哈尔滨商业大学

一. 广义线性模型

- 线性回归，要求因变量是服从正态分布的连续型数据。但实际中，因变量数据可能会是类别型、计数型等。
- 要让线性回归也适用于因变量非正态连续情形，就需要推广到广义线性模型。
- Logistic 回归、softmax 回归、泊松回归、Probit 回归、二项回归、负二项回归、最大熵模型等都是广义线性模型的特例。

- 广义线性模型，相当于是复合函数。先做线性回归，再接一个变换：

$$\mathbf{w}^T X + \mathbf{b} = u \sim \text{正态分布}$$

$$\downarrow$$

$$g(u) = y$$

- 经过变换后到达非正态分布的因变量数据。

- 一般更习惯反过来写：即对因变量 y 做一个变换，就是正态分布，从而就可以做线性回归：

$$\sigma(y) = \mathbf{w}^T X + \mathbf{b}$$

- $\sigma(\cdot)$ 称为连接函数。

常见的连接函数和误差函数

回归模型	变换	连接函数	逆连接函数	误差
线性回归	恒等	$\mu_Y = x^T\beta$	$\mu_Y = x^T\beta$	正态分布
Logistic 回归	Logit	$\text{Logit}\ \mu_Y = x^T\beta$	$\mu_Y = \frac{\exp(x^T\beta)}{1+\exp(x^T\beta)}$	二项分布
泊松回归	对数	$\ln \mu_Y = x^T\beta$	$\mu_Y = \exp\left(x^T\beta\right)$	泊松分布
负二项回归	对数	$\ln \mu_Y = x^T\beta$	$\mu_Y = \exp\left(x^T\beta\right)$	负二项分布
Gamma 回归	逆	$\frac{1}{\mu_Y} = x^T\beta$	$\mu_Y = \frac{1}{x^T\beta}$	Gamma 分布

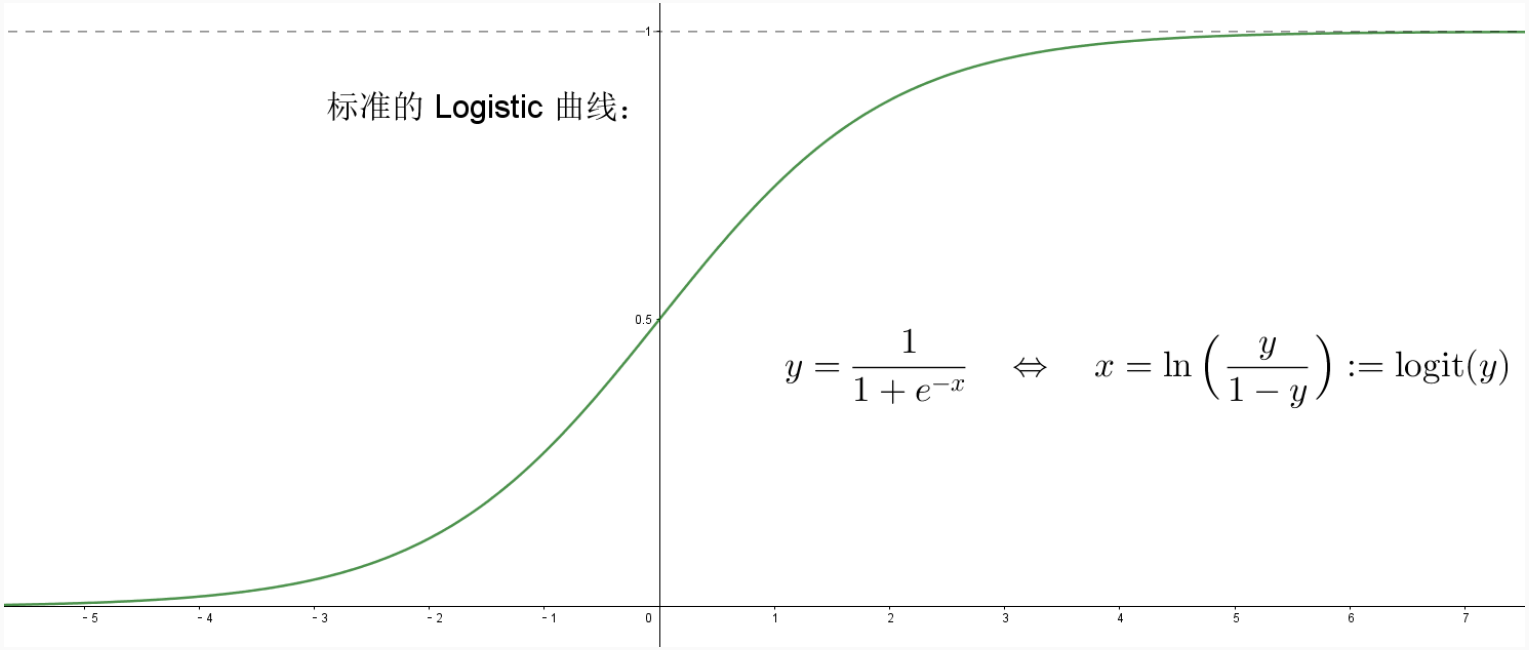
- 注：因变量数据只要服从指数族分布：正态分布、伯努利分布、泊松分布、指数分布、Gamma 分布、卡方分布、Beta 分布、狄里克雷分布、Categorical 分布、Wishart 分布、逆 Wishart 分布等，就可以使用对应的广义线性模型。

二. Logistic 回归

- Logistic 回归是分类模型，适合因变量是分类数据（例如：患病与不患病；违约与不违约）。
- 对于二分类因变量，$y=1$ 表示事件发生；$y=0$ 表示事件不发生。事件发生的条件概率 $\Pr\{y=1|X\}$ 与 x_i 之间是非线性关系，通常是单调的，即随着 X 的增加/减少，$\Pr\{y=1|X\}$ 也增加/减少。

1. Logistic 回归原理

- Logistic 回归可看作是先做线性回归，再接一个逆连接函数：sigmoid 函数



- sigmoid 函数值域在 $(0,1)$ 之间，而且是一个渐变过程，正好适合描述概率 $\Pr\{y=1|X\}$.
- 概率值 $\Pr\{y=1|X\}$ 有了，再根据阈值 0.5 做判断：大于 0.5, 则预测 $\hat{y}=1$; 小于 0.5, 则预测 $\hat{y}=0$.

- 于是，二项 Logistic 回归模型可表示为

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \ln\left(\frac{\Pr\{y=1|X\}}{\Pr\{y=0|X\}}\right) = \mathbf{w}^T X + \mathbf{b}$$

2. Logistic 回归系数的解释

- 例如，影响是否患病的因素有性别和肿瘤体积，通过 Logistic 回归建模，得到

$$\text{Odds} = \frac{p}{1-p} = e^{w_0 + w_1 \text{Gender} + w_2 \text{Volume}} = e^{w_0} \cdot e^{w_1 \text{Gender}} \cdot e^{w_2 \text{Volume}}$$

- 对于离散型变量 Gender，男用 1 表示，女用 0 表示，代入可得性别变量的发生比率为：

$$\frac{\text{Odds}_1}{\text{Odds}_2} = e^{w_1}$$

这表示男性患病的发生比约为女性患病发生比的 e^{w_1} 倍。

- 对于连续型变量 Volume，若肿瘤体积从 v_0 增加 1 个单位，则

$$\frac{\text{Odds}_{v_0+1}}{\text{Odds}_{v0}} = e^{w_2}$$

这表示在其它变量不变的情况下，肿瘤体积每增加 1 个单位，将会使患病发生比变化 e^{w_2} 倍，注意该倍数是相对于原来 v_0 而言的。

3. Logisic 回归的损失函数

- 二分类 Logistic 回归用的是交叉熵损失：

$$J(\mathbf{w}, \mathbf{b}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \ln \hat{p}_i + (1 - y^{(i)}) \ln(1 - \hat{p}_i)\right]$$

其中，y_i 为样本真实类别，$\hat{p}_i = \Pr\{y = 1 | X\}$ 为预测为正例的概率。

```
y_true = [  0,        0,        1,        1]
y_pred = [[.9, .1], [.8, .2], [.3, .7], [.01, .99]]
```

```r
y = c(0, 0, 1, 1)                    # 真实类别
p = c(0.1, 0.2, 0.7, 0.99)           # 预测为正例的概率

- mean(y * log(p) + (1-y) * log(1-p))   # 交叉熵损失
#> [1] 0.174
```

4. 阈值调参

- 得到预测概率后，通常是以 0.5 为阈值，将 y 预测为正类或负类，这只适合于均衡分类与代价不敏感的问题。
- 对于不均衡分类或代价敏感的问题，以 0.5 为阈值往往不是最优选择，需要根据具体需要，选择最优的阈值。

- 对于 y 是多分类情形，是 K 类可能的结果，任选一类，比如第 K 类，结果作为" 正例"，将其分别与其它 $K-1$ 类 (作为" 负例")，做 $K-1$ 次二分类 logistic 回归:

$$\ln \frac{\Pr(y=1)}{\Pr(y=K)} = \mathbf{w}_1^T X$$

$$\ln \frac{\Pr(y=2)}{\Pr(y=K)} = \mathbf{w}_2^T X$$

$$\cdots\cdots$$

$$\ln \frac{\Pr(y=K-1)}{\Pr(y=K)} = \mathbf{w}_{K-1}^T X$$

- 两边取 $\exp(\cdot)$, 再保证概率之和为 1, 则有

$$\Pr(y=K)=1-\sum_{k=1}^{K-1}\Pr(y=k)=1-\sum_{k=1}^{K-1}\Pr(y=K)e^{\mathbf{w}_k^T X}$$

$$\Longrightarrow \qquad \Pr(y=K)=\frac{1}{1+\sum_{k=1}^{K-1}e^{\mathbf{w}_k^T X}}$$

- 进而就能计算出其它概率，从而得到最终多项 Logistic 回归模型：

$$\begin{cases} \Pr(y=1) = \frac{e^{\mathbf{w}_1^T X}}{1+\sum_{k=1}^{K-1} e^{\mathbf{w}_k^T X}} \\ \cdots\cdots \\ \Pr(y=K-1) = \frac{e^{\mathbf{w}_{K-1}^T X}}{1+\sum_{k=1}^{K-1} e^{\mathbf{w}_k^T X}} \\ \Pr(y=K) = \frac{1}{1+\sum_{k=1}^{K-1} e^{\mathbf{w}_k^T X}} \end{cases}$$

三. Softmax 回归

- 二分类 Logistic 回归自然推广到多分类就是 Softmax 回归，它是更均衡的做法。
- 将前面多项 Logistic 回归推导公式改写：

$$\ln \Pr(y=1) = \mathbf{w}_1^T X - \ln z$$

$$\ln \Pr(y=2) = \mathbf{w}_2^T X - \ln z$$

$$\cdots\cdots$$

$$\ln \Pr(y=K) = \mathbf{w}_K^T X - \ln z$$

- 同样需要保证：

$$\sum_{k=1}^{K} \Pr(y = k) = 1$$

- 这可以推出

$$z = \sum_{k=1}^{K} e^{\mathbf{w}_k^T X}$$

- 进而，可以得到 Softmax 回归模型：

$$\Pr(y = k) = \frac{e^{\mathbf{w}_k^T X}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T X}}, \qquad k = 1, \cdots, K$$

- 于是，Softmax 回归即线性回归再接一个 softmax 函数：

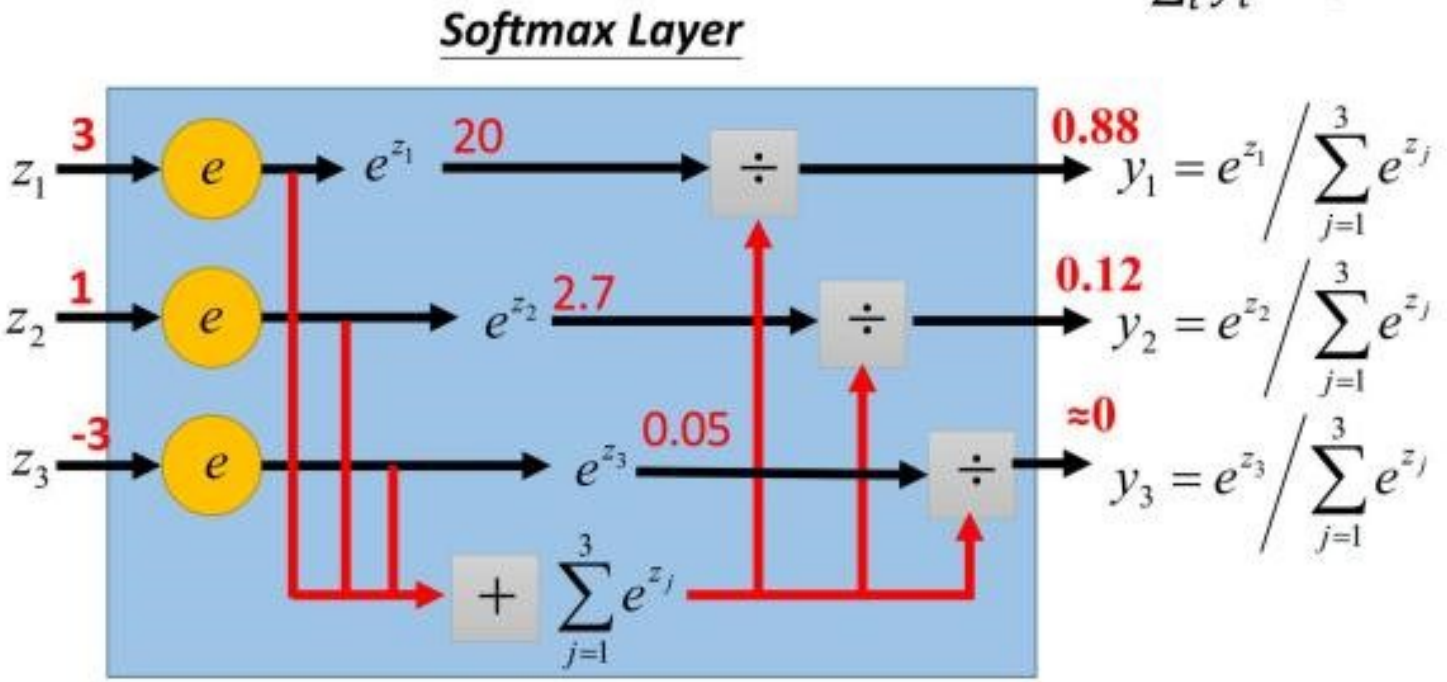
$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}, \qquad i = 1, \cdots, K$$

- softmax 函数广泛应用于深度学习，是从连续值到多分类值的激活函数，深度学习中的分类任务，最后都需要接一个这样是 softmax 层。

- Softmax layer as the output layer

Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

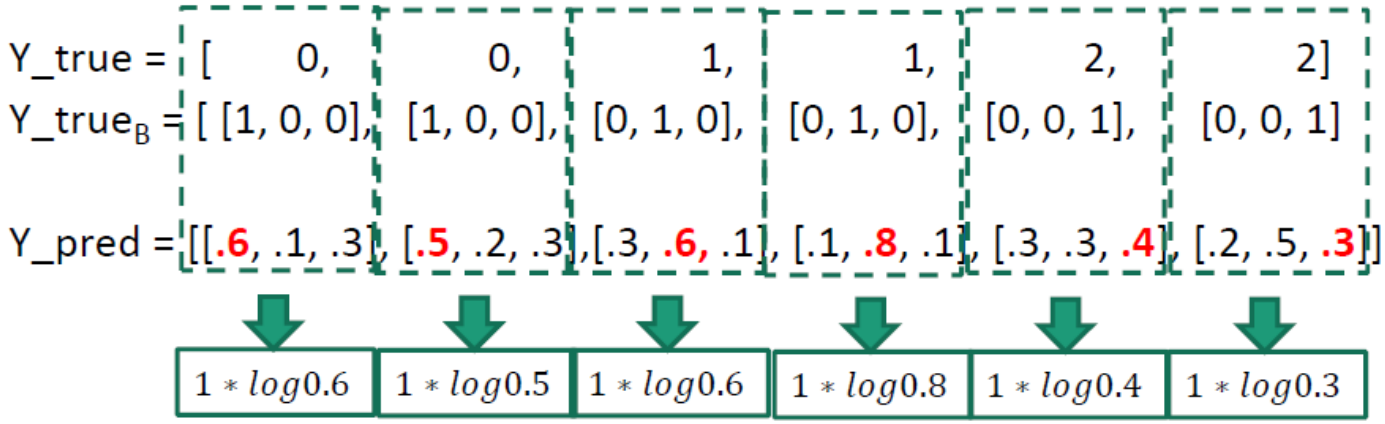


- softmax 回归的损失函数是对数似然损失：

$$J(\mathbf{W}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln \hat{y}_i$$

其中，y_i 为第 i 个样本虚拟变量表示的真实类别，$\hat{y}_i$ 为第 i 个样本预测属于每个类别的概率向量。

● **举例：** 6个样本，三个类



则$L_{\log}(Y,P) = -(log0.6 + log0.5 + log0.6 + log0.8 + log0.4 + log0.3)/6$=0.6763

```r
library(tibble)
y = sjmisc::to_dummy(tibble(c(0,0,1,1,2,2)), 1)
yh = tribble(~V1, ~V2, ~V3,
             0.6, 0.1, 0.3,
             0.5, 0.2, 0.3,
             0.3, 0.6, 0.1,
             0.1, 0.8, 0.1,
             0.3, 0.3, 0.4,
             0.2, 0.5, 0.3)
res = apply(y * log(yh), 1, sum)
- mean(res)
#> [1] 0.676
```

- **注** 1. 二分类 Logistic 回归的交叉熵损失，如果用虚拟变量表示，就是多分类对数似然损失的特例。
- **注** 2. 二项 Logistic 回归与 Softmax 回归的最优参数，都是通过对损失函数应用梯度下降法计算的，选用 sigmoid 函数与 softmax 函数的另一优势就是它们的导数形式特别简洁易算。

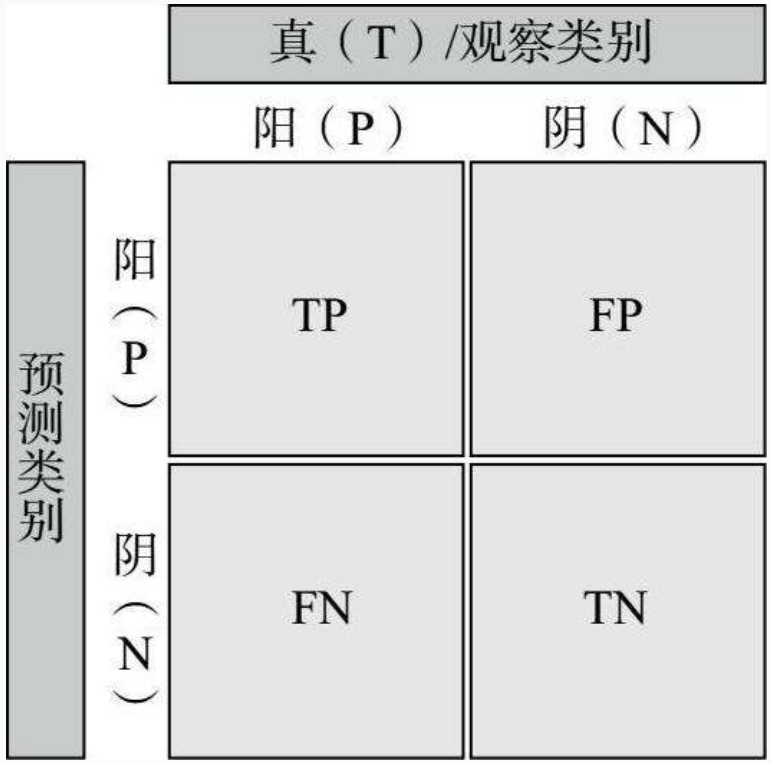
四. 二分类模型评估指标

- 准确率（Accuracy）
- 混淆矩阵（Confusion Matrix）
- 精确率（Precision）
- 召回率（Recall）
- F 得分（F-Score）
- ROC 曲线
- AUC
- PR 曲线

准确率（Accuracy）

$$Accuracy = \frac{n_{correct}}{n_{total}}$$

混淆矩阵（Confusion Matrix)



- TP（True Positive，真正)：将正类预测为正类数
- TN（True Negative，真负)：将负类预测为负类数
- FP（False Positive，假正)：将负类预测为正类数误报（Type I error)
- FN（False Negative，假负)：将正类预测为负类数–> 漏报 (Type II error)

查准率（Precision）

- 表示被分为正例的示例中实际为正例的比例

$$Perecision = \frac{TP}{TP + FP}$$

召回率（Recall）

- 是覆盖面的度量，度量有多少个正例被正确地分为正例

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

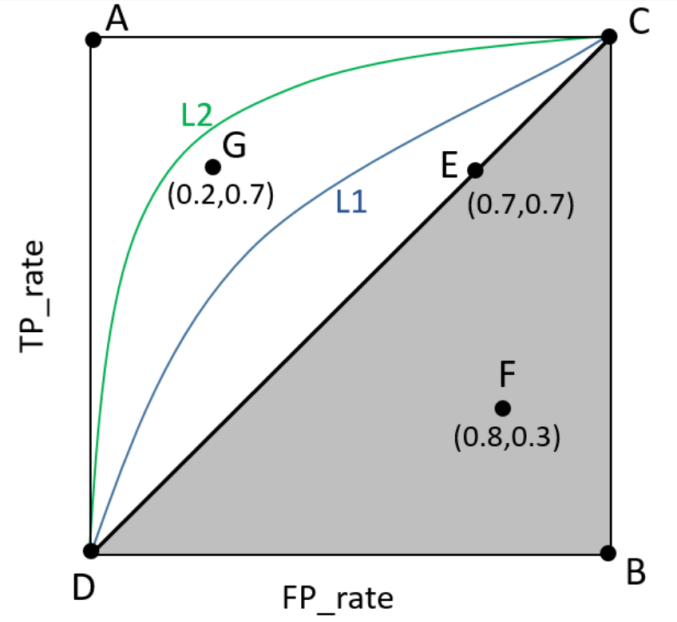
F1 得分（F-Score）

- Precision 和 Recall 一种加权组合，综合了查准率和召回率的结果，当 F1 较高时说明试验方法比较有效

$$F1 = 2 \cdot \frac{Precision * Recall}{Precision + Recall}$$

ROC 曲线和 AUC

- ROC 曲线在不同分类阈值上对比真正率（TP_rate）与假正率（FP_rate）的曲线，ROC 曲线下面的面积叫做 AUC：



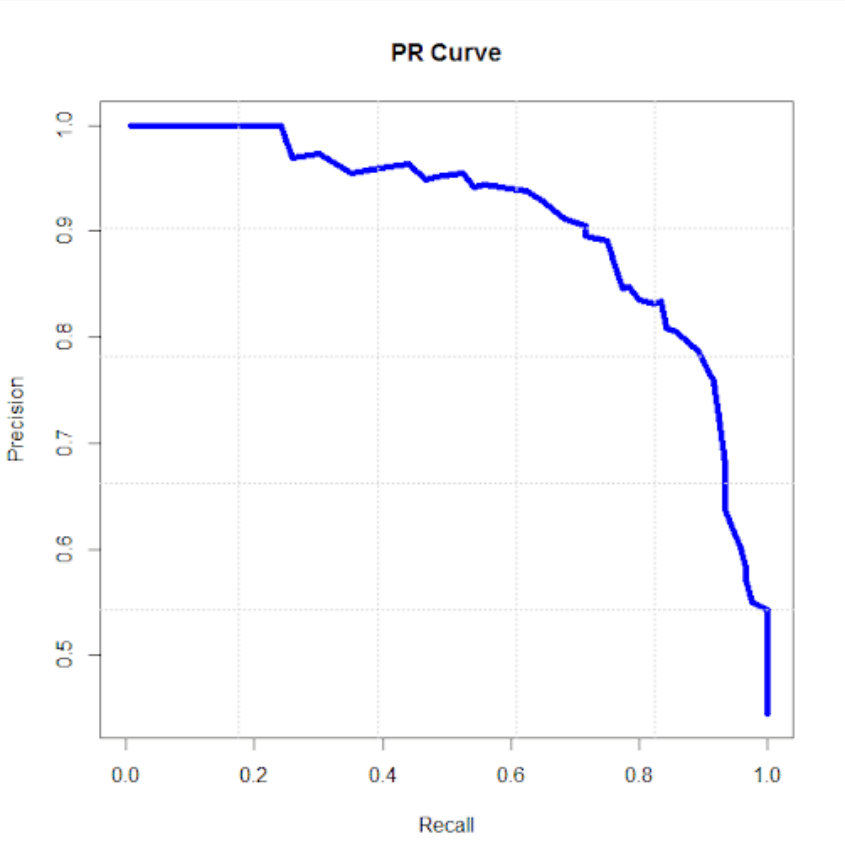
$$TP_{rate} = \frac{TP}{TP + FN}, \qquad FP_{rate} = \frac{FP}{FP + TN}$$

- ROC 曲线越接近左上角（AUC 面积越大），表示分类性能越好；
- ROC 曲线的优点是，对类分布/不平衡数据不敏感；正例负例的占比变化，ROC 曲线不变。

PR（Precision-Recall）曲线

- 若负例数远大于正例数，若 FP 很大，即有很多负例被预测为正例，用 ROC 曲线则会判断其性能很好，但是实际上其性能并不好。
- 此时应改用 PR 曲线，因为 Precision 综合考虑了 TP 和 FP 的值，因此在极度不平衡的数据下（正例的样本较少），PR 曲线可能比 ROC 曲线更实用。

- PR 曲线，即在不同分类阈值上，对比查准率（Precision）与召回率（Recall）的曲线，越靠近右上角越好：



五. Logistic 回归模型实例

- 以 German Credit 数据集为例，这是根据个人的银行贷款信息和申请客户贷款逾期发生情况来预测贷款违约倾向的数据集，数据集包含 1000 个样本，21 个变量。因变量 credit_risk 为二分类变量（不均衡，700 个 good / 300 个 bad）。
- 假设银行将对预测结果为 good 发放贷款，若预测正确将产生 35% 的利息；若预测错误将导致损失 100% 的本金。
- 本案例将讨论 Logistic 回归建模的一般流程，以及针对代价敏感和不平衡分类进行阈值调参。

- 改用代价（损失）来描述：

```
costs = matrix(c(-0.35, 0, 1, 0), nrow = 2)
dimnames(costs) = list(response = c("good", "bad"),
                       truth = c("good", "bad"))
costs
#>          truth
#> response  good bad
#>     good -0.35   1
#>     bad   0.00   0

# 若不给任何人发放贷款
(700 * costs[2, 1] + 300 * costs[2, 2]) / 1000
#> [1] 0
```

```r
# 若给所有人发放贷款
(700 * costs[1, 1] + 300 * costs[1, 2]) / 1000
#> [1] 0.055
```

- 假设平均贷款为 20000 美元, 那么给所有人放贷, 将损失将超过 100 万美元:

```r
0.055 * 20000 * 1000
#> [1] 1100000
```

- 希望代价越小越好。这属于” 代价-敏感” 分类问题。

1. 创建任务

- 该数据集已是 mlr3 包的内置任务，直接选取该任务即可：

```
library(mlr3verse)
gc_task = tsk("german_credit")
gc_task
#> <TaskClassif:german_credit> (1000 x 21): German Credit
#> * Target: credit_risk
#> * Properties: twoclass
#> * Features (20):
#>   - fct (14): credit_history, employment_duration, foreign
#>     housing, job, other_debtors, other_installment_plans,
#>     people_liable, personal_status_sex, property, purpose,
#>     status, telephone
#>   - int (3): age, amount, duration
#>   - ord (3): installment_rate, number_credits, present_res
```

2. 探索数据 (略)

```
credit = gc_task$data()
dim(credit)
#> [1] 1000   21
table(credit$credit_risk)
#>
#> good  bad
#>  700  300
summary(credit)
#>  credit_risk      age             amount
#>  good:700    Min.   :19.0   Min.   :  250
#>  bad :300    1st Qu.:27.0   1st Qu.: 1366
#>              Median :33.0   Median : 2320
#>              Mean   :35.5   Mean   : 3271
#>              3rd Qu.:42.0   3rd Qu.: 3972
#>              Max.   :75.0   Max.   :18424
```

3. 选择学习器

```
learner = lrn("classif.log_reg")
learner$param_set$ids()
#>  [1] "dispersion"  "epsilon"     "etastart"    "maxit"
#>  [6] "mustart"     "offset"      "singular.ok" "start"
#> [11] "x"           "y"
```

注：可用 `lrns()` 查看所有可用的学习器，更多学习器在 `mlr3extralearners` 包。

4. 划分训练集测试集

- 做留出 (holdout) 重抽样，80% 作为训练集，其余 20% 作为测试集
- 为了保持训练集、测试集的因变量数据具有相似的分布，采用分层抽样方法
- 用 partition( ) 函数对任务做划分，默认按因变量分层，取出训练集索引和测试集索引

```
set.seed(123)
split = partition(gc_task, ratio = 0.8)
# 默认 stratify = TRUE
```

5. 训练模型

```
learner$train(gc_task, row_ids = split$train)
coef(learner$model)                    # 提取回归系数
#>                                                    (Intercept)
#>                                                       -1.84735
#>                                                            age
#>                                                        0.00790
#>                                                         amount
#>                                                       -0.00013
#>    credit_historycritical account/other credits elsewhere
#>                                                        0.00608
#> credit_historyno credits taken/all credits paid back duly
#>                                                        0.61689
#>    credit_historyexisting credits paid back duly till now
#>                                                        1.05449
#>     credit_historyall credits at this bank paid back duly
```

6. 模型预测

```
prediction = learner$predict(gc_task, row_ids = split$test)
prediction
#> <PredictionClassif> for 200 observations:
#>     row_ids truth response
#>           1  good     good
#>           4  good     good
#>          15  good      bad
#> ---
#>         980   bad      bad
#>         981   bad     good
#>         982   bad      bad
```

```
confusion = prediction$confusion  # 混淆矩阵
confusion
#>          truth
#> response good bad
#>     good  121  26
#>     bad    19  34
```

注： 用 msrs() 可查看所有可用的模型评估指标。

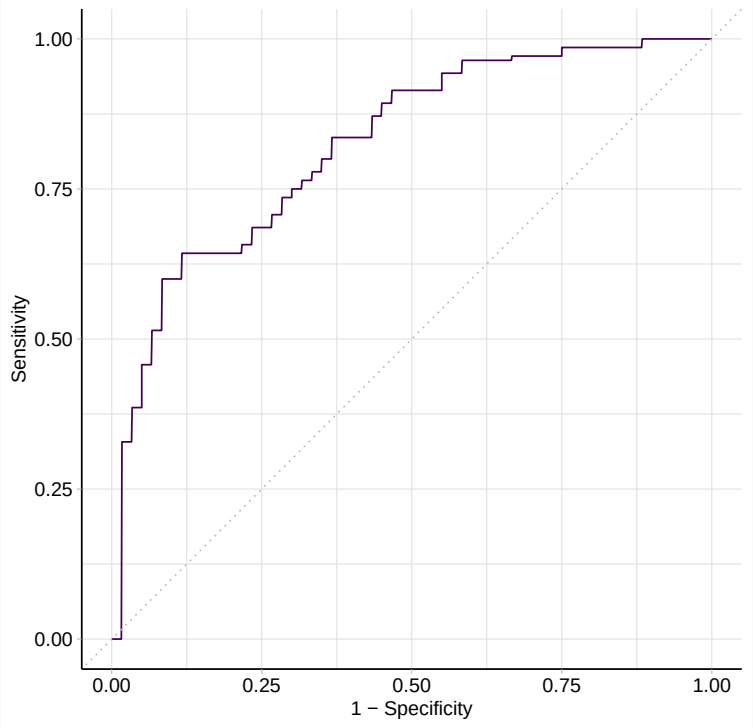
```r
prediction$score(msr("classif.acc"))      # 准确率
#> classif.acc
#>       0.775
prediction$score(msr("classif.recall"))   # 召回率
#> classif.recall
#>          0.864

avg_costs = sum(confusion * costs) / length(split$test)
avg_costs
#> [1] -0.0817
avg_costs * 20000 * 1000
#> [1] -1635000
```

- 用”代价-敏感”方式评估模型：

```
cost_msr = msr("classif.costs", costs = costs)
prediction$score(cost_msr, task = gc_task)
#> classif.costs
#>       -0.0817
```

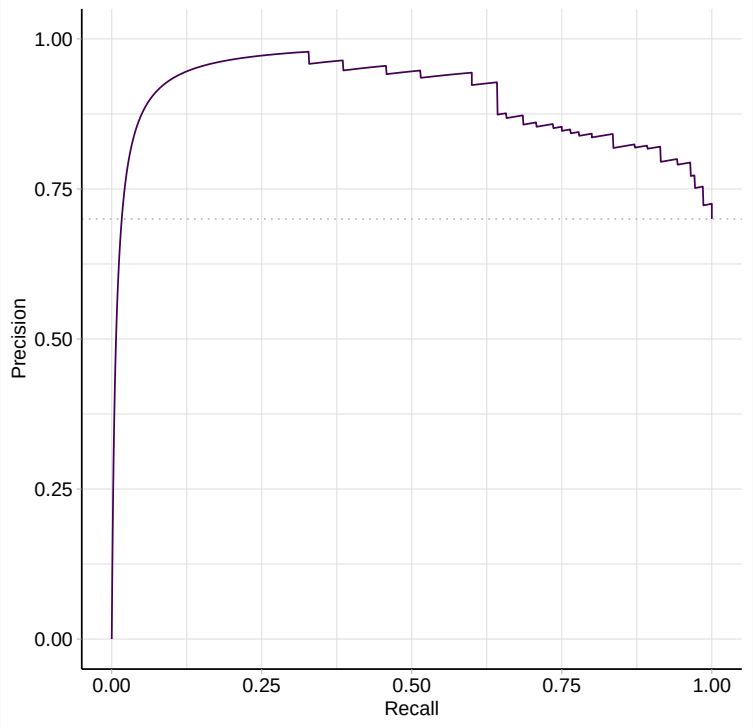
```
learner$predict_type = "prob"
learner$train(gc_task, row_ids = split$train)
prediction = learner$predict(gc_task, row_ids = split$test)
prediction
#> <PredictionClassif> for 200 observations:
#>     row_ids truth response prob.good prob.bad
#>           1  good     good     0.952   0.0479
#>           4  good     good     0.662   0.3385
#>          15  good      bad     0.376   0.6243
#> ---
#>         980   bad      bad     0.365   0.6352
#>         981   bad     good     0.765   0.2351
#>         982   bad      bad     0.431   0.5688
```

```r
autoplot(prediction, type = "roc")          # ROC 曲线
```



```r
prediction$score(msr("classif.auc"))      # AUC 面积
#> classif.auc
#>       0.824
```

```
autoplot(prediction, type = "prc")          # PR 曲线
```



8. 阈值调参

- 对于信用卡数据的二分类，主要是想要最小化这类错误：模型预测为"good"，但实际是"bad"（即假正的数量），因为这是更昂贵的错误。
- 若将阈值增加到大于 0.5 的值，则会减少假正的数量。注意，这同时也增加了假负的数量，只不过对这种代价不敏感。
- 定义函数计算不同阈值的代价：

```
with_threshold = function(th) {
  prediction$set_threshold(th)
  prediction$score(measures = cost_msr, task = gc_task)
}
```

```
with_threshold(0.5)
#> classif.costs
#>       -0.0817
with_threshold(0.75)
#> classif.costs
#>       -0.0942
```

- 将不同阈值的代价函数，传递给 optimize() 寻优, 以在 [0.5, 1] 中找到最小值

```
best = optimize(with_threshold, c(0.5, 1))
best
#> $minimum
#> [1] 0.788
#>
#> $objective
#> classif.costs
#>        -0.122
-0.12725 * 20000 * 1000
#> [1] -2545000
```

主要参考文献

[1] mlr3book. 2021. https://mlr3book.mlr-org.com/

[2] Jim Liang(梁劲). Getting Started with Machine Learning, 2019

[3] Wikipedia. Multinomial logistic regression.
https://en.wikipedia.org/wiki/Multinomial_logistic_regression

[4] A. Kassambara, Machine Learning Essential: Practiale Guide in R. 2017.
http://www.sthda.com