北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Data Analytics with R

R语言数据分析

既是世间法、自当有分别

艾新波 / 2018 • 北京

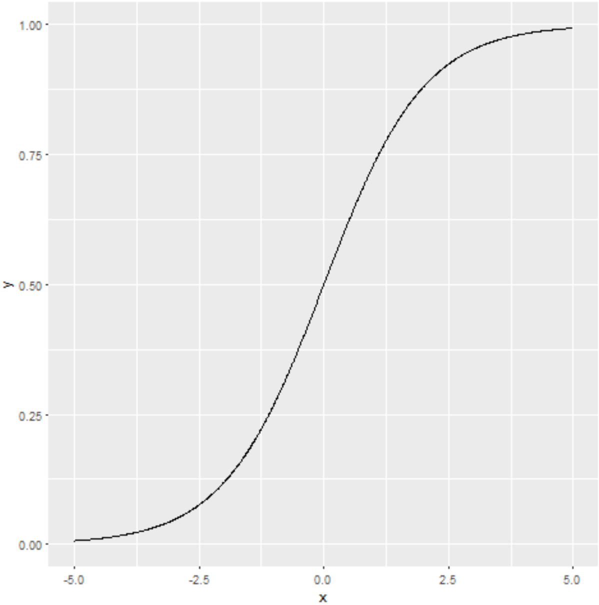
课程体系

- R语言数据分析
  - 上部：论道 
    - 第1章 气象万千、数以等观
    - 第2章 所谓学习、归类而已
    - 第3章 格言联璧话学习
    - 第4章 源于数学、归于工程
  - 中部：执具 
    - 第5章 工欲善其事必先利其器
    - 第6章 基础编程
    - 第7章 数据对象
  -
    - 第8章 人人都爱tidyverse
    - 第9章 最美不过数据框
  - 下部 博术 
    - 第10章 观数以形
    - 第11章 相随相伴、谓之关联
    - **第12章 既是世间法、自当有分别**
    - 第13章 方以类聚、物以群分
    - 第14章 庐山烟雨浙江潮

朴素贝叶斯：算算概率的大小

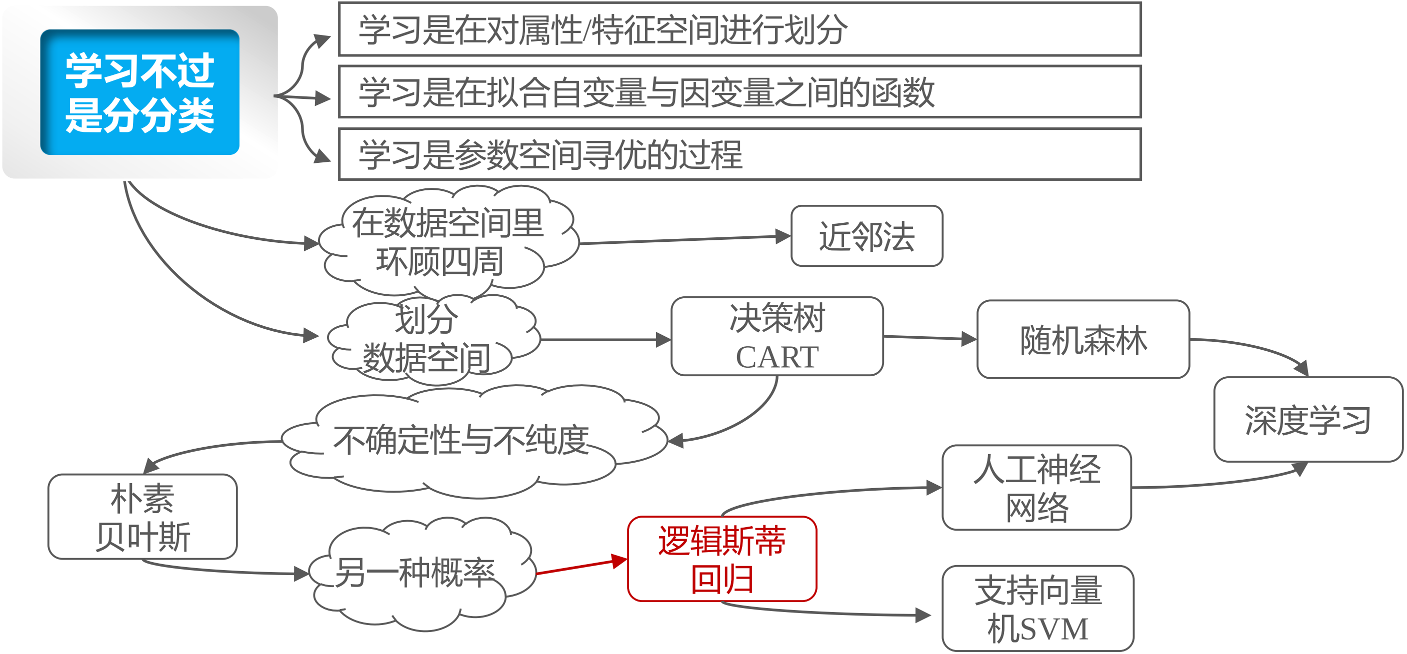
$$p(y) \rightarrow p(y|X)$$

另一种概率



$$p = \frac{1}{1 + e^{-x}}$$

算法模型



学习不过是分分类
学习是在对属性/特征空间进行划分
学习是在拟合自变量与因变量之间的函数
学习是参数空间寻优的过程
在数据空间里环顾四周
近邻法
划分数据空间
决策树CART
随机森林
深度学习
不确定性与不纯度
朴素贝叶斯
人工神经网络
另一种概率
逻辑斯蒂回归
支持向量机SVM

逻辑斯蒂回归

分类： 根据所具有的属性、特征，作出判断！

要综合利用好属性、特征信息，一个简单的办法是把他们“加”起来：

$$z = info(x) = w_0 + w_1x_1 + \cdots + w_mx_m$$

目标： $info(x) \rightarrow [0,1]$

但是： $info(x) \in [-\infty, +\infty]$

关键：把$info(x)$**挤压进**$[0,1]$**区间**

做法：引入挤压函数： $p(y=1|x) = \frac{1}{1+e^{-info(x)}} = \frac{1}{1+e^{-z}} = \phi(z) \in [0,1]$

逻辑斯蒂回归

落入[0, 1]区间的函数那么多

为何偏偏选这个？落入[0, 1]只是必要条件而已

Biometrika (1967), 54, 1 *and* 2, *p.* 167 167
Printed in Great Britain

Estimation of the probability of an event as a function of several independent variables

By STROTHER H. WALKER† and DAVID B. DUNCAN
Johns Hopkins University

Summary

A method for estimating the probability of occurrence of an event from dichotomous or polychotomous data is developed, using a recursive approach. The method in the dichotomous case is applied to the data of a 10-year prospective study of coronary disease. Other areas of application are briefly indicated.

1. Introduction

The purpose of this paper is to develop a method for estimating from dichotomous (quantal) or polychotomous data, the probability of occurrence of an event as a function of a relatively large number of independent variables. A key feature of the method is a recursive approach based on Kalman's work (Kalman, 1960 and unpublished report) in linear dynamic filtering and prediction, derivable also from the work of Swerling (1959), which provides an example of many other possible uses of recursive techniques in non-linear estimation and in related areas.

The problem that motivated the investigation is a central one in the epidemiology of coronary heart disease, and it will be used to fix ideas and illustrate the method. Some indication of the range of applications will be given in the conclusion.

In the light of present medical knowledge a reasonable assumption is that P follows a symmetric sigmoid curve...

Source: Walker, SH; Duncan, DB (1967). "Estimation of the probability of an event as a function of several independent variables". Biometrika. 54: 167–178.

逻辑斯蒂回归

定义事件发生比odds——发生与不发生的比值：

$$odd\boldsymbol{s} = \frac{p(y=1|x)}{1-p(y=1|x)} = e^{w^T x}$$

两边同时取对数：

$$\ln(odd\boldsymbol{s}) = \ln\frac{p(y=1|x)}{1-p(y=1|x)} = z = info(x)$$

显然，此时的w_i可以看做是相应的x_i增加一个单位，发生比在对数尺度上的变化

逻辑斯蒂回归，不过是logit变换下的一个线性模型罢了——执简驭繁😊

逻辑斯蒂回归

每一个观测到的样本$(x^{(i)}, y_i)$出现的概率是：

$$p(x^{(i)}, y_i) = p(y_i|x^{(i)})^{y_i}\left(1 - p(y_i|x^{(i)})\right)^{1-y_i} p(x^{(i)})$$

m**个独立样本出现的似然函数为：**

$$\prod_{i=1}^{m} p(x^{(i)}, y_i) = \prod_{i=1}^{m} p(y_i|x^{(i)})^{y_i}\left(1 - p(y_i|x^{(i)})\right)^{1-y_i} \prod_{i=1}^{m} p(x^{(i)})$$

显然，$p(x^{(i)})$与待估参数无关

$$\prod_{i=1}^{m} p(x^{(i)}, y_i) \propto \prod_{i=1}^{m} p(y_i|x^{(i)})^{y_i}\left(1 - p(y_i|x^{(i)})\right)^{1-y_i}$$

逻辑斯蒂回归

极大似然估计的基本原理：

事件A发生的概率与参数w相关，A发生的概率记为$P(A,w)$。若在一次实验中，观察到事件A。那我们认为，这个时候的w应该是最有利于A出现的w。也就是在各种各样的w中，应该是使得$P(A,w)$最大的w，才是比较合理的

同样：在我们前述的问题中，使得概率似然函数取值最大的参数，就是我们要估计的参数。

参数估计=模型的选择=模型的学习

逻辑斯蒂回归

重新审视似然函数：

$$L(w)=\prod_{i=1}^{m}p\left(y_i|x^{(i)}\right)^{y_i}\left(1-p\left(y_i|x^{(i)}\right)\right)^{1-y_i}$$

显然，这里的连乘，是不受欢迎的：

对数的发明者约翰·纳皮尔曾说过：看起来在数学实践中，最麻烦的莫过于大数字的乘法、除法、开平方和开立方，计算起来特别费事又伤脑筋，于是我开始构思有什么巧妙好用的方法可以解决这些问题……。

转引自：Eli Maor. e的故事：一个常数的传奇。周昌智，毛兆荣译。人民邮电出版社，2010年。pp.1

逻辑斯蒂回归

$$l(w)=\ln L(w)=\ln\left(\prod_{i=1}^{m}p\left(y_i\mid x^{(i)}\right)^{y_i}\left(1-p\left(y_i\mid x^{(i)}\right)\right)^{1-y_i}\right)$$

$$=\ln\left(\prod_{i=1}^{m}\phi\left(z^{(i)}\right)^{y_i}\left(1-\phi\left(z^{(i)}\right)\right)^{1-y_i}\right)$$

$$=\sum_{i=1}^{m}\left(y_i\ln\left(\phi\left(z^{(i)}\right)\right)+\left(1-y_i\right)\ln\left(1-\phi\left(z^{(i)}\right)\right)\right)$$

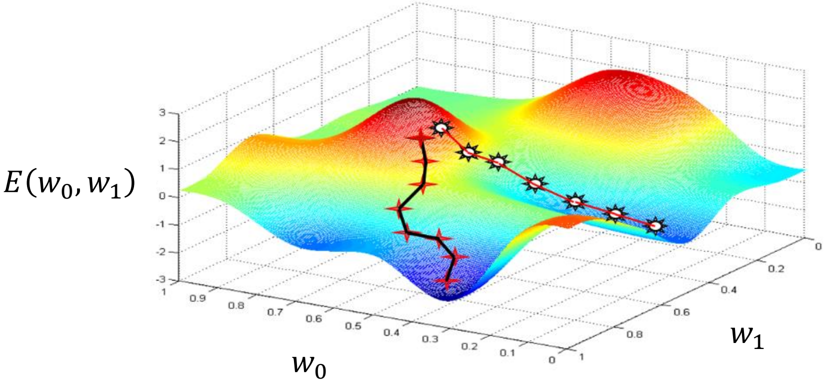
上式增加一个负号，可以作为代价函数

对于参数的估计，要么求似然函数最大、要么让代价函数最小，二者是完全等价

逻辑斯蒂回归

要求得似然/代价函数的最大/最小值，可以通过梯度上升/下降的方法

千里之行、始于随机：先随机找一个w，然后沿着$l(w)$增加最快的方向，不断迈进



学习过程犹如瞎子爬山，只清楚脚下的情况——根据梯度来上升/下降

图片引自Andrew Ng《Machine Learning》公开课，作了修改

逻辑斯蒂回归

要求得似然/代价函数的最大/最小值，可以通过梯度上升/下降的方法

千里之行、始于随机：先随机找一个w，然后沿着$l(w)$增加最快的方向，不断迈进

$$\frac{\partial l(\mathbf{w})}{\partial w_i}=\frac{\partial}{\partial w_i}\left(\sum_{i=1}^{m}\left(y_i\ln\left(\phi\left(z^{(i)}\right)\right)+\left(1-y_i\right)\ln\left(1-\phi\left(z^{(i)}\right)\right)\right)\right)$$

$$=\sum_{i=1}^{m}\left(y_i\frac{1}{\phi\left(z^{(i)}\right)}-\left(1-y_i\right)\frac{1}{1-\phi\left(z^{(i)}\right)}\right)\frac{\partial\phi\left(z^{(i)}\right)}{\partial w_i}$$

$$=\sum_{i=1}^{m}\left(y_i\frac{1}{\phi\left(z^{(i)}\right)}-\left(1-y_i\right)\frac{1}{1-\phi\left(z^{(i)}\right)}\right)\phi\left(z^{(i)}\right)\left(1-\phi\left(z^{(i)}\right)\right)\frac{\partial z^{(i)}}{\partial w_i}$$

$$=\sum_{i=1}^{m}\left(y_i\left(1-\phi\left(z^{(i)}\right)\right)-\left(1-y_i\right)\phi\left(z^{(i)}\right)\right)x_j^{(i)}$$

$$=\sum_{i=1}^{m}\left(y_i-\phi\left(z^{(i)}\right)\right)x_j^{(i)}$$

逻辑斯蒂回归

要求得似然/代价函数的最大/最小值，可以通过梯度上升/下降的方法

千里之行、始于随机：先随机找一个w，然后沿着$l(w)$增加最快的方向，不断迈进

梯度方向是增加最快的方向：$\frac{\partial l(w)}{\partial w_i} = \sum_{i=1}^{m} \left(y_i - \phi\left(z^{(i)}\right)\right) x_j^{(i)}$

权值更新：$w_j \leftarrow w_j + \lambda \sum_{i=1}^{m} \left(y_i - \phi\left(z^{(i)}\right)\right) x_j^{(i)}$

不断进行迭代，直至满足终止条件，如：当两次迭代之间的差值小于某个阈值；或是已经达到预先设定的最大迭代次数

R语言实现

```
set.seed(2012)
imodel <- glm(wlfk ~ ., data = cjb[train_set_idx,],
              family = binomial(link = "logit"))
predicted_logit <- predict(imodel,
                           newdata = cjb[train_set_idx,],
                           type = "response")
predicted_train <-
  rep(levels(cjb$wlfk)[2], length(train_set_idx))
predicted_train[predicted_logit < 0.5] <- levels(cjb$wlfk)[1]
```

R语言实现

```
Metrics::ce(cjb$wlfk[train_set_idx], predicted_train)
#> [1] 0.2181146
predicted_logit <- predict(imodel,
                           newdata = cjb[-train_set_idx, ],
                           type = "response")
predicted_test <-
  rep(levels(cjb$wlfk)[2], nrow(cjb[-train_set_idx,]))
predicted_test[predicted_logit < 0.5] <-
  levels(cjb$wlfk)[1]
Metrics::ce(cjb$wlfk[-train_set_idx], predicted_test)
#> [1] 0.1888412
```

R语言实现

```
#找到最好的分隔阈值
best_threshold <- NA
min_err <- Inf
cur_threshold <- 0.1
for (cur_threshold in seq(0.1, 0.9, by = 0.001)) {
  predicted_test <-
    rep(levels(cjb$wlfk)[2], nrow(cjb[-train_set_idx,]))
  predicted_test[predicted_logit < cur_threshold] <-
    levels(cjb$wlfk)[1]
```

R语言实现

```
  cur_err <- Metrics::ce(cjb$wlfk[-train_set_idx],
                             predicted_test)
  if (cur_err < min_err) {
    best_threshold <- cur_threshold
    min_err <- cur_err
  }
}
best_threshold
#> [1] 0.592
```

R语言实现

```r
sp <- Sys.time() #记录开始时间
cat("\n[Start at:", as.character(sp))
for (i in 1:length(kfolds)) {
  curr_fold <- kfolds[[i]] #当前这一折
  train_set <- cjb[-curr_fold,] #训练集
  test_set <- cjb[curr_fold,] #测试集
  imodel_kfold <- glm(wlfk~.,
                      data = train_set,
                      family=binomial(link="logit"))
  predicted_logit <- predict(imodel_kfold,
                             newdata = train_set,
                             type = "response")
  predicted_train <- rep(levels(cjb$wlfk)[2],
                         nrow(train_set))
```

R语言实现

```
  predicted_train[predicted_logit < best_threshold] <-
    levels(cjb$wlfk)[1]
  imetrics("LogisticRegression", "Train",
           predicted_train, train_set$wlfk)
  predicted_logit <- predict(imodel_kfold,
                             newdata = test_set,
                             type = "response")
  predicted_test <- rep(levels(cjb$wlfk)[2], nrow(test_set))
  predicted_test[predicted_logit < best_threshold] <-
    levels(cjb$wlfk)[1]
  imetrics("LogisticRegression", "Test",
           predicted_test, test_set$wlfk)
}
ep <- Sys.time()
cat("\tFinised at:", as.character(ep), "]\n")
```

R语言实现

```
#> 81  LogisticRegression Train 0.7959770  0.2040230
#> 82  LogisticRegression  Test 0.7692308  0.2307692
#> 83  LogisticRegression Train 0.7902299  0.2097701
#> 84  LogisticRegression  Test 0.8076923  0.1923077
#> 85  LogisticRegression Train 0.7945402  0.2054598
#> 86  LogisticRegression  Test 0.8205128  0.1794872
#> 87  LogisticRegression Train 0.7916667  0.2083333
#> 88  LogisticRegression  Test 0.8076923  0.1923077
#> 89  LogisticRegression Train 0.7919656  0.2080344
……
#> 96  LogisticRegression  Test 0.7012987  0.2987013
#> 97  LogisticRegression Train 0.8048780  0.1951220
#> 98  LogisticRegression  Test 0.7142857  0.2857143
#> 99  LogisticRegression Train 0.7905308  0.2094692
#> 100 LogisticRegression  Test 0.7922078  0.2077922
```

谢谢聆听

Thank you

教师个人联系方式

艾新波

手机：13641159546

QQ：23127789

微信：13641159546

E-mail: 13641159546@126.com

axb@bupt.edu.cn

地址：北京邮电大学科研楼917室

课程网址：https://github.com/byaxb/RDataAnalytics