Logistic regression with ScaleR

Revolution Analytics

Goals and Agenda

We will cover logistic regression in this session.

We will focus on how to use the `rxLogit()` function, and we will review how to interact with results once they have been estimated.

Outline

dir config

```
## dir config
big.data.path <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
if (big.data.path == "") {
  Sys.setenv(ACADEMYR_BIG_DATA_PATH = "/usr/share/BigData")
  big.data.path <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
}
data.path <- "../data"
output.path <- "../output/xdf"
if (!file.exists(output.path)) dir.create(output.path, recursive = TRUE)
sample.data.dir <- rxGetOption("sampleDataDir")
```

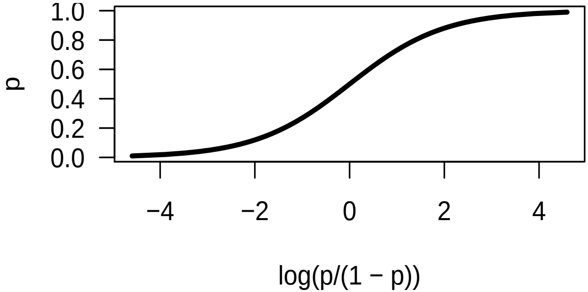
Logistic Regression

`Goal` Predict the probability of some binary outcome with a linear model

Logistic Regression

Estimates the log odds-ratio as a function of a linear combination of predictor variables:

$$ln(\frac{p_i}{1-p_i}) = b_0 + b_1X_1 + ...b_mX_m$$



Outline

Implementation in RRE

`rxLogit` implements logistic regression using the Iteratively Reweighted Least Squares algorithm (i.e. full maximum likelihood)

rxLogit() Usage

```
args(rxLogit)
```

```
## function (formula, data, pweights = NULL, fweights = NULL, cube = FALSE,
##     cubePredictions = FALSE, variableSelection = list(), rowSelection = NULL,
##     transforms = NULL, transformObjects = NULL, transformFunc = NULL,
##     transformVars = NULL, transformPackages = NULL, transformEnvir = NULL,
##     dropFirst = FALSE, dropMain = rxGetOption("dropMain"), covCoef = FALSE,
##     covData = FALSE, covariance = FALSE, initialValues = NULL,
##    coefLabelStyle = rxGetOption("coefLabelStyle"), blocksPerRead = rxGetOption("blocksPerRead"),
...
```

Virtually the same as `rxLinMod()`

Dataset Setup

`mortgages` dataset: We will attempt to predict `default` status.

```
mortgages <- file.path(sample.data.dir, "mortDefaultSmall.xdf")
rxGetVarInfo(mortgages)
```

```
## Var 1: creditScore, Type: integer, Low/High: (470, 925)
## Var 2: houseAge, Type: integer, Low/High: (0, 40)
## Var 3: yearsEmploy, Type: integer, Low/High: (0, 14)
## Var 4: ccDebt, Type: integer, Low/High: (0, 14094)
## Var 5: year, Type: integer, Low/High: (2000, 2009)
## Var 6: default, Type: integer, Low/High: (0, 1)
```

Learning to Use `rxLogit()`

Use the `mortgages` default file to build a logistic regression for the probability of default.

- Use `year` as a factor variable.
- Show that there was a difference in probability of default between 2006 and 2009.

```
myformula <- formula(default ~ F(year) + ccDebt + creditScore + houseAge +
  yearsEmploy)
```

Outline

Predicting Values

You still use `rxPredict` to generate predictions.

```
args(rxPredict)
```

```
## function (modelObject, data = NULL, ...)
## NULL
```

Prediction Exercise

Use the estimated model to predict the likelihood of default for `new_mortgages`.

```
new_mortgages <- data.frame(year = rep(c(2006, 2009), each = 4), ccDebt = rep(c(1000,
  10000), 4), creditScore = rep(c(700, 800), 4), houseAge = rep(c(1,
  5, 10, 20), 2), yearsEmploy = rep(7, 8))
str(new_mortgages)
```

```
## 'data.frame':    8 obs. of  5 variables:
##  $ year       : num  2006 2006 2006 2006 2009 ...
##  $ ccDebt     : num  1000 10000 1000 10000 1000 10000 1000 10000
##  $ creditScore: num  700 800 700 800 700 800 700 800
##  $ houseAge   : num  1 5 10 20 1 5 10 20
##  $ yearsEmploy: num  7 7 7 7 7 7 7 7
```

REVOLUTION ANALYTICS

More on Prediction

```
args(rxPredict)
```

```
## function (modelObject, data = NULL, ...)
## NULL
```

Argument: `type`

response produces predictions on the same scale as your predictor

link Produces predictions on the scale of the linear predictor

Logistic Regression and `type`

response	estimated predicted probability of the outcome
link	estimated log-odds of the outcome

Exercise: Create Link-type Predictions

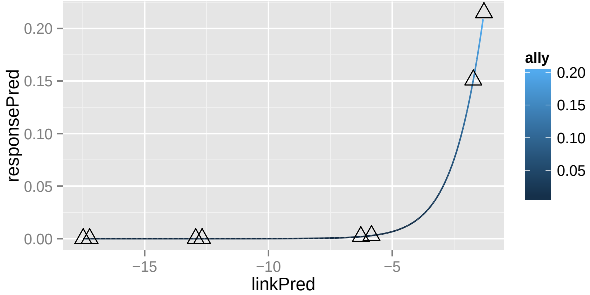
Create “link”-type predictions for the same `new_mortgages` data.

Check your Results

```
resultsObj <- file.path(data.path, "logistic_regression_with_rre_solution_objects.RData")
load(resultsObj)
```

Visualize Relationship

```
library(ggplot2)
vis.dat <- data.frame(responsePred = new_predictions$default_Pred,
  linkPred = new_predictions_link$default_Pred)
distribution.dat <- data.frame(allx = allx <- with(vis.dat, seq(min(linkPred),
  max(linkPred), by = 0.05)), ally = ally <- plogis(allx))
ggplot(data = vis.dat, mapping = aes(x = linkPred, y = responsePred)) +
  geom_line(data = distribution.dat, aes(x = allx, y = ally, col = ally)) +
  geom_point(size = 4, shape = 2) + scale_shape(solid = FALSE)
```



Outline

rxLogit() Additional Arguments

Logistic regression does not have a closed form solution, so various arguments are used to control the interation process

initialValues intiival values for the coefficients

maxIterations maximum number of iterations

coeffTolerance convergence tolerance for the coefficients

gradientTolerance convergence tolerance for the gradient

objectiveFunctionTolerance convergence tolerance for the objective function

Outline

Oversampling

One common technique with logistic regression is oversampling (or undersampling)

When you have rare events, it can be desirable to increase their relative rate of occurrence because the estimate of its probability is biased.

Oversampling Example

Trivial approach:

```
logitModelOS <- rxLogit(formula = myformula, fweights = "Fweight",
  transforms = list(Fweight = ifelse(default == 1, 10, 1)), data = mortgages)
summary(logitModelOS)
```

```
## Call:
## rxLogit(formula = myformula, data = mortgages, fweights = "Fweight",
##     transforms = list(Fweight = ifelse(default == 1, 10, 1)))
##
## Logistic Regression Results for: default ~ F(year) + ccDebt +
##     creditScore + houseAge + yearsEmploy
## Data: mortgages (RxXdfData Data Source)
...
```

Problems

1. Estimate of the intercept will be incorrect.
2. Predicted probabilities will be inaccurate.

Undersampling and Probability weights

Another approach is to undersample your non-event rows and then use appropriate weights.

First create a new variable to determine whether that entry should be used for estimation

Create the Subsample

```
(pOutFull <- rxSummary( ~ default, data = mortgages)$sDataFrame$Mean)
```

```
## [1] 0.00471
```

```
myNewmort <- file.path(output.path, "myMort.xdf")
set.seed(100)
rxDataStep(inData = mortgages, outFile = myNewmort,
           transforms = list(inSubSample = ifelse( default == 1 | runif(.rxNumRows) < pOutFull*4,
           transformObjects = list(pOutFull = pOutFull),
           overwrite = TRUE)
```

What weights are appropriate?

$$w_1 = \frac{p_1}{r_1}$$

$$w_0 = \frac{1 - p_1}{1 - r_1}$$

$$w_i = w_1 Y_i + w_0 (1 - Y_i)$$

Estimate with Weighted Model

```
pOutSub <- rxSummary(~default, data = myNewmort, rowSelection = inSubSample ==
  1)$sDataFrame$Mean
(fullDatP <- c(1 - pOutFull, pOutFull))
```

```
## [1] 0.99529 0.00471
```

```
(subDatP <- c(1 - pOutSub, pOutSub))
```

```
## [1] 0.8051303 0.1948697
```

```
logitModelUS <- rxLogit(formula = myformula, pweights = "Pweight",
  rowSelection = inSubSample == 1, transforms = list(Pweight = fullProp[default +
    1]/subProp[default + 1]), transformObjects = list(fullProp = fullDatP,
    subProp = subDatP), data = myNewmort)
```

Summarize Undersampled Model

```
summary(logitModelUS)
```

```
## Call:
## rxLogit(formula = myformula, data = myNewmort, pweights = "Pweight",
##     rowSelection = inSubSample == 1, transforms = list(Pweight = fullProp[default +
##         1]/subProp[default + 1]), transformObjects = list(fullProp = fullDatP,
##         subProp = subDatP))
##
## Logistic Regression Results for: default ~ F(year) + ccDebt +
...
```

Summary

- Use `rxLogit()` to estimate logistic regression
- Use `rxPredict()` to predict probabilities (or log-odds)
- Appropriate use of `pweights` argument can be used for over/under-sampling.

Questions?

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR