CS 660: Mathematical Foundations for Analytics

Dr. Francis Parisi

Pace University

Spring 2017

Course Overview

1. Part I: Data Science Fundamentals
   - Data Science Concepts and Process
   - The R Language
   - Exploratory Data Analysis
   - Cleaning & Manipulating Data
   - Presenting Results
2. Part II: Graphs & Statistical Methods
   - Basic Graphics
   - Advanced Graphics
   - Probability & Statistical Methods
3. Part III: Modeling Methods
   - Model Selection and Evaluation
   - Linear and Logistic Regression
   - Unsupervised Methods
   - Advanced Modeling Methods

Logistic Regression Models

- In our discussion of OLS we noted that the assumption of normality must hold for the dependent variable Y
- *Generalized linear models* extend the linear-model framework to include dependent variables that are non-normal
- In R we use the `glm()` function to fit generalized linear models
- Two popular models in this framework are logistic regression (dependent variable is categorical) and Poisson regression (dependent variable is a count variable)

Logistic Regression Models

- Logistic regression is a good first choice for binary classification problems
- Logistic regression can directly predict values that are restricted to the $(0, 1)$ interval, like probabilities
- Logistic regression assumes that $\log\left(\frac{p}{1-p}\right)$ is a linear function of the $X's$

Logistic Regression Models

- In the linear regression framework we model the expected value of Y given $X_1, X_2, \ldots, X_p$

$$\mu_Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

- In the generalized linear model framework we model a function of the expected value of Y

$$g(\mu_Y) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

where $g(\mu_Y)$ is called a *link function*

Logistic Regression Models

- The format for the `glm()` is as follows:

  `glm(`*`formula`*`, family=`*`family`*`(link=`*`function`*`), data=)`

  where *family* is the probability distribution, and *function* is the link function

Family	Default link function
binomial	(link = "logit")
gaussian	(link = "identity")
gamma	(link = "inverse")
inverse.gaussian	(link = "1/mu^2")
poisson	(link = "log")
quasi	(link = "identity", variance = "constant")
quasibinomial	(link = "logit")
quasipoisson	(link = "log")

Logistic Regression Models

- Logistic regression applies to situations in which the response variable is dichotomous (0 or 1)
- The model assumes that Y follows a binomial distribution
- It takes the form

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

where $\pi = \mu_Y$ is the conditional mean of Y (that is, the probability that $Y = 1$ given a set of X values), $\left(\frac{\pi}{1-\pi}\right)$ is the *odds* that $Y = 1$, and $\log\left(\frac{\pi}{1-\pi}\right)$ is the log odds, or *logit*

Logistic Regression Models

- $\log\left(\frac{\pi}{1-\pi}\right)$ is the link function
- The probability distribution is binomial
- Suppose we have a data frame called `mydata` with dependent variable Y and independent variables $X1, X2,$ and $X3$
- We can fit our logistic regression model as follows:

```
glm(Y~X1+X2+X3, family=binomial(link="logit"), data=mydata)
```

Logistic Regression Models

Let's consider an example..

```
install.packages("AER")
data(Affairs, package = "AER")
summary(Affairs)
table(Affairs$affairs)
```

Looking at the table we see `Affairs$affairs` contains count data – we can convert count data to a binary variable for our logistic regression

```
## create a dichotomous variable from the counts
Affairs$ynaffair[Affairs$affairs > 0] <- 1
Affairs$ynaffair[Affairs$affairs == 0] <- 0
```

Logistic Regression Models

```
Affairs$ynaffair <- factor(Affairs$ynaffair,
                           levels=c(0,1),
                           labels=c("No","Yes"))
table(Affairs$ynaffair)
```

Now our dependent variable is in a form we can use for logistic regression

Logistic Regression Models

Fit a logistic regression model using all the variables

```
fit.full <- glm(ynaffair ~ gender + age
                + yearsmarried + children
                + religiousness + education
                + occupation +rating,
                data=Affairs, family=binomial())
summary(fit.full)
```

What do you observe?

Logistic Regression Models

Fit a reduced logistic regression model using only the significant variables

```
fit.reduced <- glm(ynaffair ~ age
                   + yearsmarried + religiousness
                   + rating, data=Affairs,
                   family=binomial())
summary(fit.reduced)
```

What do you observe?

Use `anova()` to compare nested models

```
anova(fit.reduced, fit.full, test="Chisq")
```

Logistic Regression Models

Let's try to interpret the model

```
coef(fit.reduced)
(Intercept)     age yearsmarried religiousness rating
      1.931 -0.035         0.101          -0.329 -0.461
```

Since the logistic output is log odds the coefficients are difficult to interpret

Let's look at the odds instead...

```
exp(coef(fit.reduced))
(Intercept)   age yearsmarried religiousness rating
      6.895 0.965         1.106          0.720  0.630
```

Logistic Regression Models

Here's what we observe

```
exp(coef(fit.reduced))
(Intercept)   age yearsmarried religiousness rating
      6.895 0.965        1.106         0.720  0.630
```

- For each year increase in `yearsmarried` the odds of having an affair increase by a factor of 1.106
- The odds decrease for a unit increase in each of `age`, `religiousness`, and `rating`
- Since none of the variables takes on a value of 0 the intercept has no interpretation

For the effect of an n unit change we use $(e^{\hat{\beta}_j})^n$

Logistic Regression Models

What is often most useful is the probability of the "outcome" so we transform the output to a probability

$$\Pr(Y = 1|\mathbf{X}) = \frac{e^{\hat{\beta}_0 + \sum \hat{\beta}_i X_i}}{1 + e^{\hat{\beta}_0 + \sum \hat{\beta}_i X_i}}$$

in R we just use the `predict()` function

Logistic Regression Models

Let's use our fitted model to test the effect of `rating` on the probability of having an affair

```
testdata <- data.frame(rating=c(1, 2, 3, 4, 5),
                       age=mean(Affairs$age),
                       yearsmarried=mean(Affairs$yearsmarried),
                       religiousness=mean(Affairs$religiousness))

testdata$prob <- predict(fit.reduced,
                         newdata=testdata,
                         type="response")

testdata
```

Logistic Regression Models

Now let's use our fitted model to test the effect of `age` on the probability of having an affair

```
testdata <- data.frame(rating=mean(Affairs$rating),
                       age=seq(17, 57, 10),
                       yearsmarried=mean(Affairs$yearsmarried),
                       religiousness=mean(Affairs$religiousness))

testdata$prob <- predict(fit.reduced,
newdata=testdata,
type="response")

testdata
```

Logistic Regression Models

- Recall that in the logistic model Y is assumed to have a binomial distribution
- We validate this assumption by testing for *overdispersion*
- Overdispersion occurs when the observed variance of the response variable is larger than what would be expected from a binomial distribution; this can lead to inaccurate tests of significance

Logistic Regression Models

- One way to detect overdispersion is to compare the residual deviance to the residual degrees of freedom in the model

$$\phi = \frac{Residual\ Deviance}{Resdiual\ df}$$

if $\phi >> 1$ there is evidence of overdispersion

```
summary(fit.reduced)$deviance/df.residual(fit.reduced)
```

Logistic Regression Models

- Or fit the same model but use `family=quasibinomial()` instead of `family=binomial()`

```
fit.od <- glm(ynaffair ~ age + yearsmarried
              + religiousness + rating,
              family = quasibinomial(),
              data = Affairs)

pchisq(summary(fit.od)$dispersion * fit.full$df.residual,
               fit.full$df.residual, lower = F)
```

Logistic Regression Models

- While using the results of a logistic regression model to estimate the probability or *likelihood* of the event there are other useful interpretations
- We saw earlier that the (*conditional*) probability is the odds ratio divided by one plus the odds ratio
- Relative risk is the ratio of the conditional probability of one group to another
- Relative risk allows us to compare the risk between two groups
- Using our earlier result we see the probability of a 17 year old having an extra-marital affair is 0.335; for a 57 year old it is 0.109
- $$RR = 0.335/0.109 \approx 3.07$$

  a 17 year old is about 3 times more likely to have an extra-marital affair than a 57 year old

Logistic Regression Models

- Probabilities estimated from logistic regression models are point estimates $\hat{\pi}$
- We can compute a confidence interval for the probability
- First we need the standard error of the probability estimate

$$SE(\hat{\pi}) = \hat{\pi}(1 - \hat{\pi})SE(logit)$$

- A 95% confidence interval is

$$\hat{\pi} \pm 1.96 \times SE(\hat{\pi})$$

Logistic Regression Models

Extensions of Logistic Regression

- *Robust logistic regression* –The `glmRob()` function in the `robust` package can be used to fit a robust logistic regression model helpful when fitting logistic regression models to data containing outliers and influential observations
- *Multinomial logistic regression* –When the response variable has more than two unordered categories (for example, arried/widowed/divorced), you can fit a polytomous logistic regression using the `mlogit()` function in the `mlogit` package
- *Ordinal logistic regression* – When the response variable is a set of ordered categories (for example, credit risk as poor/good/excellent), you can fit an ordinal logistic regression using the `lrm()` function in the `rms` package

Logistic Regression Models – Summary

- Logistic regression is a generalized linear model used for modeling binary, multi-level, and ordered dependent variables
- The transformation for a logistic regression converts the binary outcomes to the log of the odds ratio
- We can transform the predicted values into a probability
- Logistic regression is good first technique for modeling probabilities
- When we set a threshold probability value we can use a logistic regression model as a classifier
- If the Predicted proability exceed the threshold we classify the observation as *true*, otherwise it is *false*
- We use the `glm()` function in R to fit a logistic regression

Poisson Regression Models

- In the example using the `Affairs` data frame we converted the number of affairs to a binary variable
- If we wanted to predict the number of affairs rather than the probability of having an affair we would fit a Poisson regression

```
data(Affairs, package = "AER")
pois.fit <- glm(affairs ~ gender+age+yearsmarried+
                  children+religiousness+education+
                  occupation+rating, data=Affairs,
                  family = poisson())
summary(pois.fit)
```

The Poisson regression has the form

$$\log(Y) = \sum_{i=0}^{p} \beta_i X_i$$

Poisson Regression Models

- We update the model to remove the insignificant variables

```
pois.fit2 <- update(pois.fit, .~.-gender-children
                            -education)
summary(pois.fit2)
```

- Let's use our Poisson model to predict the number of affairs

```
newaffairs <- data.frame(age=c(22,32,42),
                  yearsmarried=mean(Affairs$yearsmarried),
                  religiousness=mean(Affairs$religiousness),
                  occupation=mean(Affairs$occupation),
                  rating=mean(Affairs$rating))
predict(pois.fit2, newdata = newaffairs, type="response")
```

Poisson Regression Models – Summary

- Poisson regression is useful when the dependent variable represents count data
- The transformed dependent variable is $\log(Y)$
- When predicting from a Poisson model remember to use `type=''response''` or exponentiate the raw output

Missing Values

- Sample data sets used in teaching data analysis tend to be complete
- Real world data are often incomplete and we need to deal with it appropriately
- We'll explore how to identify and deal with missing data so we can get the most actionable information form our data

Missing Values

- When dealing with missing data we need to identify the missing data
- Find the causes of the missing data
- Deal with the issue
  - Delete the cases with missing data or
  - Repalce the missing values with reasonable data values

Missing Values

Classifying Missing Data

Missing Completely at Random (MCAR) The data are MCAR if the presence of missing data on a variable is unrelated to any other observed or unobserved variable; there is no systematic reason why the data are missing

Missing at Random (MAR) The data are MAR if the presence of missing data on a variable is related to other observed variables but not to its own unobserved values

Not Missing at Random (NMAR) If the missing data on a variable are neither MCAR nor MAR then the data are NMAR

Missing Values

Table 1: R functions for identifying missing values

x	is.na(x)	is.nan(x)	is.infinite(x)
x <- NA	TRUE	FALSE	FALSE
x <- 0/0	TRUE	TRUE	FALSE
x <- 1/0	FALSE	FALSE	TRUE

Missing Values

Other functions to help us understand the missing data

- Load the `sleep` data from the `VIM` package
  `data(sleep, package="VIM")`
- List the rows without missing data using `complete.cases()`
  `sleep[complete.cases(sleep),]`
- List the ones with missing data
  `sleep[!complete.cases(sleep),]`
- Since `TRUE` and `FALSE` are equivalent to 1 and 0 we can do
  `sum(is.na(sleep$Dream))`
  `mean(is.na(sleep$Dream))`
  `mean(!complete.cases(sleep))`
- `complete.cases()` returns true for `NA` and `NaN` and not `Inf` and `-Inf`

Missing Values

Exploring missing data for patterns

- While the previous functions help us identify missing data there are other ways to understand missing data
- The `mice` package has a function `md.pattern()` that tabulates missing data

```
library(mice)
data(sleep, package="VIM")
md.pattern(sleep)
```

Missing Values

Visually exploring missing data for patterns

- Besides summarizing the missing data in a table we can visually inspect for patterns of missingness

```
library("VIM")
```

- Plot the number of missing values for each variable alone, and for each combination of variables

```
aggr(sleep, prop=FALSE, numbers=TRUE)
```

- Display missing values for each observation – lighter colors are lower values for the variable and darker are higher – red represents missing values

```
matrixplot(sleep)
```

- Plot the relationship between two variables and their distributions given missing vales

```
marginplot(sleep[c("Gest", "Dream")], pch =20,
col = c("darkgray", "red", "blue"))
```

Missing Values

The questions we want to address are...

- What percentage of the data is missing?
- Are the missing data concentrated in a few variables or widely distributed?
- Do the missing values appear to be random?
- Does the anything suggest a possible mechanism that's producing the missing values?

Missing Values

- There are several approaches we can take with missing data
- Rationalize the missing value from other variables
  - Suppose we have a survey and we want to group respondents by birth year
  - We collect date of birth, and age
  - If date of birth is missing we can fill-in the birth year from age and today's date
- Complete case analysis which means delete any observation (row) that is missing one or more value – only analyze complete data
- Use multiple imputation to impute the missing values

Missing Values

- Multiple imputation (MI) provides an approach to missing values based on repeated simulations
- MI is a widely-used method for complex missing-values problems
- Typically a set of 3 to 10 complete datasets is generated from an existing dataset that's missing values
- Monte Carlo methods are used to fill in the missing data in each of the simulated datasets
- Finally standard statistical methods are applied to each of the simulated datasets and the outcomes are combined
- These provide estimated results and confidence intervals that take into account the uncertainty introduced by the missing values

Missing Values

```
fit.na <- lm(Dream ~ Span + Gest, data = sleep)
summary(fit.na)

library(mice)
imp <- mice(sleep, m=5)
fit.mi <- with(imp, lm(Dream ~ Span + Gest))
pooled <- pool(fit.mi)
summary(pooled)
```

- We see from the results that the pooled estimate is close to the original using mising data, but the pooled results provide additional information

Methods for Missing Data – Summary

- When working with real data we will almost always encounter missing values
- To deal with missing values in your data you can delete the entire observation (complete cases) or find a suitable value to use
- Missing data may be MCAR, MAR, NMAR
- If the data are missing at random and a small percentage of your data then deletion is OK
- Impute a value through rationalization
- Multiple imputation provides an approach based on simulations

References

James, G., Hastie, T., Witten, D. and Tibshirani, R. (2013).
An Introduction to Statistical Learning with Applications in R.
Springer, second edition.
6th Printing 2015.

Kabacoff, R. I. (2015).
R in Action.
Manning, Shelter Island, NY, second edition.

Lander, J. P. (2014).
R for Everyone.
Addison-Wesley, Upper Saddle River.

Zumel, N. and Mount, J. (2014).
Practical Data Science with R.
Manning, Shelter Island, NY, second edition.