Binary Logistic Regression

Daniel Rozovsky

Introduction

Attention useRs! If you are an aspiring data scientist and want to learn about how to analyze categorical data to make cool predictions then you've arrived at the right place!

The world is full of complex problems. While this is a bland statement by itself, it is important to understand that humans have been trying to figure out efficient ways to solve complicated problems since the start of humanity. In the exponential development of computer programs, there has been an uprising in the field of data science – with a more recent emphasis on machine learning.

We are going to focus on a specific technique of machine learning called *binary logistic regression*. At the very crux of this regression, we are trying to fit a model that can predict a binary result (whether a person falls into a specific category or not) based on certain parameters. A simple example would be to build a model that could predict whether someone was over 6 feet tall based on gender and weight.

Motivation

The motivation behind this topic is twofold: to not shy away from techniques in machine learning, and to learn how to perform regression on categorical data.

With respect to machine learning, the term has always intimidated me and made me feel subpar in my understanding. Although machine learning basically just gives us tools to make better predictions using machinery and statistical models, I have always found the term daunting. My first goal for this post is therefore to make methods of machine learning more accessible to those that are intimidated by the field.

With respect to regression on categorical data, I'm excited to share the concept of regression with non-continuous data. If this terminology is unclear then don't worry: it'll all be explained clearly below as we delve into a specific example.

Background Information

Before we tackle binary logistic regression, let's first understand what regression is more generally.

When most people think of regression, they think of linear regression. Linear regression is based on the equation of a line, given by: `y = m*x + b`.

Now, we know recognize this equation from algebra, where `m` is the slope, `b` is the intercept, and `x` and `y` are variables. The cool thing about R is that it knows about linear regression and all we need to do is give R a predictor variable (x) and a predicted variable (y), and it will find the slope and the intercept of the best fit line based on the data points provided.

To solidify our understanding of linear regression, let's take a look at a quick example using the built in Iris data-set in R.

Let's get a quick sense of what the Iris data-set contains.

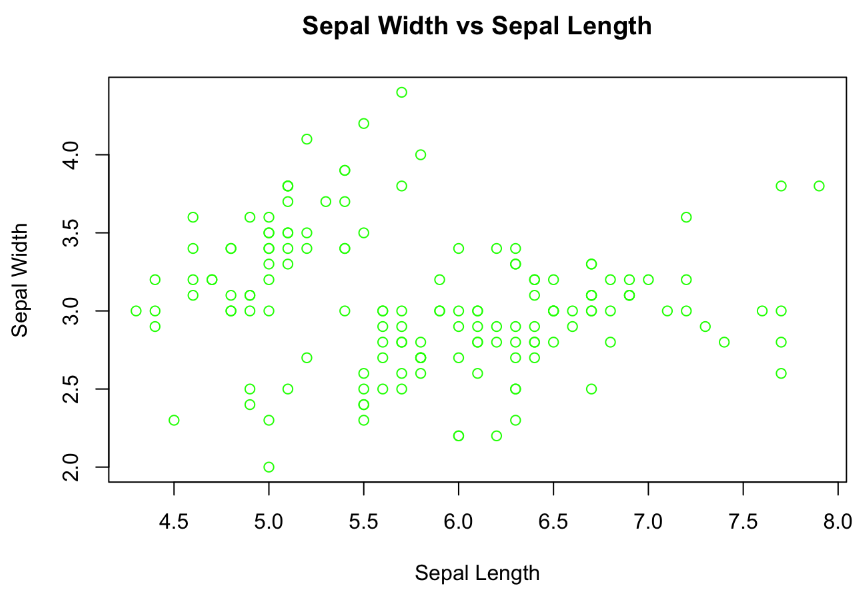
Example of Simple Linear Regression

```r
# Viewing first 5 rows of the data

head(iris, n = 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

We see that Iris has 5 columns, 4 of them numeric, and one of them categorical. Now, let's plot sepal length vs sepal width.

```r
#plotting sepal length vs speal width

plot(iris$Sepal.Length, iris$Sepal.Width, xlab = 'Sepal Length', ylab = 'Sepal Width',
     main = 'Sepal Width vs Sepal Length', col = 'green')
```

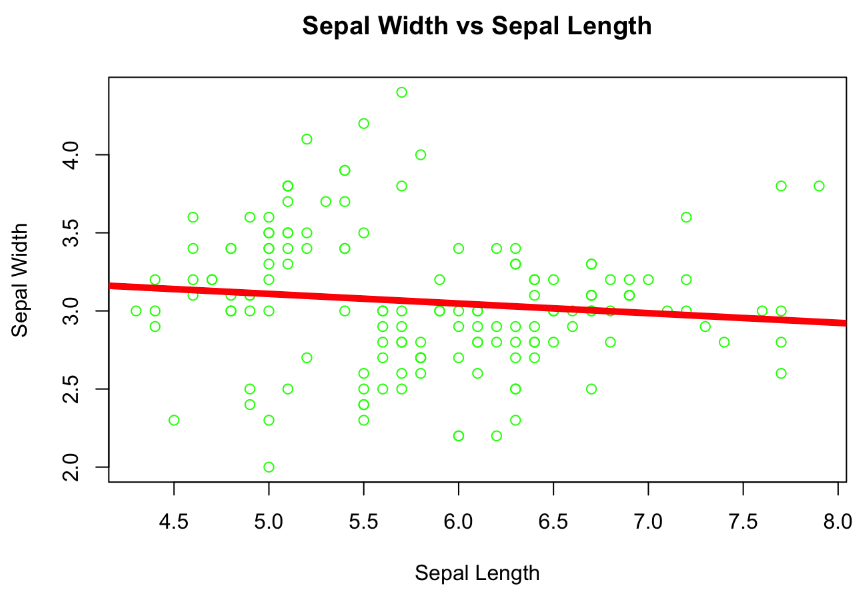


From this plot, we can obtain a best fit line using the lm function in R (lm stands for linear model)

```
# creating a linear regression model which predicts sepal width from sepal length

linear_reg_mod <- lm(Sepal.Width ~ Sepal.Length, data = iris)

# finding slope and intercept

summary(linear_reg_mod)
```

```
##
## Call:
## lm(formula = Sepal.Width ~ Sepal.Length, data = iris)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1095 -0.2454 -0.0167  0.2763  1.3338
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.41895    0.25356   13.48   <2e-16 ***
## Sepal.Length -0.06188    0.04297   -1.44    0.152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4343 on 148 degrees of freedom
## Multiple R-squared:  0.01382,    Adjusted R-squared:  0.007159
## F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

```
#plotting best fit line found with linear regression

plot(iris$Sepal.Length, iris$Sepal.Width, xlab = 'Sepal Length', ylab = 'Sepal Width',
     main = 'Sepal Width vs Sepal Length', col = 'green')

abline(linear_reg_mod, col = 'red', lwd = 5)
```



What are all of these numbers that R outputed you might ask.

Well, the 3 most important values that we will focus on for this example are:

1. The estimate for the intercept
2. The estimate for the slope
3. The p-value

The summary statistics tell us that the estimate intercept is 3.418. That means that based on the model, the predicted value of sepal width is 3.418 when sepal length is 0. Based on the plot that includes the best fit line, you can see that the slope is slightly negative, as confirmed by the summary statistics (slope = -.06). Lastly, you can see the p-value is .152 (disregard the p-value for the intercept as it is irrelevant in this example), and so we can therefore make a claim about statistical significance.

The main takeaway from this example is that the relationship between sepal length and sepal width is negative such that as sepal length increases, sepal width decreases. However, this relationship is not statistically significant and is most likely due to chance rather than some causal relationship linking the two variables.

Binary Logistic Regression

Now that the preliminaries are out of the way, let's get cracking on some binary logistic regression.

The motivation for showing an example of linear regression is to see that it is quite useful for predicting continuous dependent variables (that is numerical data on a continuous scale such as 0-100 or 1-10). We could expand on the previous example and include more variables to try and predict sepal width, but the point is that linear regression will work as long as the dependent variable (the variable we are trying to predict) is continuous.

What happens when we deal with categories and counts for our dependent variable? Yes, you guessed it: binary logistic regression!

We are interested in coming up with a model that can predict whether a condition has been met on some dependent variable. With binary logistic regression, we are not answering questions like "can we predict your salary based on specific factors." Instead, we are asking questions such as, "is your salary greater than $50,000 based on specific factors".

Let's take a look at a concrete, in depth example, and by the end of this, you will be on your way to testing your own hypotheses with binary logistic regression.

The data-set we will be using is a titanic dataset and we are going to try and predict who survived the tragedy based on a set of predictors.

Titanic Example (Binary Logistic Regression)

```r
#reading in and checking structure of the titanic dataset

titanic <- read.csv(file = '../data/train.csv')
str(titanic)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

We see that we have 891 rows and 12 columns in our dataset. For the purposes of this binary logisitic regression, we are only going to use 5 of the columns, 4 of which are going to be our predictors for whether a passenger survived or not. The predictor variables are: Pclass, Age, Sex, and SibSp.

```
#Selecting columns that we will be using

library(dplyr)

#Peeking at subsetted dataset

titanic_new <- select(titanic, Survived, Pclass, Age, Sex, SibSp)
head(titanic_new, n = 5)
```

```
##   Survived Pclass Age    Sex SibSp
## 1        0      3  22   male     1
## 2        1      1  38 female     1
## 3        1      3  26 female     0
## 4        1      1  35 female     1
## 5        0      3  35   male     0
```

We see the 5 columns that we've ended up with are almost all categorical (the only continous variable is age). Here's a quick note on what each column represents:

- Survived: takes 2 levels: 1 means the passenger survived, 0 means they did not
- Pclass (passenger class): takes 3 levels: 1 means first class, 2 means second class, 3 means third class
- Age: the age of the passenger
- Sex: takes 2 levels: male or female
- SibSp (number of family members): gives the number of family members traveling with you

Data Preparation

To deal with the NA's that appear in the age column, we'll replace each NA with the average age. Also, since there were only a few people in this dataset traveling with more than 5 family members (all of which had 8 family members with them), we will change them to have 5 people in their family (for consistency purposes in the model and testing done later).

Finally, after we create the model, we will want to test how accurate it is. For this purpose, we are going to split this dataset into a "training" dataset, where we create the model, and a "testing" dataset, where we see how accurate it is.

```
#creating training data

titanic_train <- slice(titanic_new, 1:750)

#creating testing data

titanic_test <- slice(titanic_new, 750:nrow(titanic_new))

# Replacing NA's in age column with average Age

titanic_train$Age[is.na(titanic_train$Age)] <- mean(titanic_train$Age, na.rm = TRUE)
titanic_test$Age[is.na(titanic_test$Age)] <- mean(titanic_test$Age, na.rm = TRUE)

# Replacing the people traveling with 8 family members to 5

for (i in 1:nrow(titanic_train)) {
  if (titanic_train$SibSp[i] == 8) {
    titanic_train$SibSp[i] <- 5
  }


}

table(titanic_train$SibSp)
```

```
##
##   0   1   2   3   4   5
## 505 180  27  15  14   9
```

What we want to do with this model is to try and figure out whether somebody survived or not based on their class, age, sex, and number of family members on board. So the dependent variable is survival, and the predictor variables are class, age, sex, and number of family members on board.

Data Preparation Continued

We want our categorical variables (survived, class, sex, and number of family members on board) to all be factors, as that is the preferred format for regression models. Sex has already been converted by the function `read.csv`, so let's change the remaining ones.

```
#Changing Survived, Pclass, and SibSp to factors

titanic_train$Survived <- as.factor(titanic_train$Survived)
titanic_train$Pclass <- as.factor(titanic_train$Pclass)
titanic_train$SibSp <- as.factor(titanic_train$SibSp)

#checking the new structure

str(titanic_train)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    750 obs. of  5 variables:
##  $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Age     : num  22 38 26 35 35 ...
##  $ Sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ SibSp   : Factor w/ 6 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...
```

Now that we've successfully subsetted, described, and cleaned the data we will be using, we are ready to build our model!

Quick note: A good reason for going over the linear regression model before going over binary logistic regression is that they are quite functionally similar in R. Linear regression models use the function `lm()` (linear model) and binary logistic regression models use the function `glm()` (generalized linear model).

Building the Model

```
#creating the model

titanic_mod <- glm(Survived ~ Pclass + Age + Sex + SibSp, family = binomial, data = titanic_train)
summary(titanic_mod)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Age + Sex + SibSp, family = binomial,
##     data = titanic_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7193  -0.6460  -0.4408   0.6086   2.3893
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.687088   0.432939   8.516  < 2e-16 ***
## Pclass2      -1.115120   0.283365  -3.935 8.31e-05 ***
## Pclass3      -2.228883   0.262896  -8.478  < 2e-16 ***
## Age          -0.034215   0.008578  -3.989 6.64e-05 ***
## Sexmale      -2.713659   0.212275 -12.784  < 2e-16 ***
## SibSp1        0.053457   0.225589   0.237  0.81268
## SibSp2       -0.121548   0.521141  -0.233  0.81558
## SibSp3       -2.008484   0.689786  -2.912  0.00359 **
## SibSp4       -1.212919   0.786678  -1.542  0.12312
## SibSp5      -15.742965 698.063498  -0.023  0.98201
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1001.77  on 749  degrees of freedom
## Residual deviance:  668.54  on 740  degrees of freedom
## AIC: 688.54
##
## Number of Fisher Scoring iterations: 15
```

Clarifying the Model

Alright, we've built a binary logistic regression model! Before we interpret the results, let's make crystal clear how the code works.

The general code for logistic regression is as follows:
`model <- glm(Y ~ x1 + x2 + ... + xn, family = choice of family, data = dataset)`. Let's unpack what this means.

The function `glm` stands for generalized linear model. This is the function that allows us to run a logistic regression. It takes in a variable Y (the variable we are trying to predict) and a `~` is used to show that we are predicting this dependent variable from some set of predictors. The set of predictors are written out with `+` separating them. While this might be clear, the next argument may not be as intuitive. The argument called `family` takes in some detailed specification of the model. We use `binomial` here because our dependent variable can only take on two values (0 or 1). If our dependent variable could take on more than two values, this would no longer be binary logistic regression, but would turn into multinomial logistic regression. Finally, the `data` argument is set to the dataset we use.

Now, let's figure out how to interpret these results.

Interpreting the Model

As we learned from the linear regression model that we ran earlier, the summary statistics give us an estimate for slopes, and for the intercept.

For categorical data, the intercept is known as the reference group. Notice in the summary statistics, there is no output for SibSp0, Sexfemale, or Pclass1. Therefore, R has automatically created the reference group as women who have no family members on board and that were in first class. The value associated with that group (3.68) is a point of reference for us, and all other outputs below it are comparisons to that value. The estimates disaplyed next to all the other variables listed under intercept are simply comparisons to the reference group. Going down the list, we see:

- Pclass2: If you were in the second class, you are less likely to survive than the reference group by a value of 1.11
- Pclass3: If you were in the third class, you are less likely to survive than the reference group by a value of 2.22
- Age: As your age increases by a unit value (one year), you are less likely to survive than the reference group by a value of .03
- Sexmale: If you were a male, you are less likely to survive than the reference group by a value of 2.71
- SibSp1: If you had one family member on board, you are more likely to survive by a value of .05
- SibSp2: If you had two family members on board, you are less likely to survive than the reference group by a value of .12
- SibSp3: If you had three family members on board, you are less likely to survive than the reference group by a value of 2.08
- SibSp4: If you had four family members on board, you are less likely to survive than the reference group by a value of 1.21
- SibSp5: If you had five family members on board, you are less likely to survive than the reference group by a value of 15.74

So we see that almost all groups from our model are less likely to survive than the reference group (with the exception of having one family member on board). To figure out which of these relationships are statistically significant, we look at the p-value column which is denoted by `Pr(>|z|)`. We note that the strongest relationships in terms of statistical significance are for Pclass2, Pclass3, Age, and Sexmale. We conclude that with all else constant, those passengers that were in second class, third class, elderly, and male had the smallest chance of surviving.

Discussion

The point of creating a model is not just to obtain summary statistics. What we want to do is figure out how accurately our model predicts the actual events that occurred. If we find that our model is only correct 30 percent of the time, we might want to consider a different model that could give us better results.

Testing the Model

Let’s see how accurate our model is. The dataset below is part of the original titanic datset that the model has not been trained on. For reference, we created this testing dataset using the following code: `titanic_test <- slice(titanic_new, 750:nrow(titanic_new))`.

```r
#viewing the first few rows of the dataset

head(titanic_test, n = 5)
```

```
## # A tibble: 5 x 5
##   Survived Pclass   Age    Sex SibSp
##      <int>  <int> <dbl> <fctr> <int>
## 1        0      3    31   male     0
## 2        1      2     4 female     1
## 3        1      3     6   male     0
## 4        0      3    33   male     0
## 5        0      3    23   male     0
```

Preprocessing of Testing Dataset

You’ll note that this is more data from the titanic dataset. It has the same structure as the data we used earlier, but it is on a smaller amount of passengers. This is because it is generally good practice to train you model on the majority of the data (`~80%` of your data), and test it on what remains (`~20%` of your data).

This is our test dataset, and we are going to create a set of predictions for all of the people in the test dataset. Once we have our predictions, we are going to compare it to the actual values for survival and see how accurately our model predicted who survived and who did not. Since our model only used 4 predictor variables (Pclass, Sex, Age, SibSp), we will subset the test data to only include those predictor columns.

Before we make our predictions, we can see that there are a few people in the test dataset that had 8 family members on board with them. We turned people like this into 5’s when making the model, and hence we are going to stay consistent here.

```
#Subsetting test data

titanic_testing <- select(titanic_test, Survived, Pclass, Sex, Age, SibSp)

#Changing SibSp values of 8 into 5 in test dataset

for (i in 1:nrow(titanic_testing)) {
  if (titanic_testing$SibSp[i] == 8) {
    titanic_testing$SibSp[i] <- 5
    }

}

#changing Pclass and SibSp to factors

titanic_testing$Pclass <- as.factor(titanic_testing$Pclass)
titanic_testing$SibSp <- as.factor(titanic_testing$SibSp)

#checking values for SibSp

table(titanic_testing$SibSp)
```

```
##
##   0   1   2   3   4   5
## 104  29   1   1   4   3
```

Now that we have the test data in the format we want, let's go ahead and get our predictions.

Prediction

```
# making prediction

prediction <- predict(titanic_mod, newdata = titanic_testing, type ='response')


#Assigning binary values to predictions (1s for survived and 0s for perished)

prediction_values <- ifelse(prediction > 0.5, 1, 0)
```

Now we have our vector of predictions! Note that the function `predict` takes in the model name, the data you want to predict the model on, and an optional argument called type. When `type` = 'response', it tells R to output a probability based on the model. What that means in the context of our example is that each output is the probability that a passenger in the new dataset (`titanic_testing`) survived based on the model. The cutoff I used was `.5`, and so if the probability that R calculates is greater than this value, the prediction will be that the passenger survived, and will be denoted as a 1. If it is less than .5, then the prediction will be that the passenger perished, and will be denoted as a 0.

Now we compare with the actual data for the people in the test dataset, and see how accurately our model predicted survival!

Accuracy of Model

```
#creating vector of logical values

accuracy_logical <- prediction_values == titanic_testing$Survived

#changing accuracy from logical values (TRUE/FALSE) to numerical values

accuracy_numeric <- accuracy_logical + 0

#calculating accuracy of model

accuracy_percentage <- sum(accuracy_numeric)/length(accuracy_numeric)
accuracy_percentage
```

```
## [1] 0.8098592
```

There you have it! The accuracy of the binary logistic regression model is 80.9%. That means that the model correctly predicted whether passengers survived or didn't for about 4 out of every 5 people in the testing dataset. This is quite a good result!

Conclusion

In summary, this post contained a brief description of regression, a quick example of linear regression, and then an in-depth look at how to create a binary logistic regression model to predict whether passengers survived or did not survive the Titanic. This type of analysis is at the crux of machine learning because we are interested in learning how we can most accurately predict the world around us. As stated in the introduction, humans have been trying to come up with efficient ways to solve complicated problems for as long as we have been around. If you take away one thing from this post, remember that binary logistic regression is a strong method for solving interesting yet complicated data analysis questions involving **categorical dependent variables**.

References

Here are the sources I used to help me create this post:

1. https://www.kaggle.com/c/titanic/data
2. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/
3. https://www.tutorialspoint.com/r/r_logistic_regression.htm
4. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.glm.html
5. http://stat.ethz.ch/R-manual/R-devel/library/stats/html/family.html
6. https://www.youtube.com/watch?v=xl5dZo_BSJk
7. https://datascienceplus.com/perform-logistic-regression-in-r/
8. https://www.youtube.com/watch?v=AVx7Wc1CQ7Y&t=795s