STATS216-Homework2-V0

Darragh Hanley

Thursday, February 05, 2015

#### 1) Logistic regression can give poor results when the two classes can be perfectly separated by a linear decision boundary. Consider just a logistic regression with a single predictor, X,....

##### (a) Show that the likelihood function L() is always strictly less than 1.

First we will examine p(x) under different values of

As goes to , p(x) goes to / (1 + ). is infinity, so p(x) goes to 1, but never reaches 1 as the numerator is always less than the denomiator.

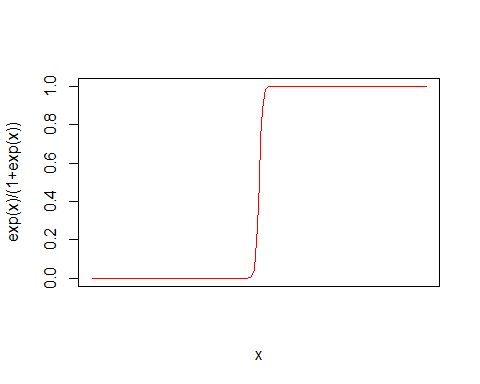
As goes to , p(x) goes to / (1 + ). is 0, so p(x) goes to 0, but never reaches 1 as the numerator never reaches 0, and the denominator goes to but never reaches 1.

Between both of the above we have the case where = 0. Here, P(x) = / (1 - ) = 1 / (1+1) = .5

We can plot these values of P(x), as seen below.

The likelihood function is made of two seperate probabilities. Each of these probabilities follows the model p(X) = / (1 + ). Therefore the likelihood function is always between 0 and 1 but never reaches 0 and 1. So, the likelihood function L() is always strictly less than 1.

x <- -100:100  
p <- exp(x)/(1+exp(x))  
plot(x, p, type="l",xlab="x", col="red", xaxt="n", ylab="exp(x)/(1+exp(x))")



##### (b) Suppose that all of the corresponding to = 0 are negative, and all of the other are positive. In that case, show that we can get L() arbitrarily close to 1. That is, show that for any value a < 1, no matter how close to 1, you can always find values and for which L() > a. Explain why this means that and are undefined.

From the book, we know when the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. This question explains such a case.

We have a single predictor, , and our response, , has two possible classes. Let us call the classes of , 0 and 1.

For all of the = 0. We know that < 0.

For all of the other , eg. where = 1, => 0.

Given the above we know the outcome of Y for any value of X. Once we know whether X is negative or not, we know the class of Y. So, Pr( Y = 0 | X < 0 ) = 1 and Pr( Y = 1 | X => 0 ) = 1.

However the likelihood function used to estimate Y, holds that p(X) = / (1 + ). And as seen in part (a), p(x) in the likelihood function never reaches 1. These are the two terms of the likelihood function.

So the likelihood function will take the p(x) as close to 1 as it can without ever reaching 1. However, no matter how close it can get to 1, for example, value a < 1; it always has room (1-a), to make p(x) closer to 1. Therefore for any value a < 1, no matter how close to 1, you can always find values and for which L() > a.

In this case, the estimates and used in the likelihood function is essentially a moving target. No matter what value we assign to and to get p(x) near to the true probability of 1, we can always change them to make them more accurate. This means and are undefined.

##### (c) Show that and are similarly undefined if there is any value c for which every corresponding to = 0 is less than c and every other is larger than c. In fact, the same is true for multivariate logistic regression. Whenever there is a linear decision boundary that perfectly separates the two classes, the maximum likelihood logistic regression coefficients are undefined (but you don't have to prove this last fact).

Again, we have a single predictor, , and our response, , has two possible classes. Let us call the classes of , 0 and 1.

For all of the = 0. We know that < c.

For all of the other , eg. where = 1, => c.

Given the above we know the outcome of Y for any value of X. Once we know whether X is less than c or not, we know the class of Y. So, Pr( Y = 0 | X < c ) = 1 and Pr( Y = 1 | X => c ) = 1. These are the two terms of the likelihood function.

However the likelihood function used to estimate Y, holds that p(X) = / (1 + ). And as seen in part (a), p(x) in the likelihood function never reaches 1.

Again the likelihood function will take the p(x) as close to 1 as it can without ever reaching 1. However, no matter how close it can get to 1, for example, value a < 1; it always has room (1-a), to make p(x) closer to 1. Therefore for any value a < 1, no matter how close to 1, you can always find values and for which L() > a.

In this case, the estimates and used in the likelihood function is essentially a moving target. No matter what value we assign to and to get p(x) near to the true probability of 1, we can always change them to make them more accurate. This means and are similarly undefined as per question part (b).

##### (d) Come up with your own data set of the form in (c) and fit a logistic regression to it in R. Plot your data, as well as the logistic regression fit ^p(x). You will probably get warning messages that the fit didn't converge, and that you have numerically 0 or 1 fitted probabilities. The first message usually signals that you have fit a logistic regression to perfectly separable classes.

First we set the seed and generate 1000 random values of mean 2 and standard deviation 5. this is the predictor.

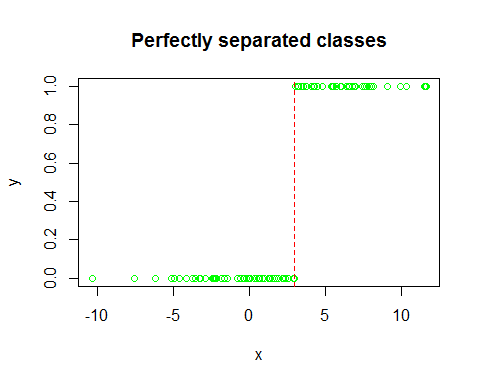
set.seed(124)  
norm <- rnorm(100, 2, 5)  
x <- norm[1:1000]

Now we create the response of the same length. We use c=3 as the seperator of classes. For all x<3 the response is 0. And for all other the response is 1.

y <- rep(NA, 1000)  
y[x>=3] <- 1  
y[x<3] <- 0

Create the plot and add a line to indicate where the seperating boundary is.

plot(x,y, col="green", main = "Perfectly separated classes")  
abline(v=3,col="red",lty=2)



Now we fit the logistic regression and check the warning message.

log.mod <- glm(y ~ x, family=binomial)

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

#### 2) We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

##### (a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

The probability that the first bootstrap observation is not the jth observation is (n-1)/n.

This can be proved as follows the case because the,

1. Probability that the *first bootstrap obervation is observation j* = 1/n
2. Probability that the *first bootstrap obervation is observation j* or *first bootstrap obervation is not observation j* = 1
3. Therefore, probability that the *first bootstrap obervation is not observation j* = 1-1/n = (n-1)/n

##### (b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

The probability that the second observation is not the jth observation from the original sample is (n-1)/n. The same as the probability for the first observation.

##### (c) Argue that the probability that the jth observation is not in the bootstrap sample is (1-(1/n))^n.

In a bootstrap sample, of n observations, we choose out a single observation, n times.

As seen in part (a), above, for each single observation the probability we will not choose the jth observation is : 1-(1/n).

If two events, A and B are independent then the joint probability is P(A and B) = P(A)P(B).

In the bootstrap case, each event is independent. Therefore the we get the product of all the events.

We have n events, each of probability (1-(1/n))^n. Therefore we get the probability of that event n times. This is (1-(1/n))^n.

Lets test this. If there are two observations(n=2), and j=2. The probability j=2 is not drawn on the first sample is .5, the same on the second sample. The probability it is not chosen in either is the .25. Plugging this into the formula :

n = 2  
 (1-(1/n))^n == .25

## [1] TRUE

##### (d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

The formula to use for the jth observation **not in the bootstrap sample** is (1-(1/n))^n, as seen in previous questions.

The formula to use for the jth observation **in the bootstrap** sample is 1-(1-(1/n))^n. Since both events are mutually exclusive and the probability of getting one of the two events is 1.

n = 5  
 print(paste( "For n=5, the probability that the jth observation is in the bootstrap sample ", round(1-(1-(1/n))^n, digits=3), sep=""))

## [1] "For n=5, the probability that the jth observation is in the bootstrap sample 0.672"

##### (e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

n = 100  
 print(paste( "For n=100, the probability that the jth observation is in the bootstrap sample", round(1-(1-(1/n))^n, digits=3)))

## [1] "For n=100, the probability that the jth observation is in the bootstrap sample 0.634"

##### (f) When n = 10,000, what is the probability that the jth observation is in the bootstrap sample?

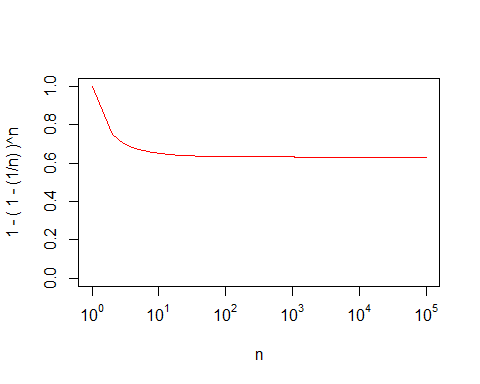
n = 10000  
 print(paste( "For n=10,000, the probability that the jth observation is in the bootstrap sample", round(1-(1-(1/n))^n, digits=3)))

## [1] "For n=10,000, the probability that the jth observation is in the bootstrap sample 0.632"

##### (g) Create a plot that displays, for each integer value of n from 1 to 100000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

It can be seen below that as the population grows larger, the probability that the jth observation will not be picked converges to one value, circa 0.632.

x <- 1:100000  
y <- 1-(1-(1/x))^x  
plot(x, y, log="x", type="l", xlab="n", col="red", xaxt="n", ylab="1 - ( 1 - (1/n) )^n", ylim=c(0, 1))  
ticks <- seq(0, 5, by=1)  
labels <- sapply(ticks, function(i) as.expression(bquote(10^ .(i))))  
axis(1, at=c(1, 10, 100, 1000, 10000,100000), labels=labels)



##### (h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

##### > store = rep (NA , 10000)

##### > for (i in 1:10000) { store [i]= sum ( sample (1:100 , rep= TRUE ) ==4) > 0 }

##### > mean ( store )

##### Comment on the results obtained.

The below code put the theory of part (g) to experiment, by actually creating 10000 different boot straps from the same population, of sample size = 100. It then checks from these 10000 samples, what proportion of the time, a particular value was taken into the sample. The answer, .6408, is very close to the probability, seen in part (g) of the question, which large values of n converge to.

set.seed(1)  
store = rep (NA , 10000)  
for (i in 1:10000) {   
 store [i]= sum ( sample (1:100 , rep= TRUE ) ==4) > 0   
 }  
mean ( store )

## [1] 0.6408

#### 3) Suppose we estimate the regression coefficients in a linear regression model by minimizing... ridge regression formula... for a particular value of lambda. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

##### (a) As we increase from 0, the training RSS will: (i. Increase initially, and then eventually start decreasing in an inverted U shape. ii. Decrease initially, and then eventually start increasing in a U shape. iii. Steadily increase. iv. Steadily decrease. v. Remain constant.)

1. Steadily increase.

with 0 , the linear regression will minimise the MSE of the training data. As we reduce the coefficients by increasing , the training MSE will increase steadily.

##### (b) Repeat (a) for test RSS.

1. Decrease initially, and then eventually start increasing in a U shape.

Initially, as we make larger from 0, the bias is pretty much unchanged (slightly increasing), but the variance drops. So a ridge regression, by shrinking the coefficient toward 0, controls the variance. It doesn't allow the coefficient to be too big, and it gets rewarded because the mean square error (sum of bias & variance) goes down. However at one point the bias starts increasing more quickly and we reach a point where the test MSE is minimised. At this point Bias is rising faster than variance is falling. This causes the test RSS to rise again.

##### (c) Repeat (a) for variance.

1. Steadily decrease.

This is rooted in the bias-variance trade-off. An increase in , from 0, causes a decrease in the flexibility of the ridge regression fit, leading to decreased variance and increased bias.

##### (d) Repeat (a) for (squared) bias.

1. Steadily increase.

This is rooted in the bias-variance trade-off. An increase in causes a decrease in the flexibility of the ridge regression fit, leading to decreased variance and increased bias.

##### (e) Repeat (a) for the irreducible error.

1. Remain constant.

Irreducible error is the noise term in the true relationship that cannot fundamentally be reduced by any model. The reducible error can be reduced by model improvements. Given this the irreducible error does not change with any model change.

#### (4) Teamed up with Sevvandi Kandanaarachchi, Tony Wu and Andrew Beckerman for this challenge.

##### (a) Fit the logistic regression model above to the data and examine the rankings. What happened to make both the team saint-mary-saint-mary and the team st.-thomas-(tx)-celts look so good? Can you explain it in terms of your answers to the first question in this problem set?

First we shall load the NACC data set and pull out all teams to a variable sorted by name.

games <- read.csv("http://www.stanford.edu/~wfithian/games.csv",as.is=TRUE)  
teams <- read.csv("http://www.stanford.edu/~wfithian/teams.csv",as.is=TRUE)  
all.teams <- sort(unique(c(teams$team,games$home,games$away)))

Next we assign a variable for each game to indicate whether the home team won or not. (My learning here is Basketball does not allow tied games like soccer). Initialise a data frame of the games(rows) and teams (columns), indicating for each game which teams were home or away.

### Assign score win or loss to vector z. This will be used a response in the model.  
z <- with(games, ifelse(homeScore>awayScore,1,0))  
X0 <- as.data.frame(matrix(0,nrow(games),length(all.teams)))  
names(X0) <- all.teams  
# Assign to the data frame for each game which team was home or away  
for(tm in all.teams) {  
 X0[[tm]] <- 1\*(games$home==tm) - 1\*(games$away==tm)  
}

Remove stanford's column to make it the baseline team against which other teams will be compared. Create the index of regular season games and assign a homeadvantage coefficient which is common to all games.

X <- X0[,names(X0) != "stanford-cardinal"]  
reg.season.games <- which(games$gameType=="REG")  
homeAdv <- 1 - games$neutralLocation

Perform out the logistic regression on the regular season games with the home advantage coefficient. Extract from this the top 25 teams. From this we can see clearly that the team saint-mary-saint-mary and the team st.-thomas-(tx)-celts look very good in comparison to other teams. Now we will eplore these teams a

logrega.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=reg.season.games)  
margin.top25 <- order(coef(summary(logrega.mod))[,1],decreasing=TRUE)[1:25]  
coef(summary(logrega.mod))[margin.top25,1]

## `saint-mary-saint-mary` `st.-thomas-(tx)-celts`   
## 14.126965 13.270185   
## `gonzaga-bulldogs` `louisville-cardinals`   
## 2.770782 2.412485   
## `kansas-jayhawks` `indiana-hoosiers`   
## 2.265964 2.255148   
## `new-mexico-lobos` `ohio-state-buckeyes`   
## 2.239499 2.137208   
## `duke-blue-devils` `georgetown-hoyas`   
## 2.106862 2.031684   
## `michigan-state-spartans` `michigan-wolverines`   
## 1.976361 1.960319   
## `miami-(fl)-hurricanes` `kansas-state-wildcats`   
## 1.761367 1.749419   
## `syracuse-orange` `memphis-tigers`   
## 1.604678 1.532102   
## `saint-louis-billikens` `marquette-golden-eagles`   
## 1.521991 1.508487   
## `butler-bulldogs` `wisconsin-badgers`   
## 1.459961 1.450762   
## `florida-gators` `oklahoma-state-cowboys`   
## 1.316371 1.303918   
## `unlv-rebels` `arizona-wildcats`   
## 1.293854 1.292302   
## `pittsburgh-panthers`   
## 1.266337

Now we will explore these two teams. We draw up a table of the two top teams, how many games they were in as home and away. It can be seen that both team only played 1 away game.

table(X[,c("saint-mary-saint-mary","st.-thomas-(tx)-celts")])

## st.-thomas-(tx)-celts  
## saint-mary-saint-mary -1 0  
## -1 0 1  
## 0 1 5539

In addition it can be seen that both teams one their game which they played.

bool.condition.1=games$home=="saint-mary-saint-mary"|games$away=="saint-mary-saint-mary"  
games[bool.condition.1,]

## date home away homeScore  
## 5300 2012-11-11 nebraska-omaha-mavericks saint-mary-saint-mary 86  
## awayScore neutralLocation gameType  
## 5300 96 0 REG

bool.condition.2=games$home=="st.-thomas-(tx)-celts"|games$away=="st.-thomas-(tx)-celts"  
games[bool.condition.2,]

## date home away homeScore awayScore  
## 2899 2012-11-10 rice-owls st.-thomas-(tx)-celts 59 72  
## neutralLocation gameType  
## 2899 0 REG

We saw in question 1b, when the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. And I believe here we have such a case where a team only played an away game and had no losses.

I believe the reason we are seeing such high for the two teams, looking at the formula for the logistic model, because the data suggests that the home team will almost certainly lose against them as the away team (p almost 0), which requires to be extremely high in that scenario, and that they as the home team will almost certainly win against the opponent (p almost 1), which requires to be extremely high in that scenario.

#### (b) Get rid of teams that played less than five games and refit the model. Make a rank table like the ones we made in class, where you compare the logistic regression rankings to the linear regression rankings, the AP Rankings, and the USA Today rankings. Which model seems to correspond better to the voters' decisions, the linear regression or logistic regression?

In order to excude teams other than Stanford (our baseline for linear modelling), we have two options. Exclude the teams (columns), but leave their games (like was done for Stanford). Or exclude the games they played in. I have chosen the latter for two reasons. 1) Standofrd was our baseline by removing the team columns; this baseline will no longer be valid if many other team columns are excluded. 2) If low game teams are excluded, hoever their games are left in - the opposition's result (win or lose) will still be available as a predictor; however the strength of the low playing team they played will be lost. For logistic regression, using either approach, there are minor differences seen in the ranking result, however for the above reasons we will go with excluding the games of these low playing teams (ie. excluding rows).

We start by adding to table "games" a column of the minimum number of games played by either team. We will concentrate on regular season games only.

# Collect all teams to one variable  
teamplays <- table(c(games$home, games$away))  
# for each regular season game, add a new variable to games representing the lowest number of games played by either team  
games$mingames[reg.season.games] <- vapply(reg.season.games, function(i) {  
 min(  
 teamplays[names(teamplays)==games$away[i]],  
 teamplays[names(teamplays)==games$home[i]]  
 )   
}, 1)   
### Create a subset index on both regular season games and games played by non-low play frequency teams  
subset.vector <- which(games$gameType=="REG" & games$mingames >= 5)

Now we perform bothe the logistic regression and linear model with with the same index. For each we pull out the coefficients which they assigned to each team. These are then added to a table of rankings, where we compare each model to the AP Rankings, and the USA Today rankings.

### Perform the logistic regression excluding teams with 5 games or less and non regular season games  
logregb.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=subset.vector)  
logregb.coef <- coef(logregb.mod)[paste("`",teams$team,"`",sep="")]  
names(logregb.coef) <- teams$team  
  
### Calculate the Linear Model from class, change the variable names  
y <- with(games, homeScore-awayScore)  
homeAdvlm <- 1 - games$neutralLocation  
lmb.mod <- lm(y ~ 0 + homeAdvlm + ., data=X, subset=subset.vector)  
lmb.coef <- coef(lmb.mod)[paste("`",teams$team,"`",sep="")]  
names(lmb.coef) <- teams$team  
  
### Create the ranking table. To fit the table rows on one line, we exclude the coefficient scoring.  
rank.table <- cbind(  
# "lm Score" = lmb.coef,  
 "lm Rank" = rank(-lmb.coef,ties="min"),  
# "logreg Score" = logregb.coef,  
 "logreg Rank" = rank(-logregb.coef,ties="min"),  
 "AP Rank" = teams$apRank,  
 "USAT Rank" = teams$usaTodayRank)  
rank.table[order(logregb.coef,decreasing=TRUE)[1:25],]

## lm Rank logreg Rank AP Rank USAT Rank  
## gonzaga-bulldogs 4 1 1 1  
## louisville-cardinals 3 2 2 2  
## kansas-jayhawks 6 3 3 3  
## indiana-hoosiers 1 4 4 5  
## new-mexico-lobos 23 5 11 10  
## ohio-state-buckeyes 7 6 7 6  
## duke-blue-devils 5 7 6 7  
## georgetown-hoyas 18 8 8 8  
## michigan-state-spartans 12 9 9 9  
## michigan-wolverines 8 10 10 11  
## miami-(fl)-hurricanes 14 11 5 4  
## kansas-state-wildcats 36 12 12 14  
## syracuse-orange 10 13 16 18  
## memphis-tigers 43 14 19 15  
## saint-louis-billikens 26 15 13 13  
## marquette-golden-eagles 30 16 15 16  
## butler-bulldogs 50 17 NA NA  
## wisconsin-badgers 11 18 18 17  
## florida-gators 2 19 14 12  
## oklahoma-state-cowboys 20 20 17 19  
## unlv-rebels 32 21 NA NA  
## arizona-wildcats 16 22 21 20  
## pittsburgh-panthers 9 23 20 22  
## notre-dame-fighting-irish 34 24 23 NA  
## colorado-state-rams 22 25 NA NA