Data Valorization

Logistic Regression with Gradient Ascent

Date: 14 - April - 2017

ELANGO Veeresh
GALDO SEARA Luis
GONZÁLEZ HUESCA Juan Manuel
SANTOS BUITRAGO Nazly

Contents

Introduction

The objective of the project is to generate a logistic regression with gradient ascent from two random variables following a normal distribution (1000 samples each), converted into binary responses, to describe their behavior with a suitable regression curve. It was an interesting and challenging task which help us to consolidate the mathematical background of a regression, in this case a logistic one.

Along the project we faced many design decision challenges, one very important was the choice of the threshold for creating the binary responses out of the two random variables. This decision was key in the process because by choosing the right one, we could build a more accurate logistic regression model with independent and identically distributed random variables.

The process gave us a unique chance to get a deeper understanding of a wide range of concepts, such as logistic regression, probability distributions, gradient calculation, hypothesis testing, maximum likelihood and their interconnection; it was a long and difficult process as our background is not mathematical oriented but it was worth it. To overcome some difficulties and solve the doubts that were raised while interconnecting the different topics needed for the project, most of the time was invested in research and understanding of the principles and applications of the concepts.

The report will start with the sample generation, then we will move to the logistic regression section where we build the model which is then explained deeper in terms of the chosen parameters. After, the model is evaluated, and to finalize a Neyman-Pearson test is designed and applied to the binary responses.

Implementation

Sample generation

The first step to generate a logistic regression model is the creation of two sample sets as indicated in the exercise. The first sample set “X0” will have 1000 samples following a normal distribution with mean of 0.5 (m_0) and standard deviation of 1.2, while the second set “X1” will have a different mean of 1.1 (m_1) and same standard deviation. We used the "rnorm" distribution function in r which generates random numbers from normal distribution with the following syntax: rnorm(n, mean, sd), where the first value “n” are the samples, the second is the “mean” and the third “sd” the standard deviation.

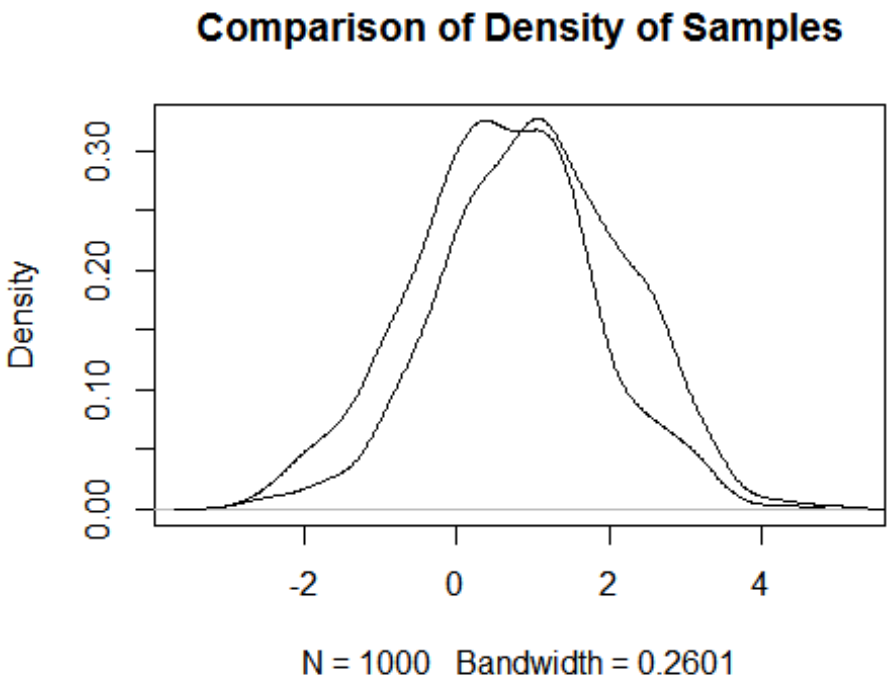
```
#Creation of random variables following a normal distribution.
set.seed(12345)
X0 <- rnorm(1000,0.5,1.2)
X1 <- rnorm(1000,1.1,1.2)
summary(X0)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.8340 -0.2152  0.5555  0.5554  1.3260  4.4970
```

From summary of “X0” we can notice the mean is around the theoretical 0.5, the minimum value is -2.83 and the maximum 4.49.

```
summary(X1)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.9020  0.2372  1.0740  1.0640  1.9100  5.0470
```

As the mean value of the "X1" distribution is greater than "X0"'s, we observed how the distribution is shifted towards a greater x value, reaching a max of 5.04, something that can be seen in plot of the probability density functions of both random variables.

```
plot(density(X0),main="Comparison of Density of Samples")
lines(density(X1))
```



Binary Responses

```
#Initial Beta values
B0 <- 0.3
B1 <- 1.7
```

To generate some binary responses from X0 and X1 samples, we first calculated the Logit transformation of the samples as below.

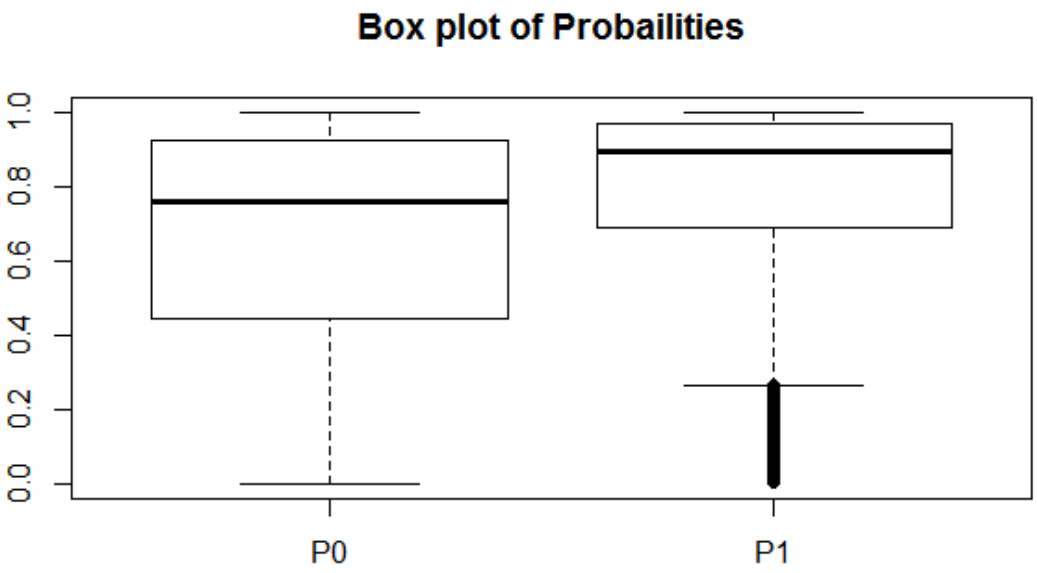
```
#Logit(P) Slide 15
logitP0 <- B0 + B1*X0
logitP1 <- B0 + B1*X1
```

Then we calculated the probability of the Logit values as shown below.

```
#Slide 14, P=f(logit(P))=1(1+e^-logit(P))
P0 = 1/(1+exp(-logitP0))
P1= 1/(1+exp(-logitP1))
summary(P0)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0108  0.4836  0.7763  0.6815  0.9279  0.9996
summary(P1)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.009634 0.668900 0.893400 0.784300 0.972000 0.999900
```

We could notice from the summary of the P0 and P1, that the difference between minimum and 1st quartile value is so high. This shows that most of the probabilities are having high values. To confirm this, we plot the Box Plot of the probabilities.

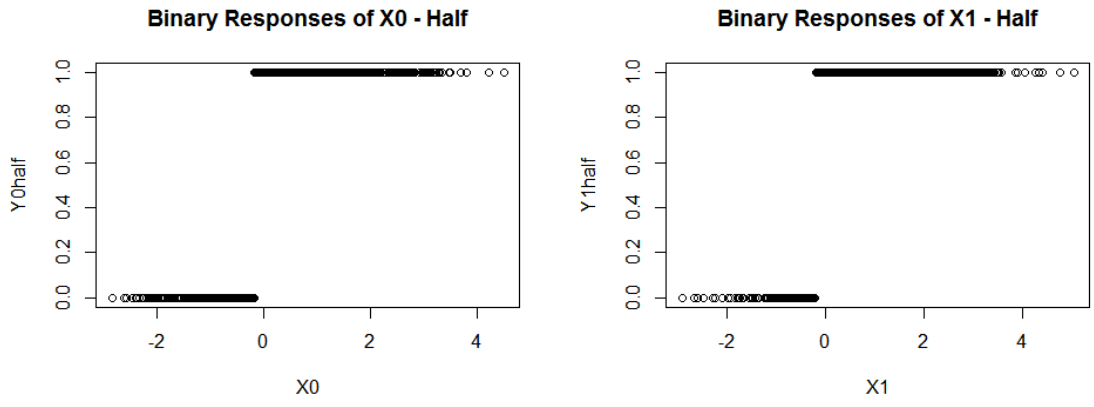
```
boxplot(P0,P1,main="Box plot of Probailities", names=c("P0","P1"))
```



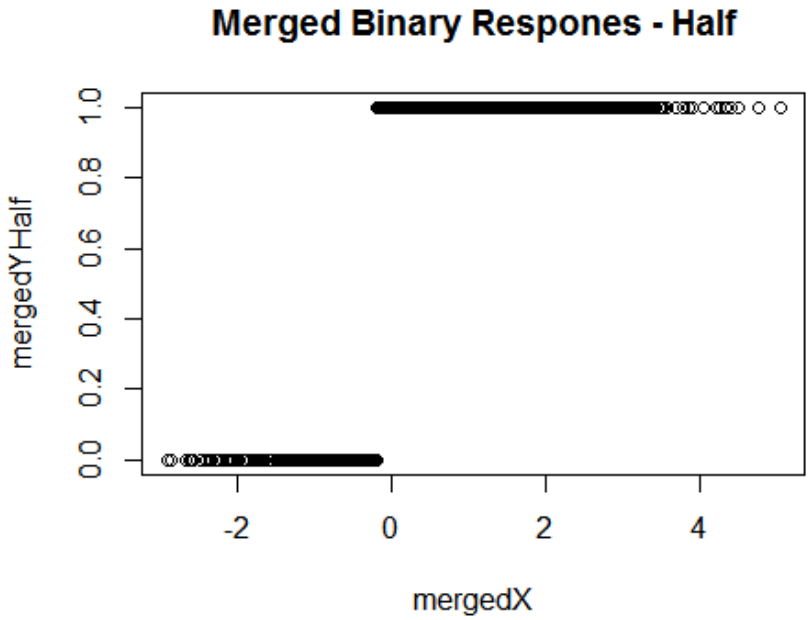
After this observation, we came up with three theoretical different approaches to decide the threshold for creating binary responses.

Threshold as 0.5

```
#Assigning binary values to Y0 and Y1.
Y0half <- ifelse(P0 > 0.5, 1, 0)
Y1half <- ifelse(P1 > 0.5, 1, 0)
plot(X0,Y0half, main="Binary Responses of X0 - Half")
plot(X1,Y1half, main="Binary Responses of X1 - Half")
```



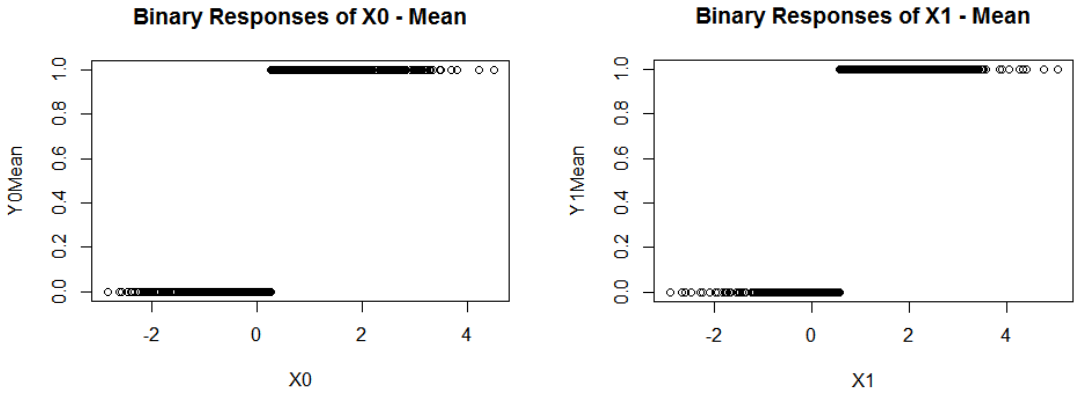

```
mergedYHalf <- c(Y0half,Y1half)
plot(mergedX,mergedYHalf,main="Merged Binary Respones - Half")
```



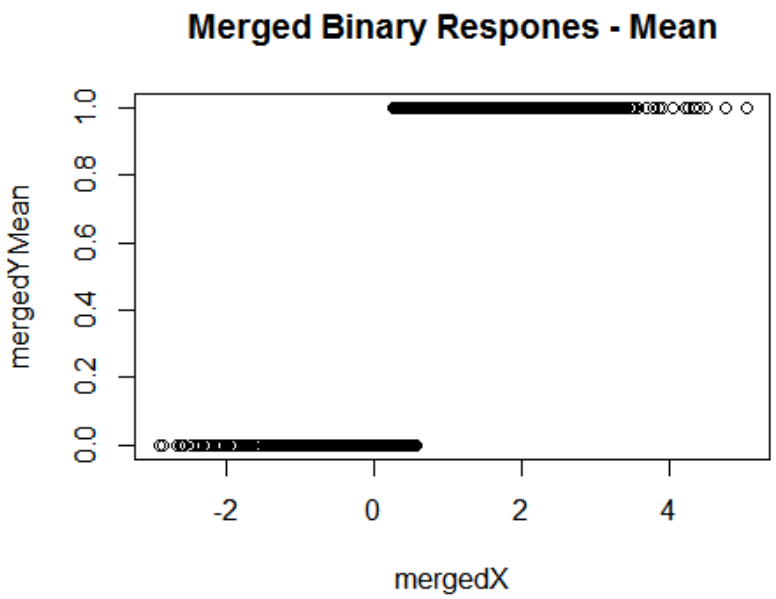
Even though considering 0.5 as threshold is meaningful, it is producing an ideal split in the responses, which is not practically possible.

Threshold as Mean

```
#Assigning binary values to Y0 and Y1.
Y0Mean <- ifelse(P0 > mean(P0), 1, 0)
Y1Mean <- ifelse(P1 > mean(P1), 1, 0)
plot(X0,Y0Mean, main="Binary Responses of X0 - Mean")
plot(X1,Y1Mean, main="Binary Responses of X1 - Mean")
```




```
mergedYMean <- c(Y0Mean,Y1Mean)
plot(mergedX,mergedYMean,main="Merged Binary Respones - Mean")
```

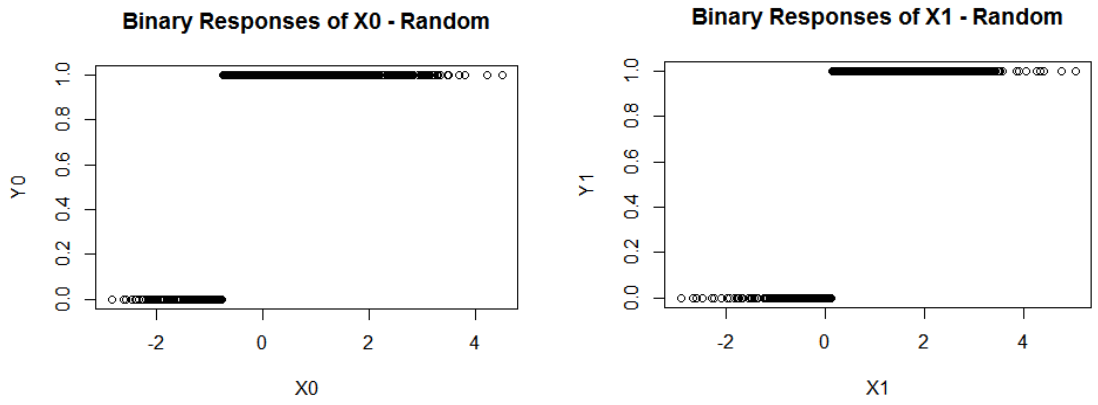


By choosing to mean as threshold, it produced binary responses with overlap which is practical.

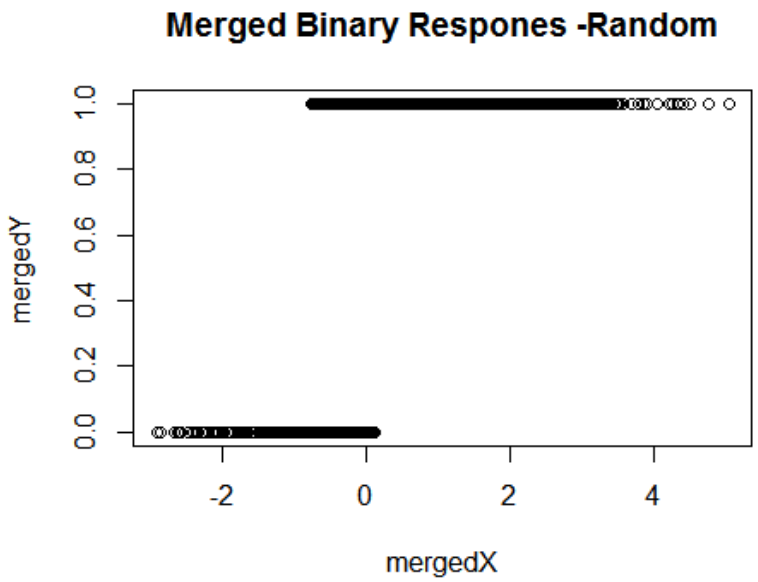
If we follow this approach it would be like saying the following:
Real Madrid has 80% chances on average of winning a game in the league. If you check game by game, when there is one that is below 80% (75% for example) you will classify it automatically as a lost, which does not make sense.

Threshold as Random values

```
#Assigning binary values to Y0 and Y1.
Y0 <- ifelse(runif(1,0,1) <= P0, 1, 0)
Y1 <- ifelse(runif(1,0,1) <= P1, 1, 0)
plot(X0,Y0, main="Binary Responses of X0 - Random")
plot(X1,Y1, main="Binary Responses of X1 - Random")
```




```
#Merging the data for following points.
mergedX <- c(X0,X1)
mergedP <- c(P0,P1)
mergedY <- c(Y0,Y1)
plot(mergedX,mergedY,main="Merged Binary Respones -Random")
```



This produces overlap and overcomes the limitation of choosing mean as threshold. By choosing a random threshold for every sample, we are making every sample independently identical from other samples.

Logistic Regression

Implementation

As in the exercise, the code will be divided in the three following parts to make the explanation more understandable:

- Gradient
- Update of estimates β_0 and β_1
- Maximum Likelihood

These steps will be part of a loop that will end when the cost (maximum likelihood) stops increasing.

```
##Logisitc Regression Model
listCost <- c() #store the costs to verify the convergence of the
function
listNormGradient <- c()
gamma <-0.001 #Depends on random sample, number of random samples and
the computation power
iteration <- 0 #variable to store number of iterations
oldCost <- 0 #variable to store the previous cost
currentCost <- 0 #variable to store the current cost
```

Gradient

Following the gradient calculation formula (below), the derivative of the equation was performed with respect to β_0 and β_1. The gradient indicates in which direction to move.

$$J(\beta) = \sum_i y_i \beta X_i - \sum_i \ln(1 + \exp(\beta X_i))$$

Derivative respect to β_0:

$$\frac{\delta J(\beta)}{\delta \beta_0} = \sum_i y_i - \sum_i \frac{\exp(\beta_0 + \beta_1 X_i)}{1 + \exp(\beta_0 + \beta_1 X_i)} = \sum_i y_i - \hat{p}_i$$

Derivative respect to β_1:

$$\frac{\delta J(\beta)}{\delta \beta_1} = \sum_i y_i X_i - \sum_i \frac{\exp(\beta_0 + \beta_1 X_i)}{1 + \exp(\beta_0 + \beta_1 X_i)} X_i = \sum_i X_i (y_i - \hat{p}_i)$$

These values, together with gamma, will indicate the step taken to update the beta values.

Update of estimates β_0 and β_1

Once the gradient is performed and gamma is established (0.001 in our case), the new betas can be calculated with the following formulas.

$$\hat{\beta}_{0,t+1} = \hat{\beta}_{0,t} + \gamma \sum_i (y_i - \hat{p}_{i,t})$$

$$\hat{\beta}_{1,t+1} = \hat{\beta}_{1,t} + \gamma \sum_i X_i (y_i - \hat{p}_{i,t})$$

Maximum Likelihood (cost)

Once the estimates are calculated, it is time to perform the cost function. The value of it will increase until the best result is achieved. At that point, the obtained beta values will be the values used to define the logistic regression function. The formula to compute the cost is the following:

$$J(\beta) = \sum_i y_i \beta X_i - \sum_i \ln(1 + \exp(\beta X_i))$$

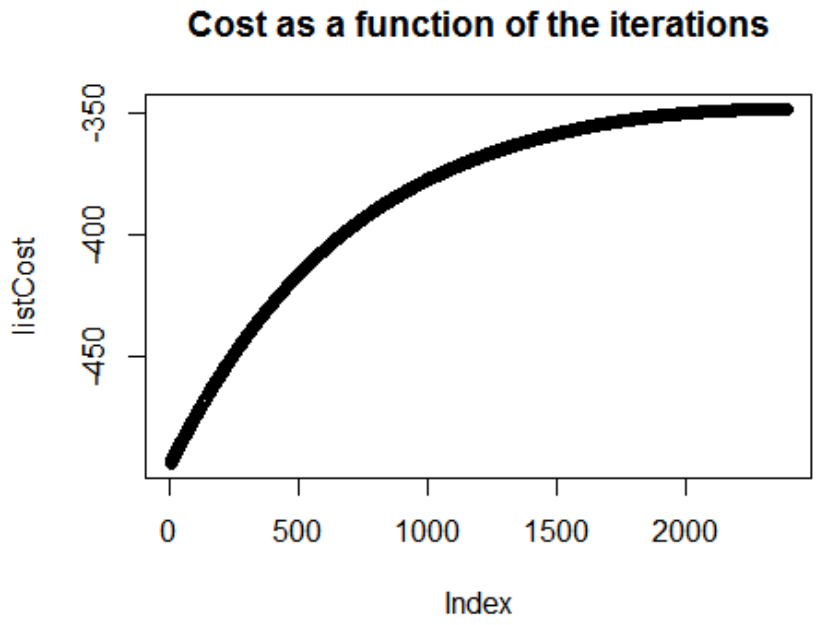
```
#LOOP - until the current cost converges (current cost is lower or
equal than the old cost)
while(abs(oldCost)>=abs(currentCost)){
  Phat <- 1/(1+exp(-B0 -B1*mergedX))#slide 29
  #Gradient is calcualte derivating the cost function
  gradientB0 <- sum(mergedY-Phat)
  gradientB1 <- sum(mergedX*(mergedY-Phat))
  #----------------------------
  #As mentioned in the assignment, gamma is tuned based on gradient
divided by its norm
  normGradient <-sqrt(gradientB0^2 +gradientB1^2)

  normGradientB0 <- gradientB0/normGradient
  normGradientB1 <- gradientB1/normGradient
  #-----------------------------
  #Estimation of B0 and B1 according to gamma factor
  B0 <- B0 + gamma*normGradientB0
  B1 <- B1 + gamma*normGradientB1
  #---------------------------
  #computation of Cost - slide 28
  cost <- sum(mergedY*(B0 +B1*mergedX))-sum(log(1+exp(B0
+B1*mergedX)))
  #-------------------------
  #code for detecting the convergence
```

```
  listCost <- c(listCost,cost)
  listNormGradient <- c( listNormGradient,normGradient)
  if(iteration==0){
    currentCost <- cost
    oldCost <- cost
  }
  else{
    oldCost <- currentCost
    currentCost <- cost
  }
  iteration <- iteration+1
}
print(B0)
## [1] 1.406216
print(B1)
## [1] 3.734048
print(iteration)
## [1] 2395
plot(listCost,main="Cost as a function of the iterations")
```

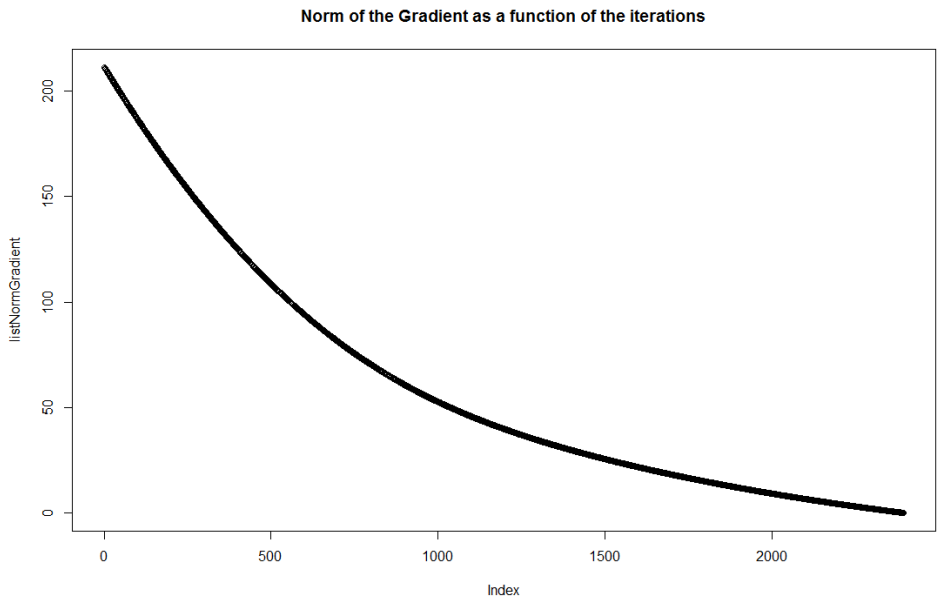
At the end of the loop we obtain the beta values (β_0 and β_1) that maximizes the cost with 1000 samples and with a gamma of 0.001, having 2395 iterations before we reach the maximum likelihood.

As shows in the graph below the cost is converging as per the number of iterations, but it's worth to mention it depends also on the number of samples and the gamma value chosen.

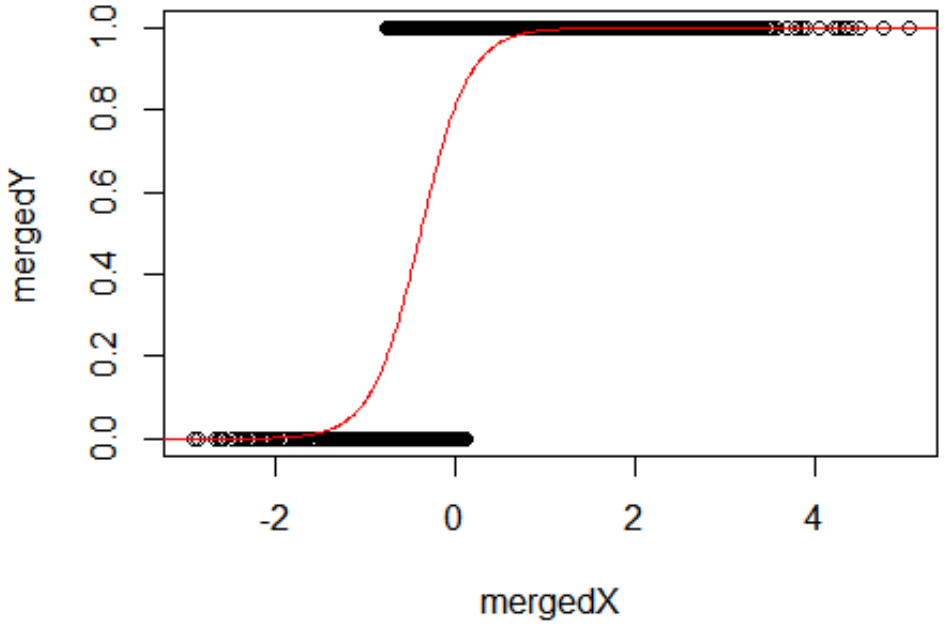


Plotting the norm of the gradient as a function of the iterations helps analyze how in each step the norm decreases because β_0 and β_1 are getting closer to their optimal values.

```
plot(listNormGradient,main="Norm of the Gradient as a function of the
iterations")
```




```r
#Plot of our Logistic Regression Model
t=seq(-10,10,0.01)
y=1/(1+exp(-B0-B1*t))
plot(mergedX,mergedY)
lines(t,y,type="l", xlab="x", ylab="y",col="red")
```



Once we have the values β_0 and β_1 for our logistic regression model, we can create a classifier, where we can give an input on the x-axis and get a binary Y response following the model. This model was tested with 2000 samples with an accuracy of 90.75%.

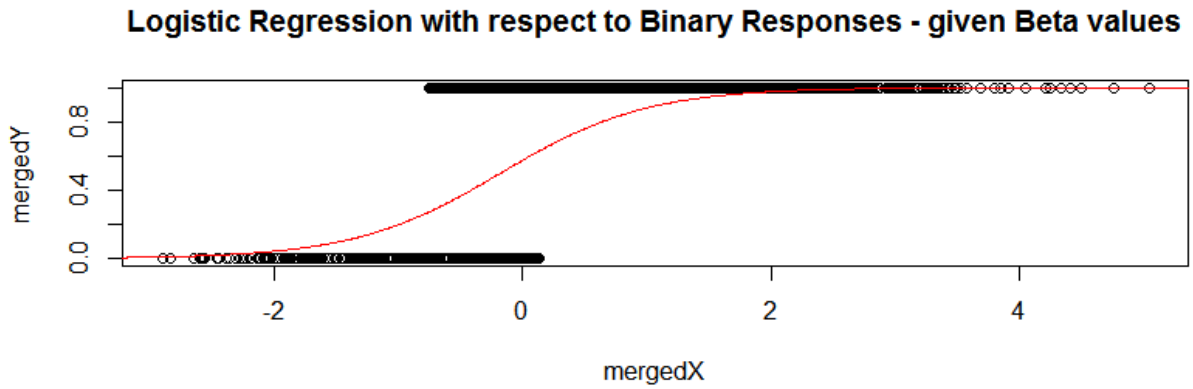
```
#Classifier
classifier <- function(t){
  val <- 1/(1+exp(-B0-B1*t))

  if((val) > 0.5){
    return (1)
  }
  else{
    return (0)
  }
}
predictedValues <- lapply(mergedX,classifier)
count <- 0
for(i in 1:2000){
  if(mergedY[i]!=predictedValues[i]){
    count <- count+1
  }
}
#Count of Misclassified samples
print(count)
## [1] 185
```

Study on Convergence and quality of Estimates

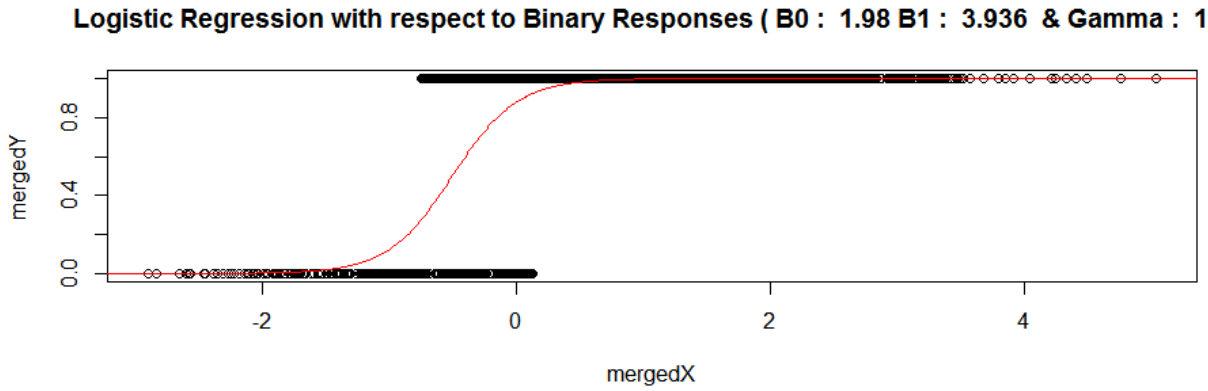
After creating the logistic regression and perform the number of iterations required for the cost function to converge, given our parameters, we compute different loops variating the gamma values and the sample values. The results are shown in the shape of the logistic regression function and in the number of iterations required to reach convergence.

With the original B_0 and B_1, we obtained a logistic regression function which is not very accurate when dividing the binomial response Y of our X values, shown below:

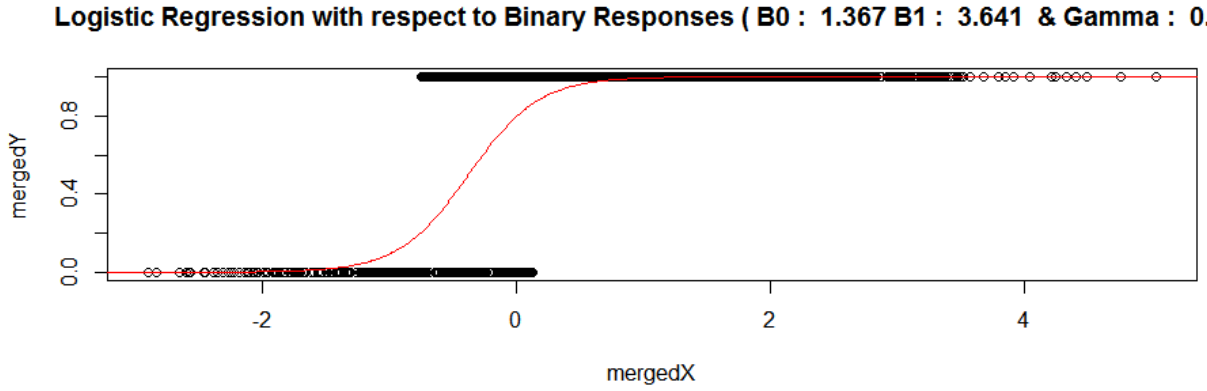


Once we compute the estimates of B_0 and B_1, the resulting logistic regression function is adapting its behavior to the data. Even when 3 different gamma values are used, the results are very similar. However, as expected, the impact of the gamma step is reflected in the number of iterations required to converge.

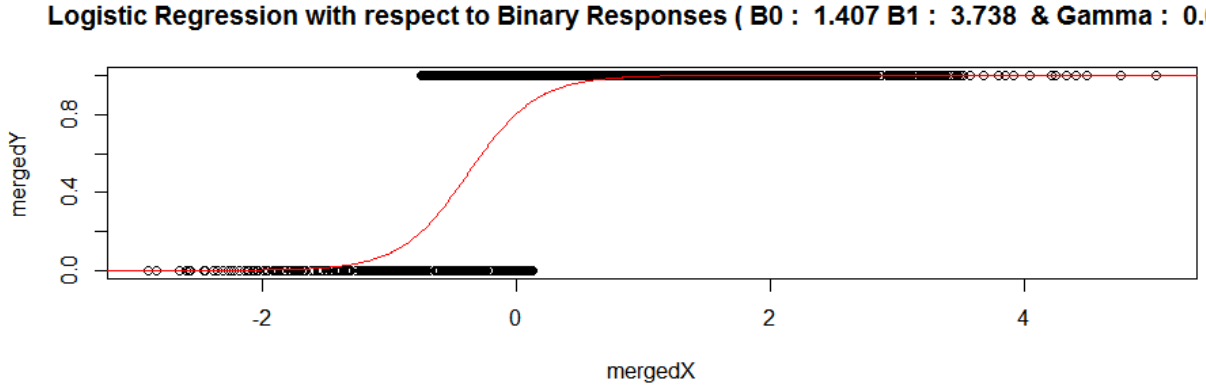
Samples = 1000
Iterations = 3
Cost = -362.514



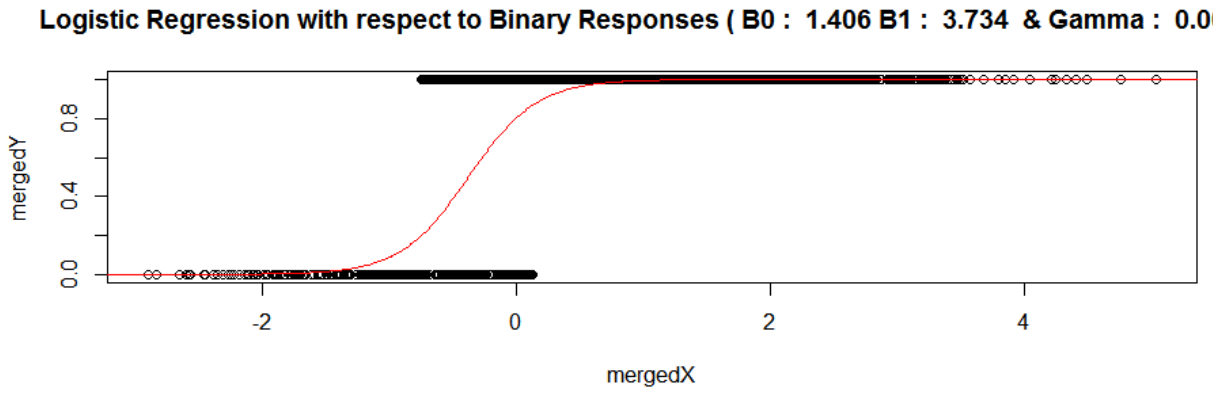
Samples = 1000
Iterations = 25
Cost = -348.409



Samples = 1000
Iterations = 240
Cost = -348.297



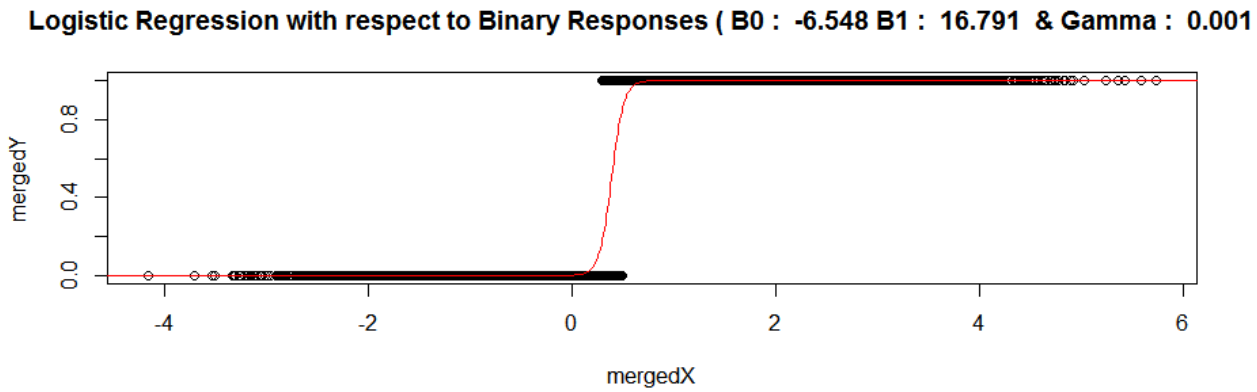
Samples = 1000
Iterations = 2395
Cost = -348.296



Samples 10 000

As additional test, we perform the loop with a random variable of 10 000 samples, with the fixed gamma value used in our original model. As result, the binomial response Y and the final logistic regression function are more accurate. The number of iterations is bigger with respect to our original exercise of 1000 samples, but the regression fits better.

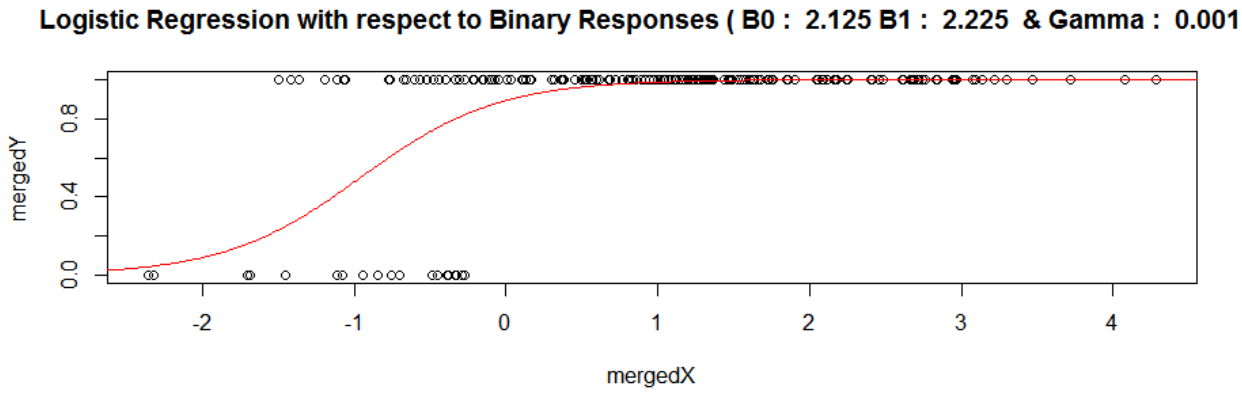
Iterations = 16879
Cost = -1216.9



Samples 100

With a few number of samples, the logistic regression method for the binomial response does not perform well, even when the gamma step is small enough. Also, the merged samples and their response behave in a difficult way to reach a good approximation with the logistic regression.

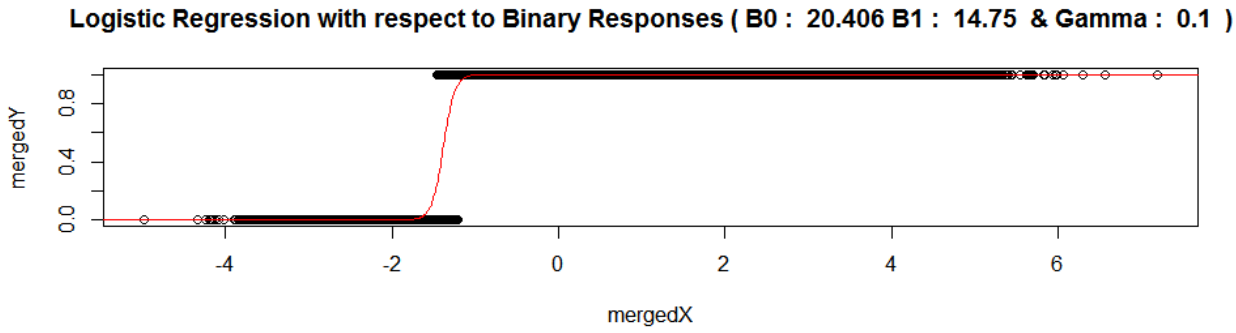
Iterations = 2007
Cost = -36.312



Samples 100 000

With 100 000 samples the model performs well, reaching a clear logistic regression function which adapts to the data which is following a better binomial response. The test was performed only with one value of gamma due to computation constraint, but still giving good results with short iterations.

Iterations = 1133
Cost = -3073.8



Estimating the Probabilities

We calculated the probabilities using Naïve Bayes formula :

$P_{m0}(Y=1)$ => P(Y=1|sample=0) = P(sample=0|Y=1)*P(Y=1)/P(sample=0)
$P_{m1}(Y=1)$ => P(Y=1|sample=1) = P(sample=1|Y=1)*P(Y=1)/P(sample=1)

```
#BAYES
#P(Y=1|sample=0) = P(sample=0|Y=1)*P(Y=1)/P(sample=0)
#P(Y=1|sample=1) = P(sample=1|Y=1)*P(Y=1)/P(sample=1)
tableY <- table(mergedY)
#P(Y=1) = number of 1 responses /  total number of responses
ProbY1 <- tableY[names(tableY)==1]/length(mergedY)
```

P(sample=0|Y=1) is calculated as below

```
#code to get data for below steps
tablem0Y <- table(mergedY[1:1000])
tablem1Y <- table(mergedY[1001:2000])
#P(sample=0|Y=1) = number of 1 responses in first 1000 samples
(X0) / total number of 1 responses in all the samples
```

```
Probm0y1 <-
tablem0Y[names(tablem0Y)==1]/tableY[names(tableY)==1]
```

P(sample=1|Y=1) is calculated as below

```
#P(sample=1|Y=1) = number of 1 responses in next 1000 samples
(X1) / total number of 1 responses in all the samples
Probm1y1 <-
tablem1Y[names(tablem1Y)==1]/tableY[names(tableY)==1]
#Prm0(Y=1)
PY1S0 <- Probm0y1*ProbY1/0.5
#Prm1(Y=1)
PY1S1 <- Probm1y1*ProbY1/0.5
print(PY1S0)
##     1
## 0.863
print(PY1S1)
##     1
## 0.774
```

Neyman-Pearson Test

From the exercise above we have obtained the following probabilities:
$Pr_{m0}(Y=1) = 0.863$
$Pr_{m1}(Y=1) = 0.774$

Yes, it is possible to design a Neyman-Paerson test that decides between H_0 {m=m_0} and H_1 = {m=m_1} using only the response $y_1,\ldots,y_{2n}$ without using the corresponding x_i's. In this case, the hypothesis testing is between two Bernoulli distributions. To perform it, we checked the slide 16 of Lesson 6. The computations are:

$$\frac{L(p_1;y)}{L(p_0;y)} = \frac{P_{m_1}(Y=y)}{P_{m_0}(Y=y)} = \frac{p_1^y(1-p_1)^{1-y}}{p_0^y(1-p_0)^{1-y}} \geq h$$

Now, the logarithm is applied and transformations are performed to get the decision rule:

$$\ln\left(\frac{p_1^y(1-p_1)^{1-y}}{p_0^y(1-p_0)^{1-y}}\right) \geq \ln(h)$$

$$y \underset{H_1}{\overset{H_0}{\lessgtr}} \frac{\ln(h) - (\ln(1-p_1) - \ln(1-p_0))}{\ln(p_1) - \ln(1-p_1) - \ln(p_0) + \ln(1-p_0)} = B$$

If the y that is passed as an input is less or equal than B (which depends on the threshold h, p_0 and p_1) the null hypothesis will be accepted, otherwise, it will be rejected. When it is accepted,

it means that y is presumed to belong to the sample with p p_0. When it is rejected, it means that y is presumed to belong to the sample with p p_1.

False positive probability

Is it possible to satisfy any false positive probability with this test? No, it is not possible, three different cases have been derived to prove so. The following formula is used to obtain the three different possible values of alpha.

$$\begin{aligned} P_{m_0}(Y > B) &= 1 - P_{m_0}(Y \leq B) \\ &= \alpha \end{aligned}$$

The next step is to compute the value of $P_{m_0}(Y \leq B)$. Because this expression can only have three possible values, it is only possible to have three cases.

$$P_{m_0}(Y \leq B) = F_Y(B) = \begin{cases} 0 & \text{if } B < 0 \\ 1 - p_0 & \text{if } 0 \leq B < 1 \\ 1 & \text{if } 1 \leq B \end{cases}$$

These are the levels of false positive that we can satisfy.

Case 1 (B<0 <=> alpha=1)

Because B is less than 0, looking at the formula above, we get that $P_{m_0}(Y \leq B)$ is equal to 0, so alpha is equal to 1.

$$\begin{aligned} P_{m_0}(Y > B) &= 1 - P_{m_0}(Y \leq B) \\ &= 1 - 0 = 1 = \alpha \end{aligned}$$

In this case, for y being 0 or 1, it is going to be greater than B (that is less than 0). As explained before, this rejects the null hypthesis for every possible y.

Case 2 (0<=B<1 => alpha=p_0, p_0<=alpha<1 => 0<=B<1)

Because B belongs to the interval [0,1), looking at the formula above, we get that $P_{m_0}(Y \leq B)$ is equal to 1-p_0, so alpha is equal to p.

$$\begin{aligned} P_{m_0}(Y > B) &= 1 - P_{m_0}(Y \leq B) \\ &= 1 - (1 - p) = p = \alpha \end{aligned}$$

In this case, for y being equal to 0, it is going to be less or equal than B. This accepts the null hypthesis. For y being equal to 1, it is going to greater than B. This rejects the null hypthesis.

Case 3 (p < alpha >= 0 => B>=1, B>=1 => alpha=0)

Because B is greater or equal than 1, looking at the formula above, we get that $P_{m_0}(Y \leq B)$ is equal to 1, so alpha is equal to 0.

$$\begin{aligned} P_{m_0}(Y > B) &= 1 - P_{m_0}(Y \leq B) \\ &= 1 - 1 = 0 = \alpha \end{aligned}$$

In this case, for y being 0 or 1, it is going to be less or equal than B. This accepts the null hypthesis for every possible y.

Now, we proceed to compute the levels of false negative probability given the false positive probability. First, we compute the possible beta values given alpha.

$$P_{m_1}(Y \leq B) = \beta$$

There are three possible values for beta, as shown below:

$$P_{m_1}(Y \leq B) = \beta = \begin{cases} 0 & \text{if } B < 0 \\ 1 - p_1 & \text{if } 0 \leq B < 1 \\ 1 & \text{if } 1 \leq B \end{cases} = \begin{cases} 0 & \text{if } \alpha = 1 \\ 1 - p_1 & \text{if } \alpha = p_0 \\ 1 & \text{if } \alpha = 0 \end{cases}$$