Likelihood Optimization to Linear Regression with R

Fernando Delgado

Contents

Introduction

Linear regression is a statistic model used for predicting a numerical quantity. The parameters of a linear regression model can be estimated using least squares or by maximum likelihood optimization. Maximum likelihood estimation is a probabilistic framework for automatically finding the probability distribution for the observed data.

Through this report, we perform a Likelihood Optimization to linear regression with R.

Loading the Data

First, we simulate a dataframe to work with:

```r
# Create a first column to test
a <- rnorm(1000,0)
df <- a

# Create 5 columns with random data
columns <- cbind(1:5)

# For loop data into columns
for (i in columns){
  name <- paste("var",i, sep="")
  tmp <- rnorm(1000,0)
  df <- data.frame(df, tmp)
  names(df)[names(df) == 'tmp'] <- name
}
```

We create a dataframe with 1000 observations to 5 predictor variables using R's rnorm() assignin a mean of 0 to obtain normalized random data.

Then, to give a little bit of sense to our data, we use USArrests built-in data set and fakeR's simulate_data() to obtain a random target variable of Arrests by Assaults by city. We obtain 1000 random observations that we append to our previous dataset. The final result is a data frame with 1000 observations for 5 random predictor variables to 1 target variable (number of arrests by assault by city).

```
## Classes 'data.table' and 'data.frame':   1000 obs. of  6 variables:
##  $ var1   : num  1.2507 -0.4434 -0.0904 -0.9828 0.2991 ...
##  $ var2   : num  1.023 -0.76 -0.799 0.511 -0.803 ...
##  $ var3   : num  0.465 2.447 -0.347 -0.721 1.491 ...
##  $ var4   : num  0.686 -1.393 -0.548 -0.859 -0.607 ...
##  $ var5   : num  -0.00569 2.93659 -0.43356 0.32744 0.44956 ...
##  $ Assault: num  187 171 242 102 160 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Linear Regression

To optimize a function, it is important to consider what function we should optimize. To work with a linear regression function in this case, we know that it follows a normal distribution with a mean of 0 and an unknown standard deviation.

$\sum_{i=1}^{i=n} R_i\tilde{N}(0, s)$

Where R equals:

$R_i = y1 - \hat{y}_i$

Therefore, the objective is obtaining a function of y_i which minimizes the residuals and shows the best coefficients for our function.

With this in mind, we write an optimization function with the following code:

```r
#Define likelihood function to optimise
ll_lm <- function(par, y, x1, x2, x3, x4, x5){

  alpha <- par[1]
  beta1 <- par[2]
  beta2 <- par[3]
  beta3 <- par[4]
  beta4 <- par[5]
  beta5 <- par[6]
  sigma <- par[7]

  R = y - alpha - beta1 * x1 - beta2 * x2 - beta3 * x3 - beta4 * x4 - beta5 * x5

  -sum(dnorm(R, mean = 0, sigma, log = TRUE))
}
```

Alpha refers to our target variable, and the 5 betas represent each predictor variable.

In order to use the optim() function, it must have par arguments. Par arguments need a vector with guesses for all unknown parameters. In our code above, par arguments include initial values in all 7 of the unknown parameters.

It is important to mention that by using dnorm() we obtain logarithmic values. This helps to sum the single likelihood values instead of the product.

The linear model we are fitting looks like this:

$E(Y|X) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$

Therefore, the residuals are calculated like this:

$R = y - \alpha - \beta_1 x_1 - \beta_2 x_2 - \beta_3 x_3 - \beta_4 x_4 - \beta_5 x_5$

Moreover, since residuals are following a normal distribution with a mean of 0, what's left Is to find the standard deviation that best fits our data. Hence, we minimize the sum of errors with optim() command by using a minus sign before the sum.

However, before running the optimization, we also estimate our coefficients by simply calculating the mean of each variable:

```
#Estimate Betas
est_alpha <- mean(fake_arrests$Assault)
est_beta1 <- mean(fake_arrests$var1)
est_beta2 <- mean(fake_arrests$var2)
est_beta3 <- mean(fake_arrests$var3)
est_beta4 <- mean(fake_arrests$var4)
est_beta5 <- mean(fake_arrests$var5)
est_sigma <- sd(fake_arrests$Assault)
```

To keep it simple, we do it manually, but this could be looped for a larger dataset.

Moving forward, we optimize our model searching our maximum likelihood estimates for the different coefficients. We introduce our function ll_lm(), the estimated coefficients and the data to be optimized:

```
mle_par <- optim(fn = ll_lm,
                 par = c(alpha = est_alpha,
                         beta1 = est_beta1,
                         beta2 = est_beta2,
                         beta3 = est_beta3,
                         beta4 = est_beta4,
                         beta5 = est_beta5,
                         sigma = est_sigma),
                 y = fake_arrests$Assault,
                 x1 = fake_arrests$var1,
                 x2 = fake_arrests$var2,
                 x3 = fake_arrests$var3,
                 x4 = fake_arrests$var4,
                 x5 = fake_arrests$var5)
```

Finally, we obtain our estimated coefficients

```
##      alpha      beta1      beta2      beta3      beta4      beta5      sigma
## 153.209290  -1.310258   1.612228   1.133520  -1.654349   5.703489  74.355542
```

Validation

If we compare the estimate with the result of the lm() command for the same model, we observe some slight differences in the coefficients. However, since they are rather small it is probably due to our initial guesses for the parameters.

```
##
## Call:
## lm(formula = Assault ~ var1 + var2 + var3 + var4 + var5, data = fake_arrests)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -161.10  -48.62  -14.40   45.97  167.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  153.190      2.363  64.816   <2e-16 ***
## var1          -1.419      2.341  -0.606   0.5445
## var2           1.573      2.317   0.679   0.4974
## var3           1.029      2.363   0.436   0.6633
## var4          -1.766      2.364  -0.747   0.4553
## var5           5.695      2.327   2.448   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74.61 on 994 degrees of freedom
## Multiple R-squared:  0.007842,   Adjusted R-squared:  0.002851
## F-statistic: 1.571 on 5 and 994 DF,  p-value: 0.1654
```

References

Creating fake simulated data was inspired from this following post:

https://rviews.rstudio.com/2020/09/09/fake-data-with-r/

Maximization of likelihood was inspired by this following post:

https://www.joshua-entrop.com/post/optim_linear_reg/