Logistic Regression and Regularization

Applied Machine Learning for Educational Data Science

true

10/26/2021

Contents

[Updated: Fri, Oct 29, 2021 - 18:25:39]

Overview of the Logistic Regression

Logistic regression is a type of model that can be used to predict a binary outcome variable. Linear regression and logistic regression are indeed members of the same family of models called *generalized linear models*. While linear regression can also technically be used to predict a binary outcome, the bounded nature of a binary outcome, [0,1], makes the linear regression solution suboptimal. Logistic regression is a more appropriate model and takes the bounded nature of the binary outcome into account when making predictions.

The binary outcomes can be coded in a variety of ways in the data such as 0 vs 1, True vs False, Yes vs. No, Success vs. Failure. The rest of the notes, it is assumed that the category of interest to predict is represented by 1s in the data.

The notes in this section will first introduce a suboptimal solution to predict a binary outcome by fitting a linear probability model using linear regression and discuss the limitations of this approach. Then, the logistic regression model and its estimation will be demonstrated. Finally, different regularization approaches for the logistic regression will be discussed.

Throughout these notes, we will use the Recidivism dataset from the NIJ competition to discuss different aspects of logistic regression and demonstrations. This data and variables in this data were discussed in detail in Lecture 1a and Lecture 2a. The outcome of interest to predict in this dataset is whether or not an individual will be recidivated in the second year after initial release. In order to make demonstrations easier, I randomly sample 20 observations from this data. Six observations in this data have a value of 1 for the outcome (recidivated) while 14 observations have a value of 0 (not recidivated).

```
# Download the random sample of 20 observations from the recidivism dataset

recidivism <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/mai
                       header=TRUE)

# Outcome variable

table(recidivism$Recidivism_Arrest_Year2)
```

```
 0  1
14  6
```

Linear Probability Model

Linear probability model is just fitting a typical regression model to a binary outcome. When the outcome is binary, the predictions from a linear regression model can be considered as probability of outcome being equal to 1,

$$\hat{Y} = P(Y = 1).$$

Suppose that we want to predict the recidivism in the second year (`Recidivism_Arrest_Year2`) by using the number of dependents they have. Then, we could fit this using the `lm` function.

```
mod <- lm(Recidivism_Arrest_Year2 ~ 1 + Dependents,
          data = recidivism)

summary(mod)
```

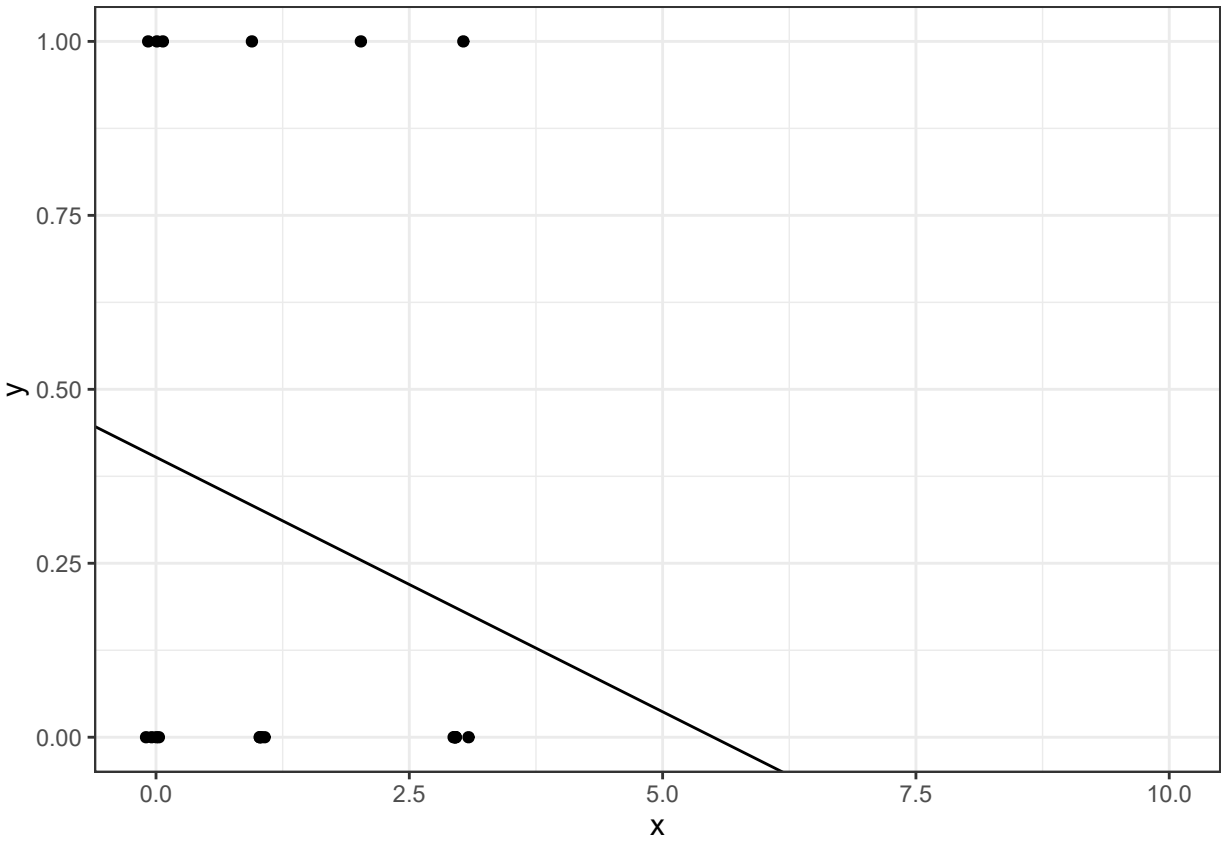
```
Call:
lm(formula = Recidivism_Arrest_Year2 ~ 1 + Dependents, data = recidivism)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-0.4024 -0.3293 -0.1829  0.5976  0.8171

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.40244    0.15665   2.569   0.0193 *
Dependents  -0.07317    0.08256  -0.886   0.3872
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4728 on 18 degrees of freedom
Multiple R-squared:  0.04181,   Adjusted R-squared:  -0.01142
F-statistic: 0.7855 on 1 and 18 DF,  p-value: 0.3872
```

The intercept is 0.402 and the slope for the predictor `Dependents` is -.073. We can interpret the intercept and slope as the following for this example. Note that the predicted values from this model now can be interpreted as probability predictions because the outcome is binary.

- Intercept (0.402): When the number of dependent is equal to 0, the probability of being recidivated in Year 2 is 0.402.
- Slope (-0.073): For every additional dependent (one unit increase in X) the individual has, the probability of being recidivated in Year 2 is reduced by .07.

The intercept and slope still represent the best fitting line to our data, and this fitted line can be shown here.



Now, suppose we want to calculate the model predicted probability of being recidivated in Year 2 for different number of dependents a parolee has. Let's assume that the number of dependents can be any number from 0 to 10. What would be the predicted probability of being recidivated in Year 2 for a parolee with 8 dependents?

```
X <- data.frame(Dependents = 0:10)

predict(mod,newdata = X)
```

```
         1          2          3          4          5          6
 0.40243902  0.32926829  0.25609756  0.18292683  0.10975610  0.03658537
         7          8          9         10         11
-0.03658537 -0.10975610 -0.18292683 -0.25609756 -0.32926829
```

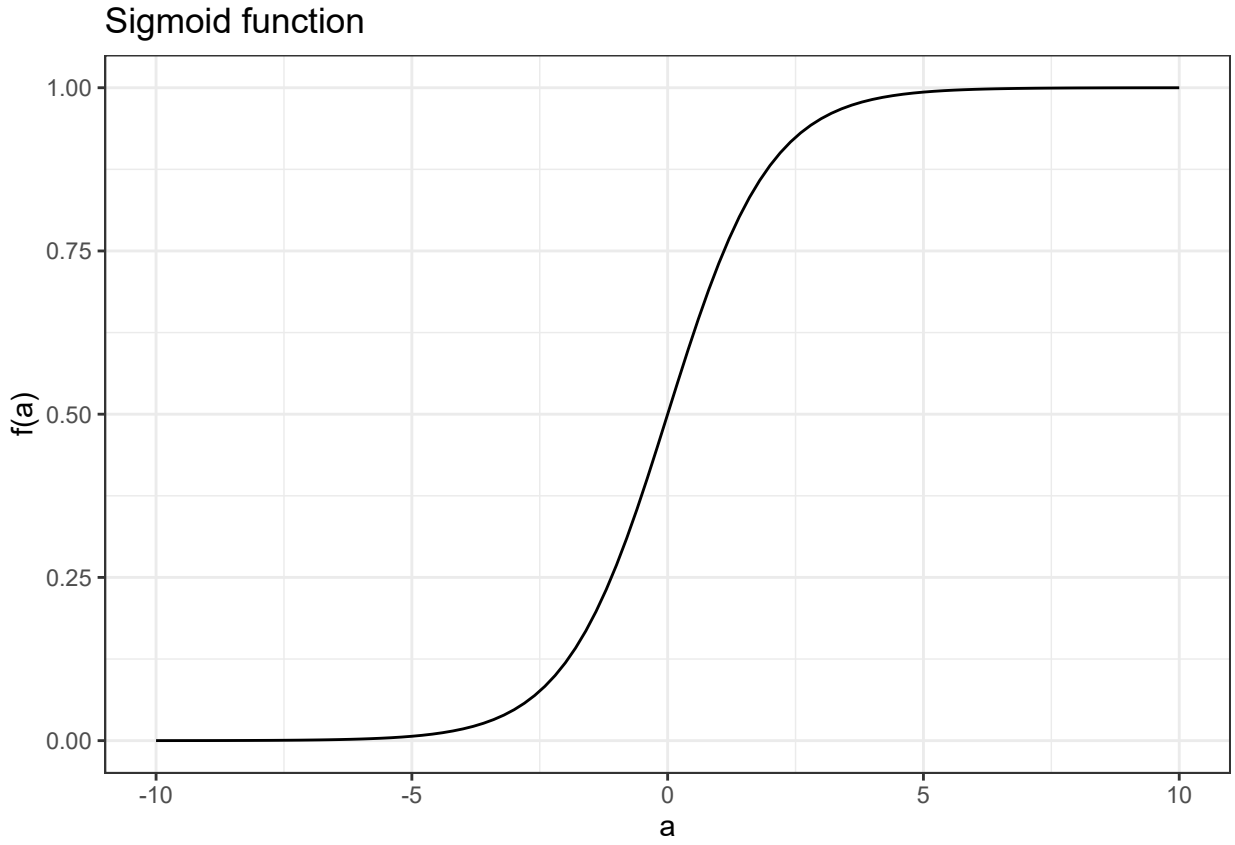
It is not reasonable for a probability prediction to be negative. One of the major issues with using a linear regression to predict a binary outcome using a linear-probability model is that the model predictions can easily be go outside of the boundary [0,1] and yield unreasonable predictions. So, a linear regression model may not necessarily the best tool to predict a binary outcome. We should use a model that respect the boundaries of the outcome variable.

Description of Logistic Regression Model

In order to overcome the limitations of the linear probability model, we bundle our prediction model in a sigmoid function. Suppose there is real-valued function of a such that

$$f(a) = \frac{e^a}{e^a + 1}.$$

It can be shows than the output of this function is always bounded to be between 0 and 1 regardless of the value of a. Therefore, sigmoid function is an appropriate choice for the logistic regression because it assures that the output is always bounded between 0 and 1.



If we revisit the previous example, we can specify a logistic regression model to predict the probability of being recidivated in Year 2 as the following by using the number of dependents a parolee has as the predictor,

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X}}{e^{\beta_0 + \beta_1 X} + 1}.$$

When the values of predictor variable is entered into the equation, the model output can be directly interpreted as the probability of the binary outcome being equal to 1 (or whatever category and meaning a value of one represents). Then, we assume that the actual outcome follows a binomial distribution with the predicted probability.

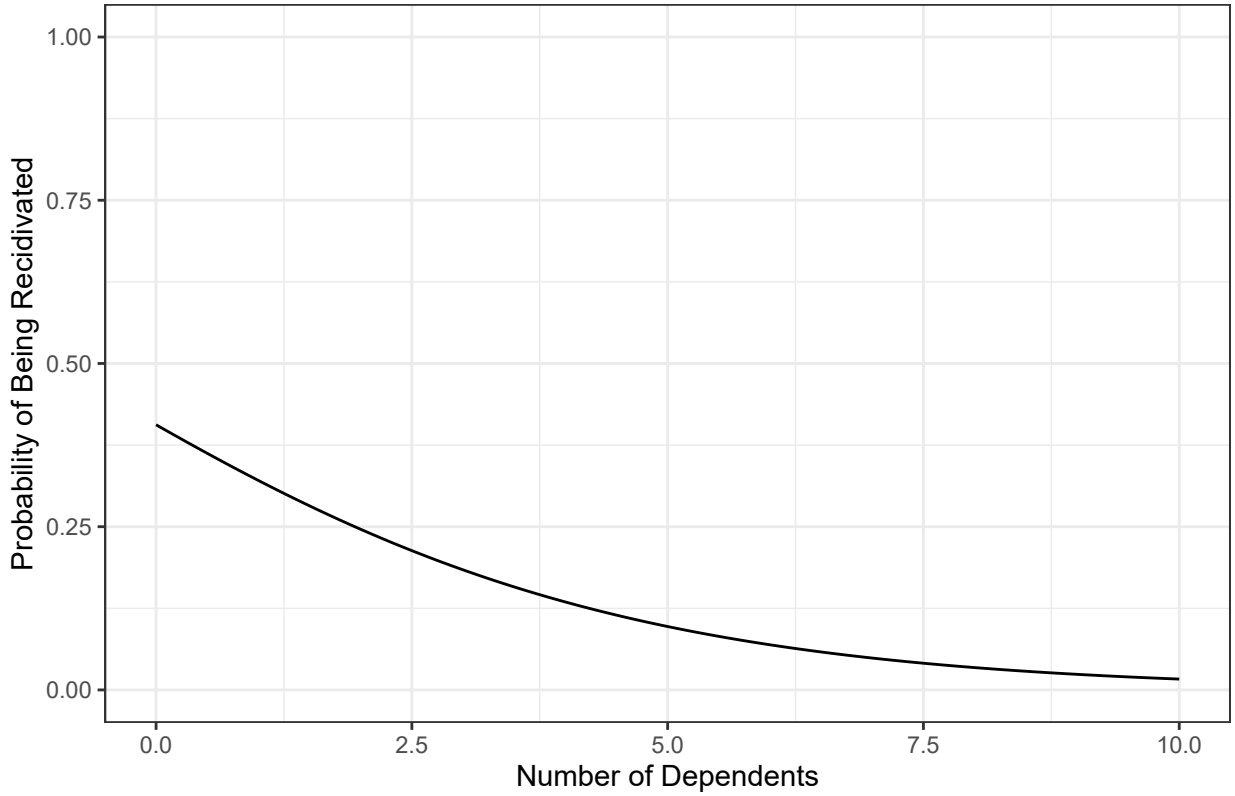
$$P(Y = 1) = p$$
$$Y \sim Binomial(p)$$

Suppose the coefficient estimates of this model are β_0=-0.38 and β_1=-0.37. Then, for instance, we can compute the probability of being recidivated for a parolee with 8 dependents as the following:

$$P(Y=1) = \frac{e^{(-0.38-0.37\times 8)}}{e^{(-0.38-0.37\times 8)+1}} = 0.034.$$

```
b0 = -0.38
b1 = -0.37

x = 0:10

y = exp(b0+b1*x)/(1+exp(b0+b1*x))

data.frame(number.of.dependents = x, probability=y)
```

```
   number.of.dependents probability
1                     0  0.40612690
2                     1  0.32082130
3                     2  0.24601128
4                     3  0.18392173
5                     4  0.13470305
6                     5  0.09708864
7                     6  0.06913842
8                     7  0.04879972
9                     8  0.03422416
10                    9  0.02389269
11                   10  0.01662636
```



In its original form, it is difficult to interpret the parameters of the logistic regression because a one unit increase in the predictor is not anymore linearly related the probability of outcome being equal to 1 due to the nonlinear nature of the sigmoid function. Most common presentation of logistic regression is obtained after a bit of algebraic manipulation to rewrite the model equation. The logistic regression model above can also be specificed as the following without any loss of meaning as they are mathematically equivalent.

$$ln\left[\frac{P(Y=1)}{1-P(Y=1)}\right] = \beta_0 + \beta_1 X.$$

The term on the left side of the equation is known as the **logit**. So, when the outcome is a binary variable, the logit transformation of the probability that the outcome is equal to 1 can be represented as a linear equation. This provides a more straightforward interpretation. For instance, we can know say that when the number of dependents is equal to zero, the predicted logit is equal to -0.38 (intercept), and for every additional dependent the logit decreases by 0.37 (slope).

It is also common to transform the logit to odds when interpreting the parameters. For instance, we can say that when the number of dependents is equal to zero, the odds of being recidivated is 0.68 ($e^{-0.38}$), and for every additional dependent the odds of being recidivated is reduced by 31% ($1 - e^{-0.37}$).

The right side of the equation can be expanded by adding more predictors, adding polynomial terms of the predictors, or adding interactions among predictors. A model with only main effects of P predictors can be written as

$$ln\left[\frac{P(Y=1)}{1-P(Y=1)}\right] = \beta_0 + \sum_{p=1}^{P} \beta_p X_p,$$

and the coefficients can be interpreted as

- β_0: the predicted logit when the values for all the predictor variables in the model are equal to zero. e^{β_0}, the predicted odds of outcome being equal to 1 when the values for all the predictor variables in the model are equal to zero.
- β_p: the change in the predicted logit for one unit increase in X_p when the values for all other predictors in the model are held constant. For every one unit in increase in X_p, the odds of the outcome being equal to 1 is multiplied by e^{β_p} when the values for all other predictors in the model are held constant. In other words, e^{β_p} is odds ratio, the ratio of odds at $\beta_p = a + 1$ to the odds at $\beta_p = a$.

It is important that you get familiar with the three concepts (probability, odds, logit) and how these three are related to each other for interpreting the logistic regression parameters.

NOTE

Sigmoid function is not the only tool to be used for modeling a binary outcome. One can also use the cumulative standard normal distribution function, $\phi(a)$, and the output of $\phi(a)$ is also bounded between 0 and 1. When ϕ is used to to transform the prediction model, this is known as **probit regression** and it serves the same purpose as the logistic regression, which is to predict probability of a binary outcome being equal to 1. However, it is always easier and more pleasant to work with logarithmic functions, and logarithmic functions has nice computational properties. Therefore, logistic regression is more commonly used than the probit regression.

Model Estimation

The concept of likelihood It is important to understand the concept of likelihood for estimating the coefficients of a logistic regression model. We will consider a simple example of flipping coins for this. Suppose you flip the same coin 20 times and observe the following outcome. We don't necessarily know whether or not this is a fair coin in which the probability of observing a head or tail is equal to 0.5.

$$\mathbf{Y} = (H, H, H, T, H, H, H, T, H, T)$$

Suppose that we define p as the probability of observing a head when we flip this coin. By definition, the probability of observing a tail is $1 - p$.

$$P(Y = H) = p$$

$$P(Y = T) = 1 - p$$

Then, we can calculate the likelihood of our observations of heads and tails as a function of p.

$$\mathfrak{L}(\mathbf{Y}|p) = p \times p \times p \times (1-p) \times p \times p \times p \times (1-p) \times p \times (1-p)$$

For instance, if we say that this is a fair coin and therefore p is equal to 0.5, then the likelihood of observing 7 heads and 3 tails would be equal to

$$\mathfrak{L}(\mathbf{Y}|p = 0.5) = 0.5 \times 0.5 \times 0.5 \times (1-0.5) \times 0.5 \times 0.5 \times 0.5 \times (1-0.5) \times 0.5 \times (1-0.5) = 0.0009765625$$

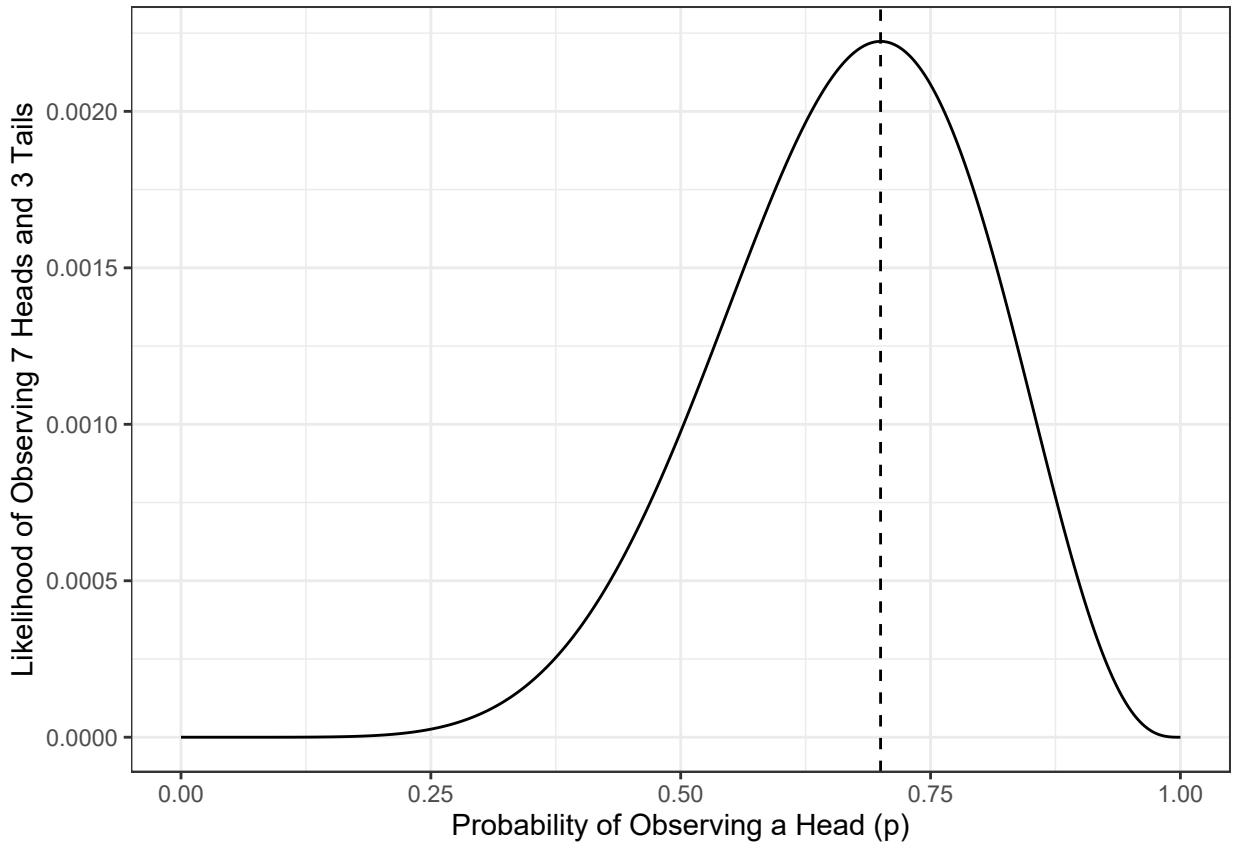
On the other hand, another person can say this is probably not a fair coin and the p should be something higher than 0.5. How about 0.65?

$$\mathfrak{L}(\mathbf{Y}|p = 0.65) = 0.65 \times 0.5 \times 0.65 \times (1-0.65) \times 0.65 \times 0.65 \times 0.65 \times (1-0.65) \times 0.65 \times (1-0.65) = 0.00210183$$

We can say that based on our observation, an estimate of p being equal to 0.65 is more likely than an estimate of p being equal to 0.5. Our observation of 7 heads and 3 tails is more likely if we estimate p as 0.65 rather than 0.5.

Maximum likelihood estimation (MLE) Then, what would be the best estimate of p given our observed data (7 heads and 3 tails). We can try every possible value of p between 0 and 1, and calculate the likelihood of our data ($\mathbf{Y}$). Then, we can pick the value that makes our data most likely (largest likelihood) to observe as our best estimate. This would be called the maximum likelihood estimate of p given the data we observed.

```
p <- seq(0,1,.001)
L <- p^7*(1-p)^3

ggplot()+
  geom_line(aes(x=p,y=L)) +
  theme_bw() +
  xlab('Probability of Observing a Head (p)')+
  ylab('Likelihood of Observing 7 Heads and 3 Tails')+
  geom_vline(xintercept=p[which.max(L)],lty=2)
```



We can show that the p value that makes the likelihood largest is 0.7, and the likelihood of observing 7 heads and 3 tails is 0.002223566 when p is equal to 0.7. Therefore, the maximum likelihood estimate of probability of observing a head for this particular coin is 0.7 given the 10 observations we have made.

```
L[which.max(L)]
```

```
[1] 0.002223566
```

```
p[which.max(L)]
```

```
[1] 0.7
```

Note that our estimate can change and be updated if we continue collecting more data by flipping the same coin and record our observations.

The concept of loglikelihood The computation of likelihood requires the multiplication of so many p values, and When you multiply values between 0 and 1, the result gets smaller and smaller. This creates problems when you multiply so many of these small p values due the maximum precision any computer can handle. For instance, you can see what is the minimum number that can be represented in R and meets the requirements of IEEE 754 technical standard.

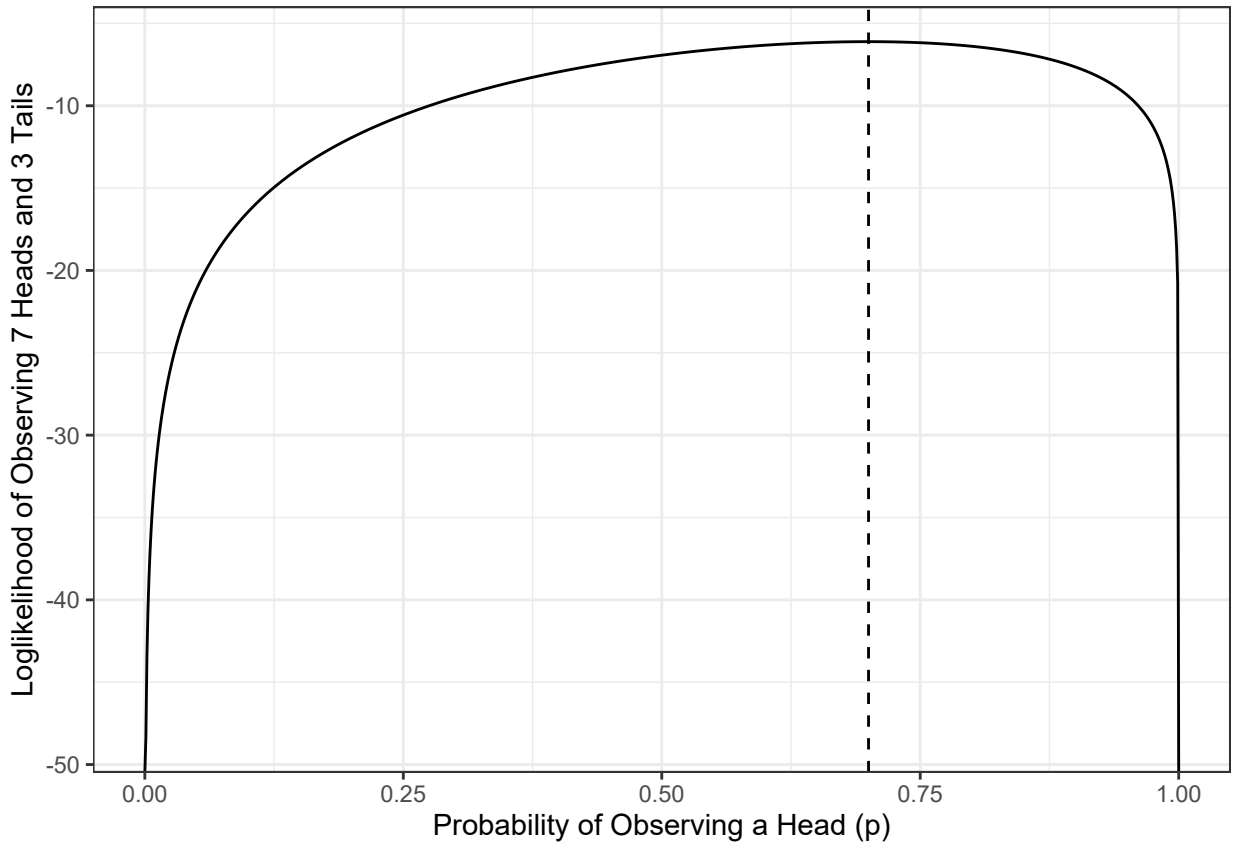
```
.Machine$double.xmin
```

```
[1] 2.225074e-308
```

When you have hundreds of thousands of observations, it is probably not a good idea to directly work with likelihood. Instead, we work with the log of likelihood. This has two main advantages:

- We are less concerned about the precision of small numbers our computer can handle.
- Loglikelihood has nicer mathematical properties to work with for optimization problems (log of product of two numbers is equal to the sum of log of the two numbers).
- The point that maximizes likelihood also the same number that maximizes the loglikelihood, so our end results (MLE estimate) does not care if we use loglikelihood instead of likelihood.

```
p <- seq(0,1,.001)
logL <- log(p)*7 + log(1-p)*3

ggplot()+
  geom_line(aes(x=p,y=logL)) +
  theme_bw() +
  xlab('Probability of Observing a Head (p)')+
  ylab('Loglikelihood of Observing 7 Heads and 3 Tails')+
  geom_vline(xintercept=p[which.max(logL)],lty=2)
```



```
logL[which.max(logL)]
```

```
[1] -6.108643
```

```
p[which.max(logL)]
```

```
[1] 0.7
```

MLE for Logistic Regression coefficients Now, we can apply these concepts to estimate the logistic regression coefficients. Let's revisit our previous example in which we predict the probability of being recidivated in Year 2 given the number of dependents a parolee has. Our model can be written as the following.

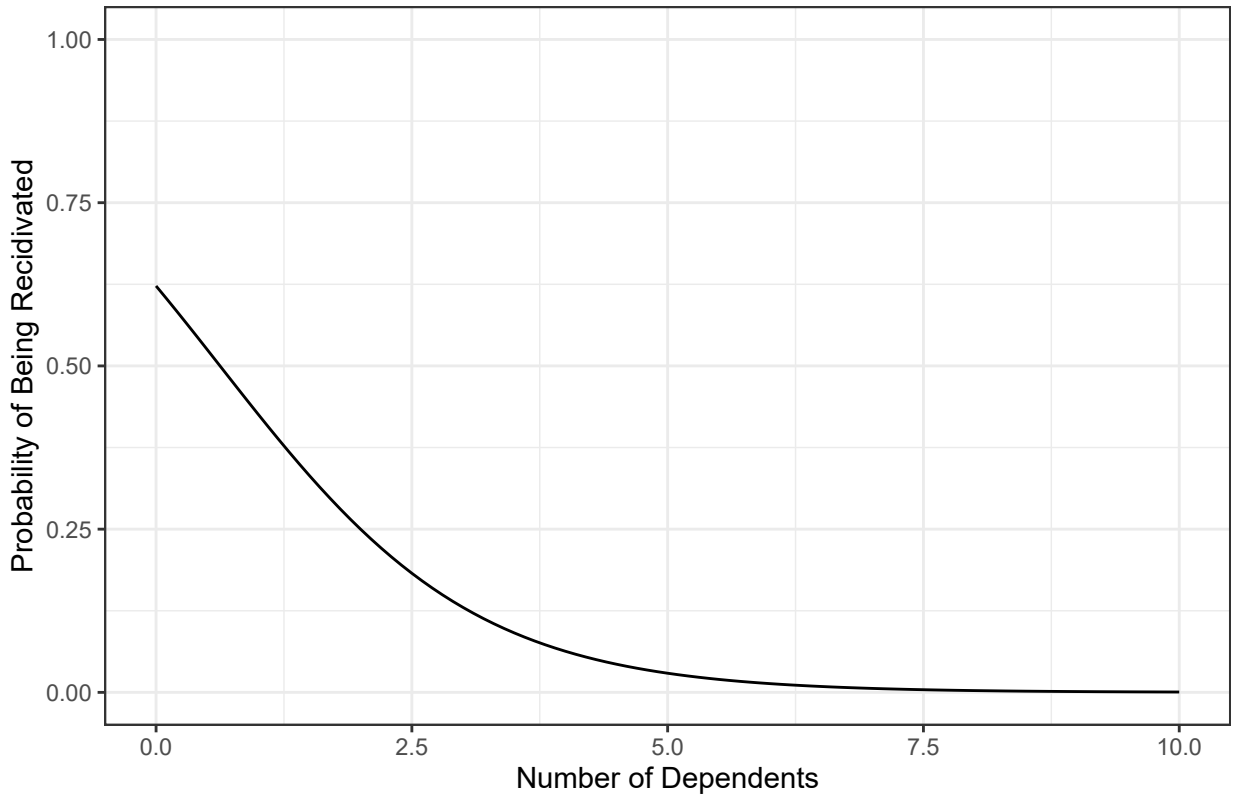
$$ln\left[\frac{P_i(Y=1)}{1-P_i(Y=1)}\right] = \beta_0 + \beta_1 X_i.$$

Note that X and P has a subscript i to indicate that each individual may have a different X value, and therefore each individual will have a different probability. Our observed outcome is a set of 0s and 1s. Remember that there are 6 individuals recidivated (Y=1) and 14 individuals not recidivated (Y=0).

```
recidivism$Recidivism_Arrest_Year2
```

```
 [1] 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0
```

Given a set of coefficients, $\{\beta_0, \beta_1\}$, we can calculate the logit for every observation using the model equation, and then transform this logit to a probability,$P_i(Y=1)$. Finally, we can calculate the log of the probability for each obsrvation, and sum them across observations to obtain the loglikelihood of observing this set of observations (14 ones and 6 zeros). Suppose that we have two guesstimates for $\{\beta_0, \beta_1\}$, and they are 0.5 and -0.8, respectively. These coefficients imply the following predicted model.



If these two coefficients are our estimates, how likely would it be to observe the outcome in our data given the number of dependents. The below R code first finds the predicted logit for every single observation assuming that $\beta_0 = 0.5$ and $\beta_1 = -0.8$.

```
b0 = 0.5
b1 = -0.8

x = recidivism$Dependents
y = recidivism$Recidivism_Arrest_Year2

pred_logit <- b0 + b1*x

pred_prob1 <- exp(pred_logit)/(1+exp(pred_logit))
pred_prob0 <- 1 - pred_prob1

data.frame(Dependents      = x,
           Recidivated     = y,
           Prob1 = pred_prob1,
           Prob0 = pred_prob0)
```

```
  Dependents Recidivated     Prob1     Prob0
1          0           1 0.6224593 0.3775407
2          1           0 0.4255575 0.5744425
3          2           1 0.2497399 0.7502601
4          1           0 0.4255575 0.5744425
```

```
5           1           0 0.4255575 0.5744425
6           0           0 0.6224593 0.3775407
7           0           0 0.6224593 0.3775407
8           1           1 0.4255575 0.5744425
9           3           1 0.1301085 0.8698915
10          0           0 0.6224593 0.3775407
11          1           0 0.4255575 0.5744425
12          0           1 0.6224593 0.3775407
13          0           0 0.6224593 0.3775407
14          3           0 0.1301085 0.8698915
15          3           0 0.1301085 0.8698915
16          0           1 0.6224593 0.3775407
17          3           0 0.1301085 0.8698915
18          3           0 0.1301085 0.8698915
19          3           0 0.1301085 0.8698915
20          3           0 0.1301085 0.8698915
```

```
logL <-  y*log(pred_prob1) + (1-y)*log(pred_prob0)

sum(logL)
```

```
[1] -12.65336
```

We can sumamrize this by saying that if our model coefficients are $\beta_0 = 0.5$ and $\beta_1 = -0.8$, then the log of likelihood of observing the outcome in our data would be -12.65.

$$\mathbf{Y} = (1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0)$$

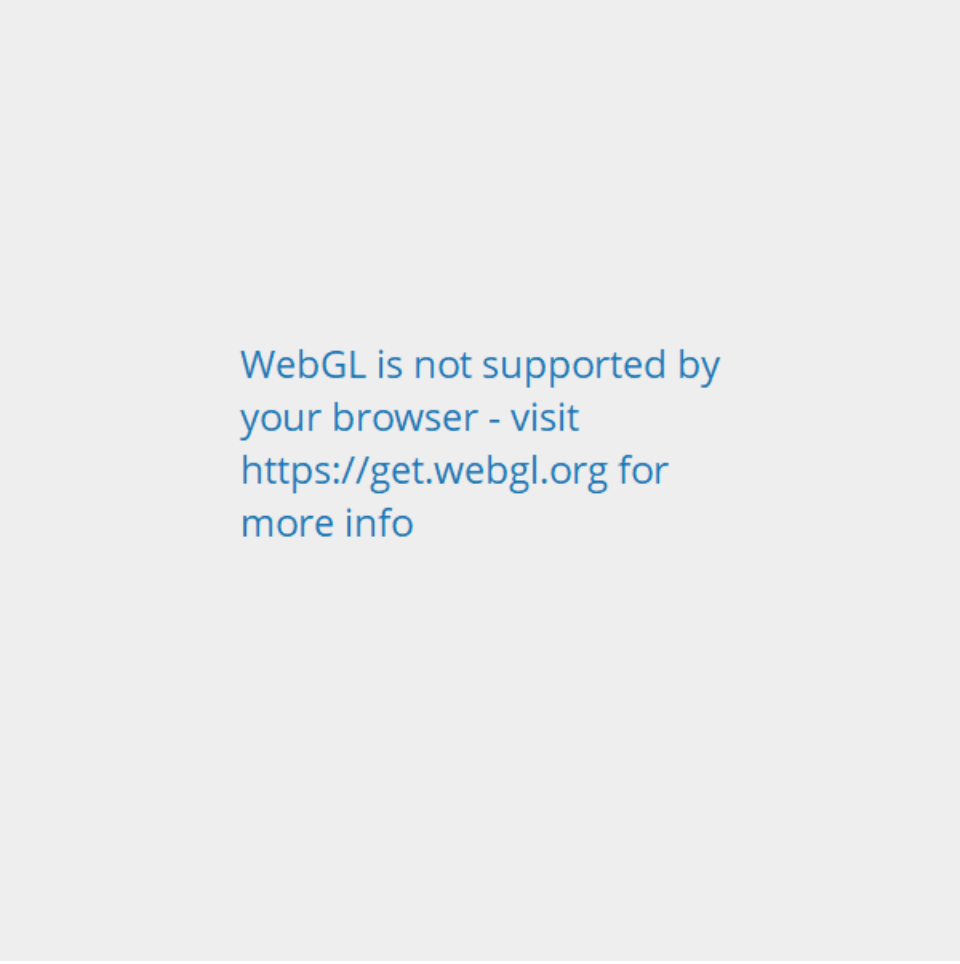
$$\mathfrak{log}\mathfrak{L}(\mathbf{Y}|\beta_0 = 0.5, \beta_1 = -0.8) = -12.65$$

The critical question to ask is whether or not there is another pair of values we can assign to β_0 and β_1 that would provide a higher likelihood of data. If there is, then they would be better estimates for our model. If we can find such a pair with the maximum loglikelihood of data, then they would be our maximum likelihood estimates for the given model.

We can approach this problem in a very crude way to gain some intuition about what Maximum Likelihood Estimation is about. Now, suppose that a reasonable range of values for β_0 is from -1 to 1 and a reasonable range of values for β_1 is also from -1 to 1. Let's think about every possible combinations of values for β_0 and β_1 within these ranges with increments of .01. Then, let's calculate the loglikelihood of data for every possible combination and plot these in a 3D plot as a function of β_0 and β_1.

```
grid    <- expand.grid(b0=seq(-1,1,.01),b1=seq(-1,1,.01))
grid$logL <- NA

for(i in 1:nrow(grid)){

  x = recidivism$Dependents
  y = recidivism$Recidivism_Arrest_Year2

  pred_logit    <- grid[i,]$b0 + grid[i,]$b1*x
  pred_prob1    <- exp(pred_logit)/(1+exp(pred_logit))
  pred_prob0    <- 1 - pred_prob1
```

```
  logL          <- y*log(pred_prob1) + (1-y)*log(pred_prob0)
  grid[i,]$logL <- sum(logL)

  print(i)
}


require(plotly)

plot_ly(grid, x = ~b0, y = ~b1, z = ~logL,
        marker = list(color = ~logL,
                      showscale = TRUE,
                      cmin=min(grid$logL),
                      cmax=max(grid$logL),cauto=F),
        width=600,height=600) %>%
  add_markers()
```



What is the maximum point of this surface? Our crude search indicates that it is -11.78708, and the set of β_0 and β_1 coefficients that make the observed data most likely is -0.38 and -0.37.

```
grid[which.max(grid$logL),]
```

```
         b0    b1      logL
12726 -0.38 -0.37 -11.78708
```

Therefore, given our dataset with 20 observations, our maximum likelihood estimates for the coefficients of the logistic regression model above are -0.38 and -0.37.

$$ln\left[\frac{P_i(Y=1)}{1-P_i(Y=1)}\right] = -0.38 - 0.37 \times X_i.$$

Logistic Loss function Below is a compact way of writing likelihood and loglikeihood in mathematical notation. For simplification purposes, we write P_i to represent $P_i(Y=1)$.

$$\mathfrak{L}(\mathbf{Y}|\boldsymbol{\beta}) = \prod_{i=1}^{N} P_i^{y_i} \times (1-P_i)^{1-y_i}$$

$$\mathfrak{log}\mathfrak{L}(\mathbf{Y}|\boldsymbol{\beta}) = \sum_{i=1}^{N} y_i \times ln(P_i) + (1-y_i) \times ln(1-P_i)$$

The final equation above, $\mathfrak{log}\mathfrak{L}(\mathbf{Y}|\boldsymbol{\beta})$, is known as the **logistic loss** function. By finding the set of coefficients in a model, $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_P)$, that maximizes this quantity, we obtain the maximum likelihood estimates of the coefficients for the logistic regression model.

Unfortunately, the naive crude search we applied above would be a bad solution when you have a complex model with so many predictors. Another unfortunate thing is that there is no closed form solution (as we had for the linear regression) for the logistic regression. Therefore, the only way to estimate the logistic regression coefficients is to use numerical approximations and computational algorithms to maximize the logistic loss function. Luckily, we have tools available to accomplish this task.

NOTE

One may wonder why we do not use least square estimation and minimize the sum of squared residuals when estimating the coefficients of the logistic regression model.

```
grid    <- expand.grid(b0=seq(-1,1,.01),b1=seq(-1,1,.01))
grid$logL <- NA

for(i in 1:nrow(grid)){

  x = recidivism$Dependents
  y = recidivism$Recidivism_Arrest_Year2

  pred_logit    <- grid[i,]$b0 + grid[i,]$b1*x
  pred_prob1    <- exp(pred_logit)/(1+exp(pred_logit))
  pred_prob0    <- 1 - pred_prob1
  logL          <- y*log(pred_prob1) + (1-y)*log(pred_prob0)
  grid[i,]$logL <- sum(logL)
```

```
  print(i)
}


require(plotly)

plot_ly(grid, x = ~b0, y = ~b1, z = ~logL,
        marker = list(color = ~logL,
                      showscale = TRUE,
                      cmin=min(grid$logL),
                      cmax=max(grid$logL),cauto=F),
        width=600,height=600) %>%
  add_markers()
```

---

glm function

Building a Prediction Model for Recidivism

Initial Data Preparation

Train/Test Split

Model Fitting with the caret package

Regularization in Logistic Regression

Ridge Penalty

Model Fitting with the caret package

Variable Importance

Lasso Penalty

Model Fitting with the caret package

Variable Importance

Elastic Net

Model Fitting with the caret package

Variable Importance

Using the Prediction Model for Future Observations