```r
#### Custom Logistic Regression

### Setup
set.seed(12345)
data.iris = as.data.frame(iris)
ids = data.iris$Species %in% c("virginica", "versicolor")
y.iris = ifelse(data.iris[ids,]$Species=="virginica", 1, 0)
X.iris = data.iris[ids,c(1,3)] # Select only Lengths to be able to
plot
X.iris = as.matrix(X.iris / max(X.iris)) # Rescale for faster
convergence

### Functions

## Returns the sigmoid value of given number z
sigmoid = function(z){
  return(1 / (1 + exp(-z)))
}

## Returns the logistic regression β.hat
logistic_reg = function(X, y, epochs, learning.rate=5){
  X = cbind(1, X)
  n = dim(X)[1]
  k = dim(X)[2]
  β.hat = rep(1, k)
  for(i in 1:epochs){
    # Calculate the error: yfit - y, yfit = sigmoid(Xβ)
    residuals = sigmoid(X %*% β.hat) - y
    #residuals = X %*% β.hat - Y # linear regression

    # Calculate the gradient at that point
    delta = (t(X) %*% residuals) / n

    # Move β.hat in opposite direction of gradient, to find local
error minima
    β.hat = β.hat - learning.rate * delta
  }
  rownames(β.hat)[1] = "Intercept"
  return(β.hat)
}

## Returns predictions of given logistical regresion
pred = function(β.hat, X){
  X = cbind(1, X)
  return(X %*% β.hat)
}

### Implementation
# Plot sigmoid curve
curve(sigmoid, -10, 10)

# Logistical regression
β.logreg = logistic_reg(X.iris, y.iris, 300000, 5)
```

```
yfit = pred(β.logreg, X.iris) > 0

# Plot original data and decision boundry
plot(X.iris[,1], X.iris[,2], col=rgb(y.iris, 1-y.iris, 0, 0.5),
pch=19,
     xlab="Sepal.Length", ylab="Petal.Length")
#points(X.iris[,1], X.iris[,2], col=rgb(yfit, 1-yfit, 0, 0.5), pch=19)
# optional to show predictions

# Calculate boundry, add to plot through abline
intercept = -β.logreg[1]/β.logreg[3]
slope = -β.logreg[2]/β.logreg[3]
abline(intercept, slope, col=1)
```