Logistic Regression from Scratch

Karen Mazidi

Notebook for the Logistic Regression Chapter

First, load the package and plasma data set.

```
library(HSAUR)
```

```
## Loading required package: tools
```

```
data(plasma)
```

Logistic Regression using R

Use R's glm() function first as our ground truth.

```
set.seed(1234)
i <- sample(1:nrow(plasma), 0.75*nrow(plasma), replace=FALSE)
train <- plasma[i,]
test <- plasma[-i,]
glm1 <- glm(ESR~fibrinogen, data=train, family=binomial)
probs <- predict(glm1,  newdata=test, type="response")
pred <- ifelse(probs> 0.5, 2, 1)
acc <- mean(pred == as.integer(test$ESR))
summary(glm1)
```

```
##
## Call:
## glm(formula = ESR ~ fibrinogen, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9852  -0.7375  -0.5074  -0.1920   2.2554
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.6141     2.6591  -2.111   0.0347 *
## fibrinogen    1.5084     0.8543   1.766   0.0775 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26.992  on 23  degrees of freedom
## Residual deviance: 22.716  on 22  degrees of freedom
## AIC: 26.716
##
## Number of Fisher Scoring iterations: 4
```

Logistic Regression from Scratch

First we need to define the sigmoid function.

```
# function to return a vector of sigmoid values from an input matrix
sigmoid <- function(z){
  1.0 / (1+exp(-z))
}
# set up weight vector, label vector, and data matrix
weights <- c(1, 1)
data_matrix <- cbind(rep(1, nrow(train)), train$fibrinogen)
labels <- as.integer(train$ESR) - 1
```

Then we need code for gradient descent. The algorithm used here first starts with all weights = 1, then iterates. Notice we get the same weights (coefficients) as R gave us, but it took a lot longer.

```
weights <- c(1, 1)  # repeat this for rerunning the block
learning_rate <- 0.001
for (i in 1:500000){
  prob_vector <- sigmoid(data_matrix %*% weights)
  error <- labels - prob_vector
  weights <- weights + learning_rate * t(data_matrix) %*% error
}
weights
```

```
##           [,1]
## [1,] -5.614088
## [2,]  1.508385
```

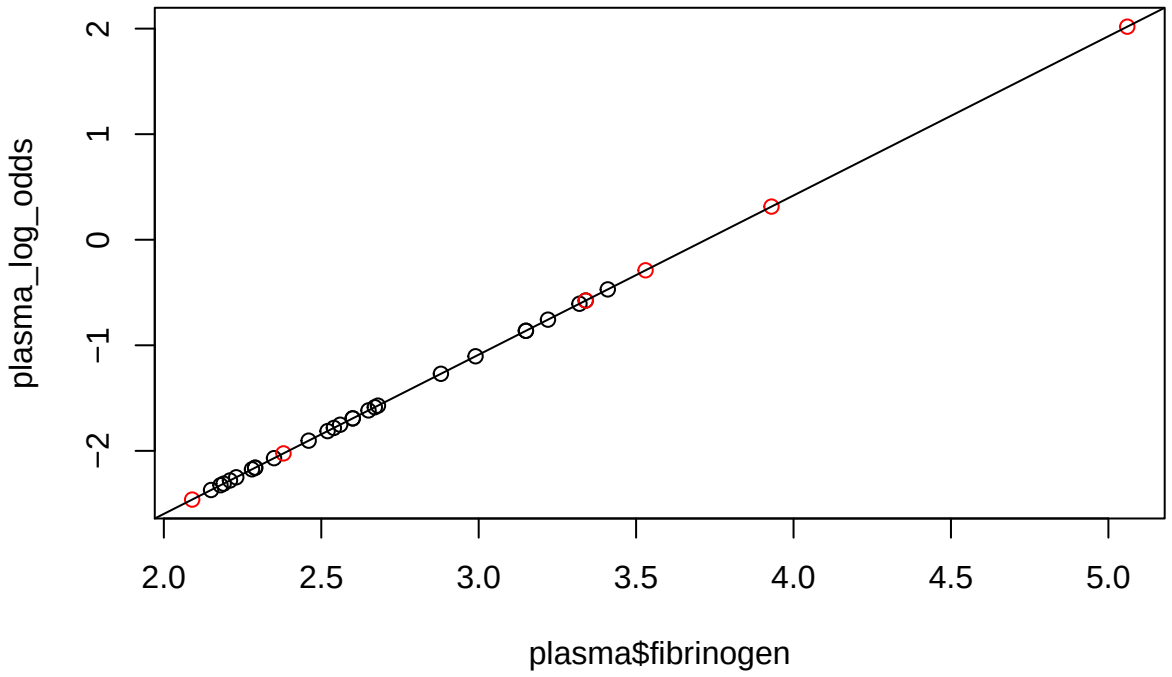
Predict with the weights we generated

```
# predict with our weights
test_matrix <- cbind(rep(1, nrow(test)), test$fibrinogen)
test_labels <- as.integer(test$ESR) - 1
predicted <- test_matrix %*% weights
probabilities <- exp(predicted) / (1 + exp(predicted))
predictions <- ifelse(probabilities > 0.5, 1, 0)
mean(predictions == test_labels)
```

```
## [1] 1
```

Visualization that the log odds is a line.

```
plasma_log_odds <- cbind(rep(1, 32), plasma$fibrinogen) %*% weights
plot(plasma$fibrinogen, plasma_log_odds, col=plasma$ESR)
abline(weights[1], weights[2])
```


plasma_log_odds
2
1
0
−1
−2
2.0
2.5
3.0
3.5
4.0
4.5
5.0
plasma$fibrinogen