CS 498 AML HW2 Report

Author: Moyu Liu

Dataset: https://www.kaggle.com/c/aml-hw2/data
Language: R-Studio

Package: caret

Screenshot of your leaderboard accuracy and mention your best test dataset accuracy obtained on kaggle.

52	new	**PengyuCheng**	0.80610	38	1h
53	new	**Lynn**	0.80610	8	14h
54	new	**Moyu Liu**	0.80610	19	now

Your Best Entry
You advanced 5 places on the leaderboard!
Your submission scored 0.80610, which is an improvement of your previous score of 0.80548. Great job! **Tweet this!**

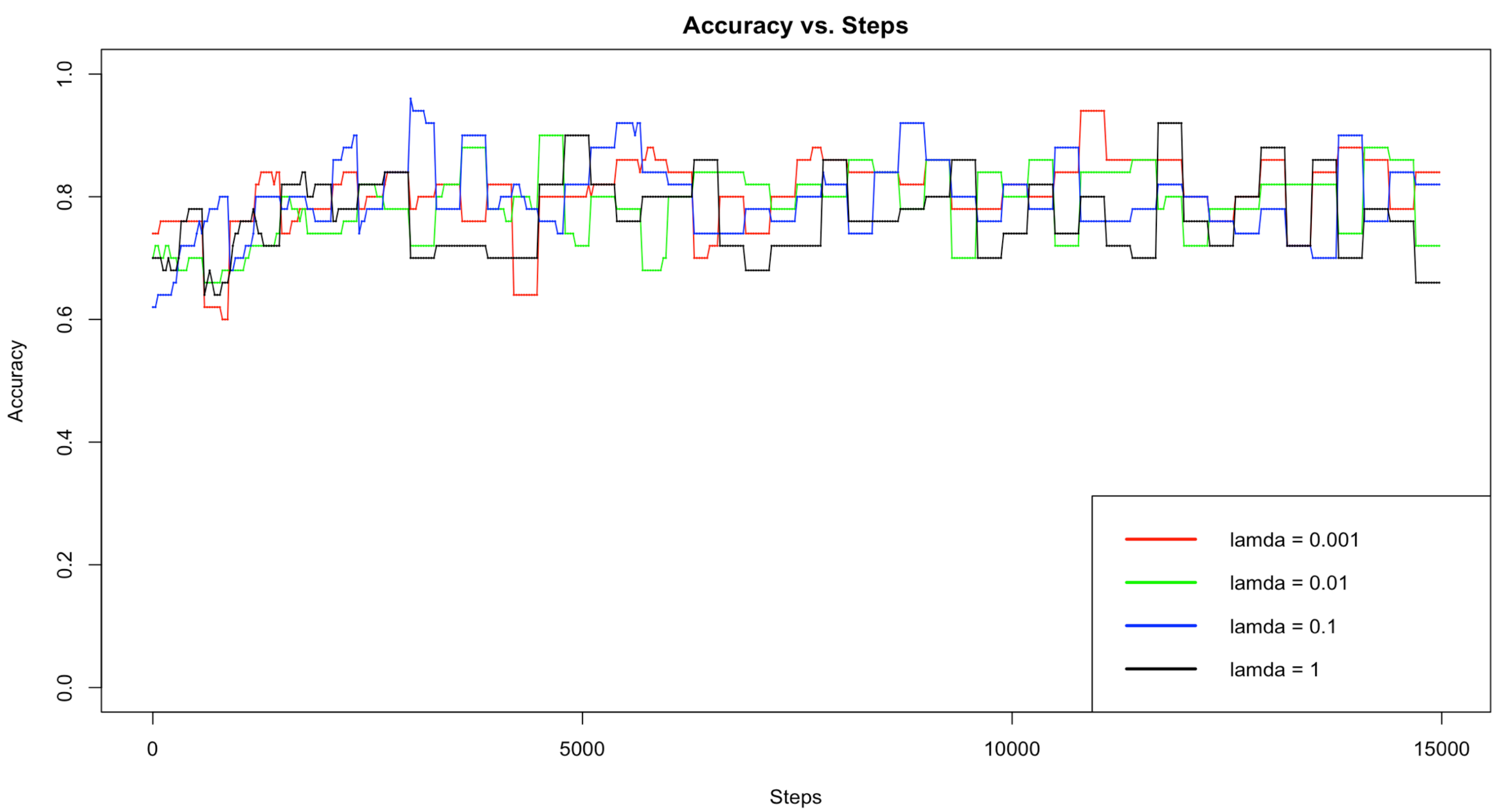
55	new	**Savya Saachi Verma**	0.80589	3	2d
56	new	**Allen Tang**	0.80589	1	19h
57	new	**Chen Pan**	0.80589	2	14h

labels_test.data 10 minutes ago by Moyu Liu add submission details	**0.80241**	☐
labels_test.data 27 minutes ago by Moyu Liu add submission details	**0.80610**	☑
labels_test.data a day ago by Moyu Liu add submission details	**0.80323**	☑

Accuracy: 0.80610

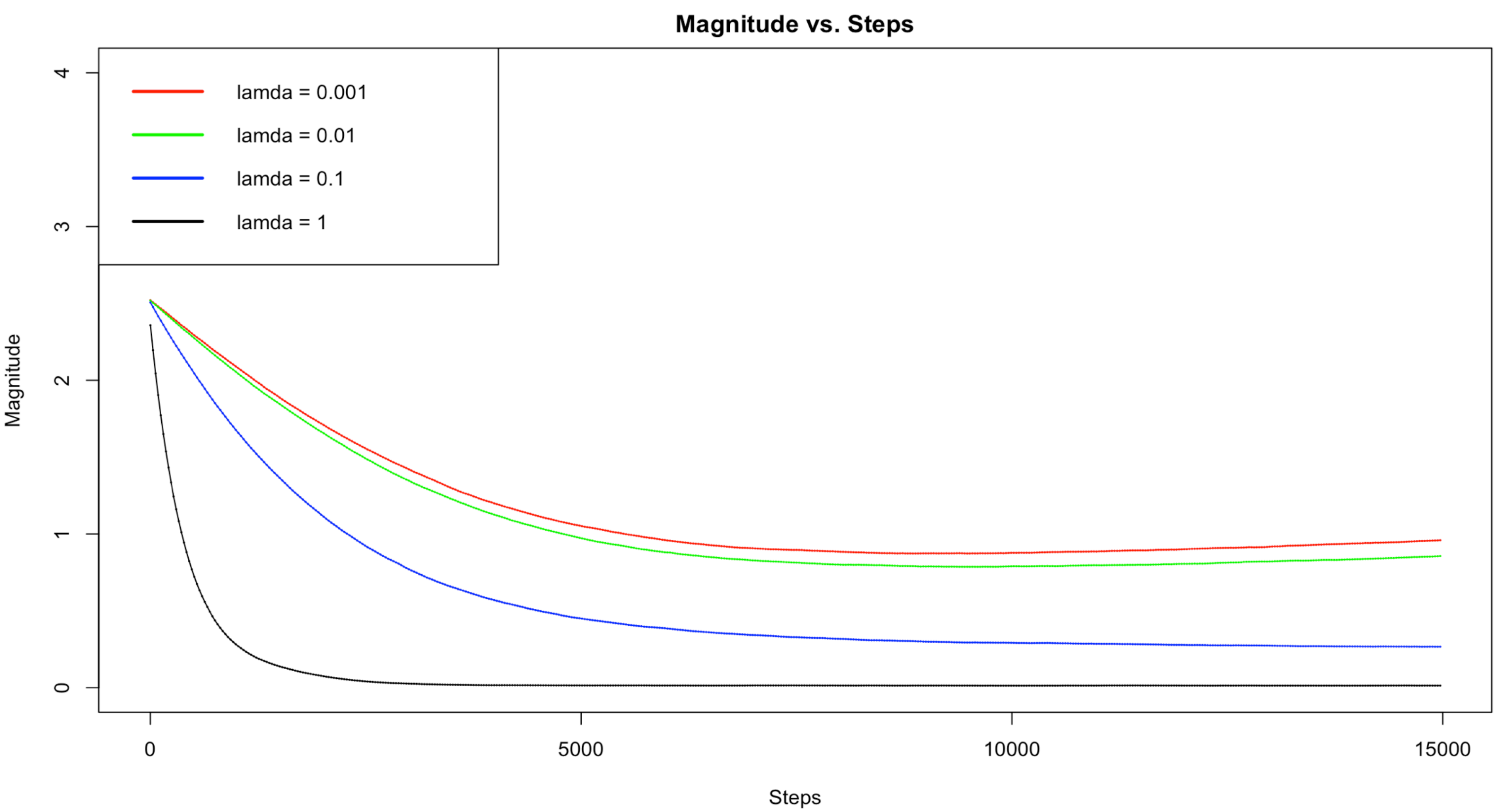
Screenshot: A plot of the accuracy every 30 steps, for each value of the regularization constant. You should plot the curves for all regularization constants in the same plot using different colors with a label showing the corresponding values

Batch Accuracy vs. Steps



Screenshot: A plot of the magnitude of the coefficient vector every 30 steps, for each value of the regularization constant. You should plot the curves for all regularization constants in the same plot using different colors with a label showing the corresponding values.

Magnitude vs. Steps



Your estimate of the best value of the regularization constant, together with a brief description of why you believe that is a good value. What was your choice for the learning rate and why did you choose it ?

Estimate of the best value of the regularization constant.

Best accuracy for each lambda and the best a, b.

```
[1] "0.001 :  0.808873720136519"
[1] "0.01 :  0.806825938566553"
[1] "0.1 :  0.786348122866894"
[1] "1 :  0.771786120591581"
[1] "Best accuracy:  0.808873720136519"
[1] "Best lambda:  0.001"
[1] "Best a:  0.270018298029943"  "Best a:  0.0377664388763483" "Best a:  0.419256351711064"  "Best a:  0.774827405872232"
[5] "Best a:  0.217890432811149"  "Best a:  0.247702093185014"
[1] "Best b:  -1.05078799476634"
```

I select learning rate to be 1/(0.01*s + 1000). By testing the magnitude and the accuracy of different lambda, this learning rate provides the best accuracy without much fluctuating. Also, the magnitude provides the correct trend with a decreasing in magnitude at first, and increasing later until it turns smoother later of the steps. Since it provides the best performance and correct trend after all, I use this as my learning rate.

Screenshot for Code

Calculate Accuracy:

````r
calculate_acc <- function(plot_feature, plot_label, a, b) {
  correct = 0
  for (i in 1:nrow(plot_feature)) {
    r = sign((t(a) %*% as.numeric(plot_feature[i,])) + b)

    if (r == plot_label[i]) {
      correct = correct + 1
    }
  }
  return(correct / nrow(plot_feature))
}
```
````

````r
# Load data
```{r}
library(caret)
train_data = read.csv("dataset/train.data", header=FALSE)
test_data  = read.csv("dataset/test.data", header=FALSE)
```
````

````r
```{r}

training_label = as.numeric(train_data[,15])
training_label[training_label == 1] = -1
training_label[training_label == 2] = 1
col_Names = c("V1","V3","V5","V11","V12","V13")
training_feature = train_data[col_Names]

testing_feature = test_data[col_Names]

head(training_feature)
head(testing_feature)
```
````

Write to CSV:

````r
#te_label = sign(t(best_a) %*% t(data.matrix(te_feature))+best_b)
pre = data.matrix(te_feature)
for(i in 1:ncol(pre)){
  pre[,i] = (pre[,i] - mean(pre[,i])) / sd(pre[,i])
}
te_label = sign(t(best_a) %*% t(pre) + best_b)
te_label[te_label == 1] = ">50K"
te_label[te_label == -1] = "<=50K"

final_output = matrix(0:4883, 4884, 1)

for(i in 1:4884) {
  final_output[i] = paste0("'", final_output[i], "'")
}

final_output = cbind(final_output, t(te_label))
colnames(final_output) = c("Example", "Label")

write.csv(final_output, file = "labels_test.data", row.names = FALSE)

```
````

```r
#90% training and 10% validation data

#Initialize data sections
tr_idx = createDataPartition(y = training_label, p = 0.9, list = FALSE)
tr_feature = training_feature[tr_idx,]
tr_label = training_label[tr_idx]

val_feature = training_feature[-tr_idx,]
val_label = training_label[-tr_idx]

te_feature = testing_feature

for(i in 1:ncol(tr_feature)){
  m = mean(tr_feature[,i])
  sd = sd(tr_feature[,i])
  tr_feature[,i] = (tr_feature[,i] - m) / sd
  val_feature[,i] = (val_feature[,i] - m) / sd
  te_feature[,i] = (te_feature[,i] - m) / sd
}

#initialize variables
four_lambda <- c(0.001, 0.01, 0.1, 1)
epochs <- 50
steps <- 300
batch_size =160
m = 1
n = 1000

def_a = runif(6)
def_b = 0
batch_acc = matrix(0, epochs*(steps/30), 5)
a_acc = matrix(0, epochs*(steps/30), 5)
colnames(batch_acc) = c("Step","0.001","0.01","0.1","1")
colnames(a_acc) = c("Step","0.001","0.01","0.1","1")
```

Calculate a, b, and all batch accuracy

```r
for (epoch_entry in 1:epochs) {
  #step length
  step_len = m/(0.01*epoch_entry+n)

  #50 examples from training set to plot
  plot_idx = sample(seq(1,nrow(val_feature)), size = 50)
  plot_feature = val_feature[plot_idx,]
  plot_label = val_label[plot_idx]

  for (step in 1:steps) {
    batch_idx = sample(seq(1,nrow(tr_feature)), size = 160)
    batch_feature = data.matrix(tr_feature[batch_idx,])
    batch_label = tr_label[batch_idx]

    sum1 = rep(0,6)
    sum2 = 0
    for(i in 1:nrow(batch_feature)) {
      r = (t(a) %*% batch_feature[i,] + b) * batch_label[i]
      if (r >= 1) {
        # a = a - (a*step_len*lambda)
        sum1 = sum1 + (a*step_len*lambda)
      }
      else {
        #a = a - (step_len * ((lambda * a) - batch_feature[i,]*batch_label[i]))
        #b = b + (step_len * batch_label[i])
        sum1 = sum1 + (step_len * ((lambda * a) - batch_feature[i,]*batch_label[i]))
if (step %% 30 == 0) {
  total_step = ((epoch_entry - 1) * steps) + step
  batch_acc[total_step/30, "Step"] = total_step
  batch_acc[total_step/30, lambda_entry+1] = calculate_acc(plot_feature, plot_label, a, b)
  a_acc[total_step/30, "Step"] = total_step
  a_acc[total_step/30, lambda_entry+1] = t(a) %*% a
}
```

Plot data:

```r
plot(seq(1,15000,30),batch_acc[,2],type = "o", cex = 0.1, col = "red", ylim = c(0,1), xlab = "Steps", ylab = "Accuracy", main = "Accuracy vs. Steps")
lines(seq(1,15000,30), batch_acc[,3],type = "o", cex = 0.1, col = "green")
lines(seq(1,15000,30), batch_acc[,4],type = "o", cex = 0.1, col = "blue")
lines(seq(1,15000,30), batch_acc[,5],type = "o", cex = 0.1, col = "black")
legend("bottomright", legend=c('lamda = 0.001','lamda = 0.01','lamda = 0.1','lamda = 1'),
       lty=c(1,1,1,1),
       lwd=c(2.5,2.5,2.5,2.5),
       col=c("red","green","blue","black")
```