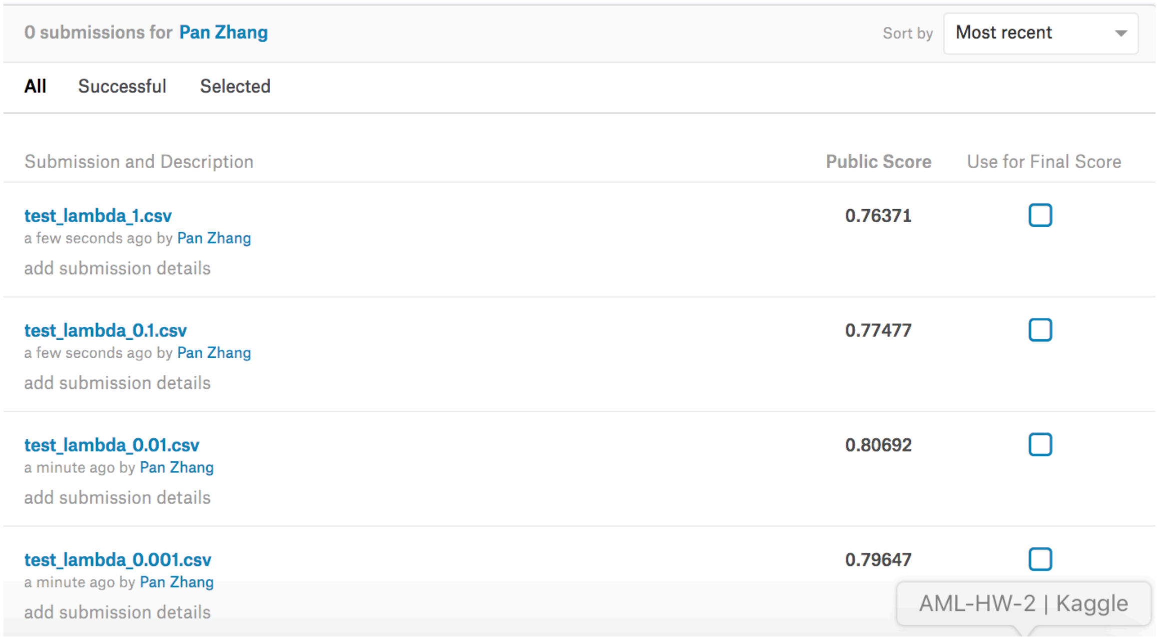
CS498AML_HW2_PanZhang

A screenshot of your leaderboard accuracy:



0 submissions for Pan Zhang

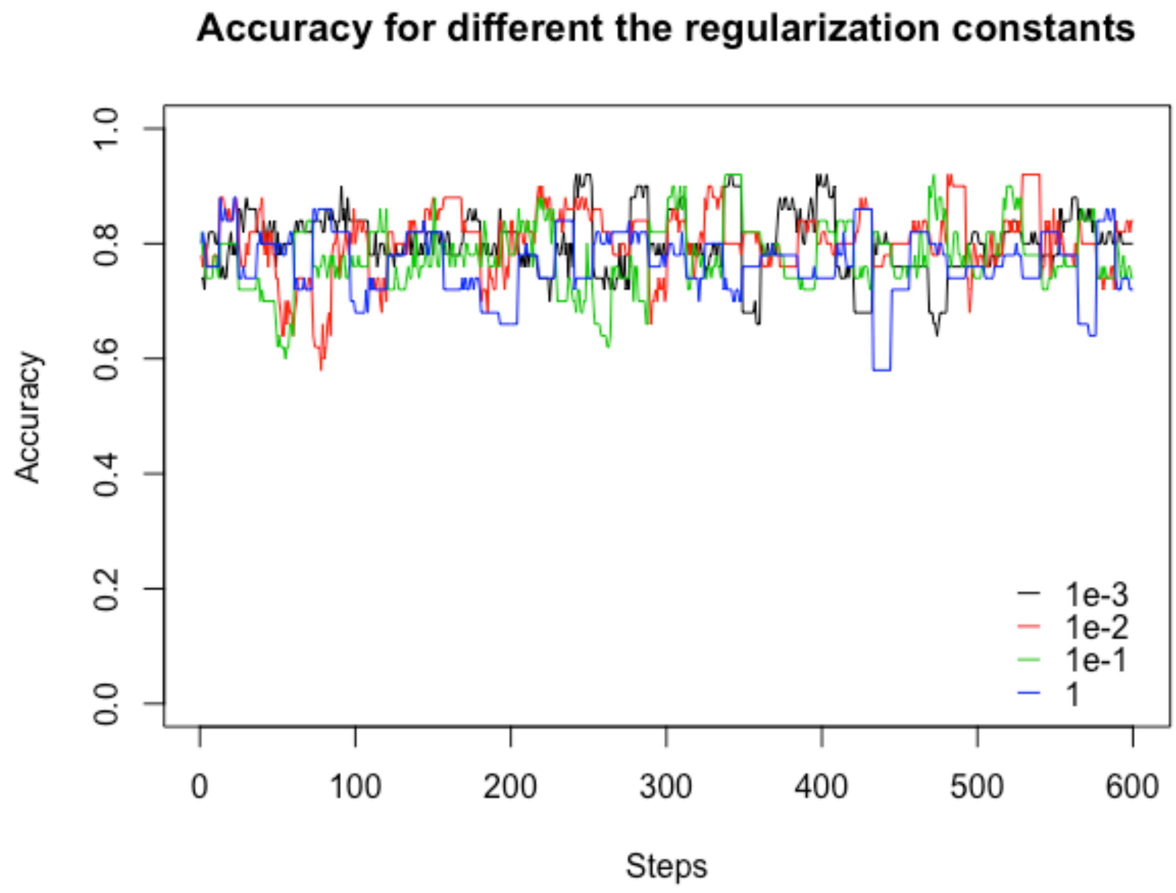
Sort by Most recent

All Successful Selected

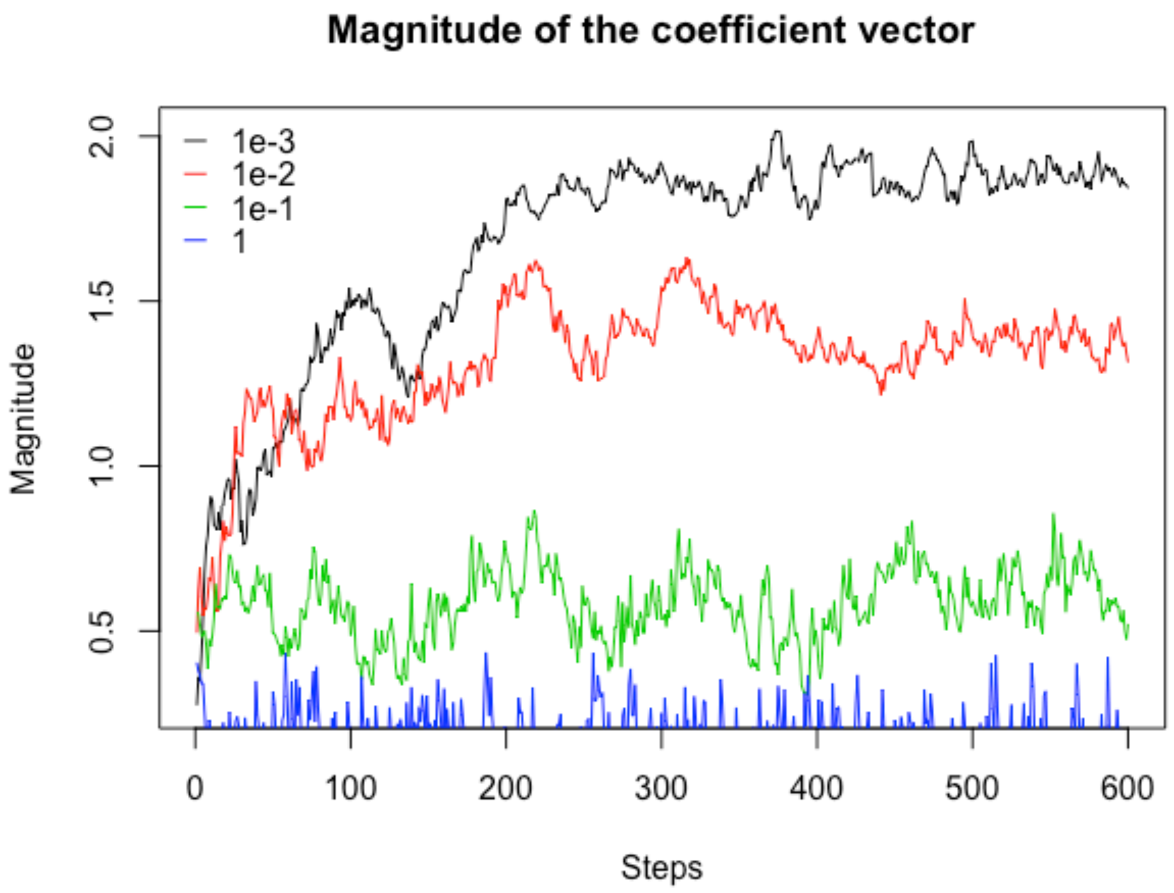
Submission and Description	Public Score	Use for Final Score
test_lambda_1.csv a few seconds ago by Pan Zhang add submission details	0.76371	☐
test_lambda_0.1.csv a few seconds ago by Pan Zhang add submission details	0.77477	☐
test_lambda_0.01.csv a minute ago by Pan Zhang add submission details	0.80692	☐
test_lambda_0.001.csv a minute ago by Pan Zhang add submission details	0.79647	☐

My best test dataset accuracy is 0.80692 when lambda=0.01

Plot of the accuracy:



Plot of the magnitude of the coefficient vector:



Best value of the regularization constant: 0.01
Reasons:

1. **When lambda is 0.01, the accuracy of both validation and test data is highest among 4 different regularization constants.** And the accuracy of lambda= 0.001 and 0.01 is very close and this four regularization constant don't influence accuracy a lot, so even if I still try the value between 0.01 and 0.001, the accuracy should be very close to 0.8.
2. **And it will not lead the λ*a too big or too small, and the accuracy and the magnitude of the coefficient vector become stable quickly.**

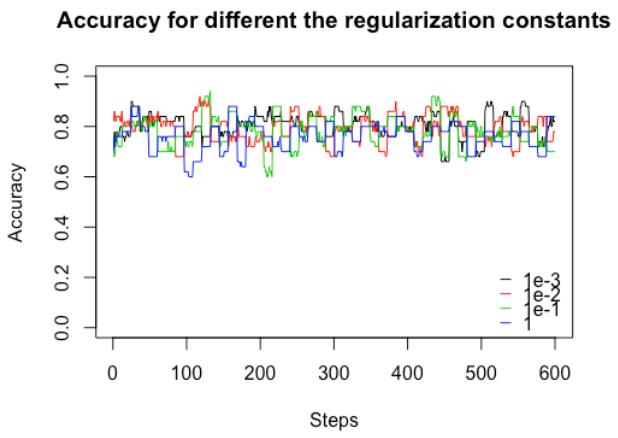
lambda	0.001	0.01	0.1	1
Val	0.811	0.817	0.775	0.761
Test	0.796	0.807	0.775	0.764

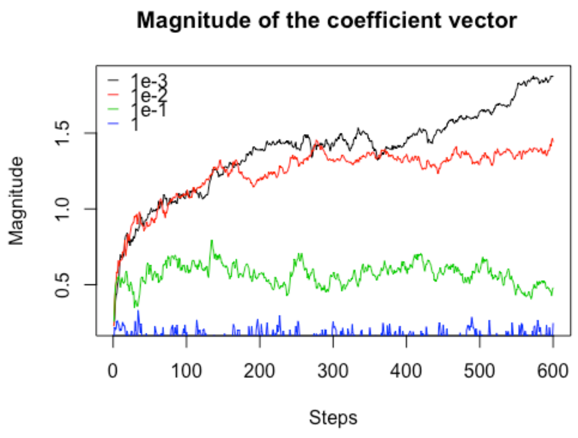
My learning rate: 1/(0.01*e+50), here e is the epoch, the start learning rate is about 0.02.

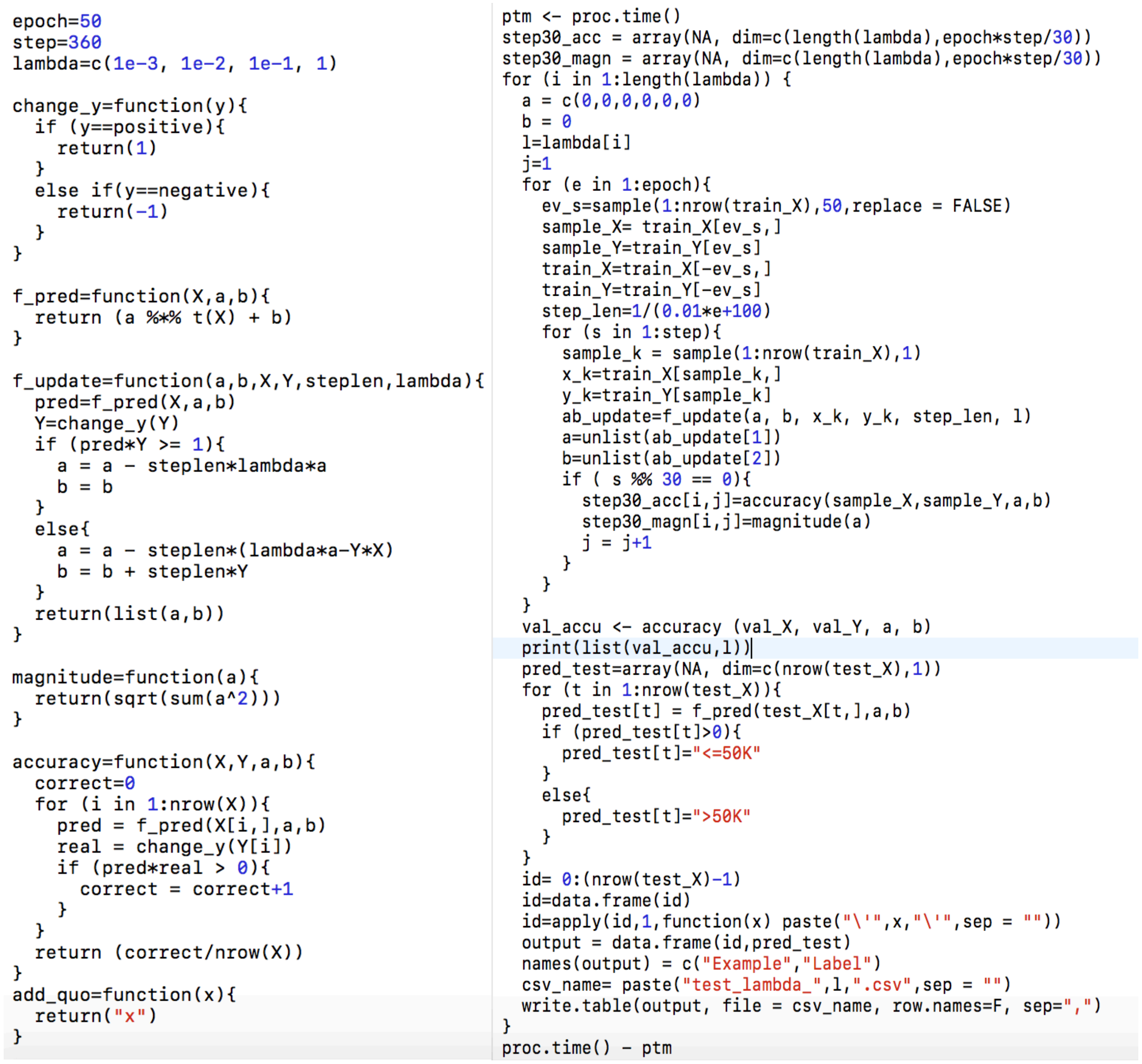
If the learning rate is low, then training is more reliable, but optimization will take a lot of time because steps towards the minimum of the loss function are tiny.
If the learning rate is high, then training may not converge or even diverge. Weight changes can be so big that the optimizer overshoots the minimum and makes the loss worse.

For my learning rate, it is about 0.02. And we can see the magnitude of the coefficient vector of 4 different regularization constants become stable quickly.

When learning rate become 1/(0.01*e+100), which is smaller. Follow figure shows magnitude is still go up when lambda is 0.001 with 50 epoch * 360 step, which means more steps and time is needed to find the a. And the accuracy doesn't improve. Thus, my learning rate 1/(0.01*e+50) is better.







```r
epoch=50
step=360
lambda=c(1e-3, 1e-2, 1e-1, 1)

change_y=function(y){
  if (y==positive){
    return(1)
  }
  else if(y==negative){
    return(-1)
  }
}

f_pred=function(X,a,b){
  return (a %*% t(X) + b)
}

f_update=function(a,b,X,Y,steplen,lambda){
  pred=f_pred(X,a,b)
  Y=change_y(Y)
  if (pred*Y >= 1){
    a = a - steplen*lambda*a
    b = b
  }
  else{
    a = a - steplen*(lambda*a-Y*X)
    b = b + steplen*Y
  }
  return(list(a,b))
}

magnitude=function(a){
  return(sqrt(sum(a^2)))
}

accuracy=function(X,Y,a,b){
  correct=0
  for (i in 1:nrow(X)){
    pred = f_pred(X[i,],a,b)
    real = change_y(Y[i])
    if (pred*real > 0){
      correct = correct+1
    }
  }
  return (correct/nrow(X))
}
add_quo=function(x){
  return("x")
}
```

```r
ptm <- proc.time()
step30_acc = array(NA, dim=c(length(lambda),epoch*step/30))
step30_magn = array(NA, dim=c(length(lambda),epoch*step/30))
for (i in 1:length(lambda)) {
  a = c(0,0,0,0,0,0)
  b = 0
  l=lambda[i]
  j=1
  for (e in 1:epoch){
    ev_s=sample(1:nrow(train_X),50,replace = FALSE)
    sample_X= train_X[ev_s,]
    sample_Y=train_Y[ev_s]
    train_X=train_X[-ev_s,]
    train_Y=train_Y[-ev_s]
    step_len=1/(0.01*e+100)
    for (s in 1:step){
      sample_k = sample(1:nrow(train_X),1)
      x_k=train_X[sample_k,]
      y_k=train_Y[sample_k]
      ab_update=f_update(a, b, x_k, y_k, step_len, l)
      a=unlist(ab_update[1])
      b=unlist(ab_update[2])
      if ( s %% 30 == 0){
        step30_acc[i,j]=accuracy(sample_X,sample_Y,a,b)
        step30_magn[i,j]=magnitude(a)
        j = j+1
      }
    }
  }
  val_accu <- accuracy (val_X, val_Y, a, b)
  print(list(val_accu,l))
  pred_test=array(NA, dim=c(nrow(test_X),1))
  for (t in 1:nrow(test_X)){
    pred_test[t] = f_pred(test_X[t,],a,b)
    if (pred_test[t]>0){
      pred_test[t]="<=50K"
    }
    else{
      pred_test[t]=">50K"
    }
  }
  id= 0:(nrow(test_X)-1)
  id=data.frame(id)
  id=apply(id,1,function(x) paste("\'",x,"\'",sep = ""))
  output = data.frame(id,pred_test)
  names(output) = c("Example","Label")
  csv_name= paste("test_lambda_",l,".csv",sep = "")
  write.table(output, file = csv_name, row.names=F, sep=",")
}
proc.time() - ptm
```