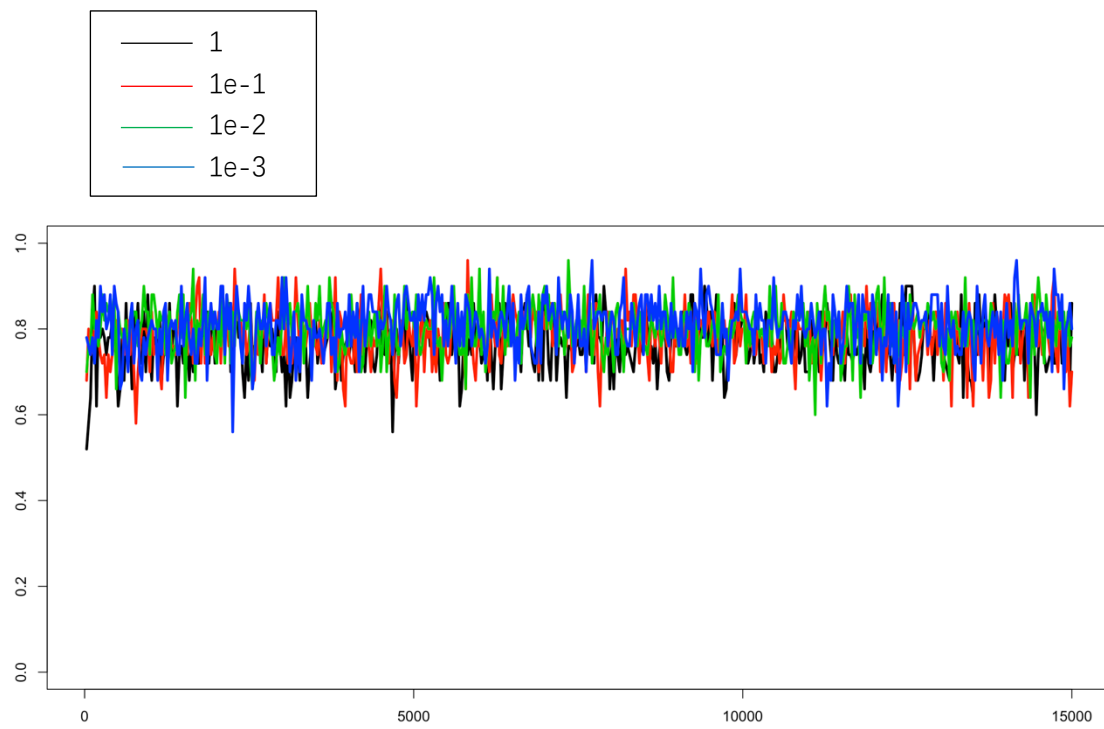
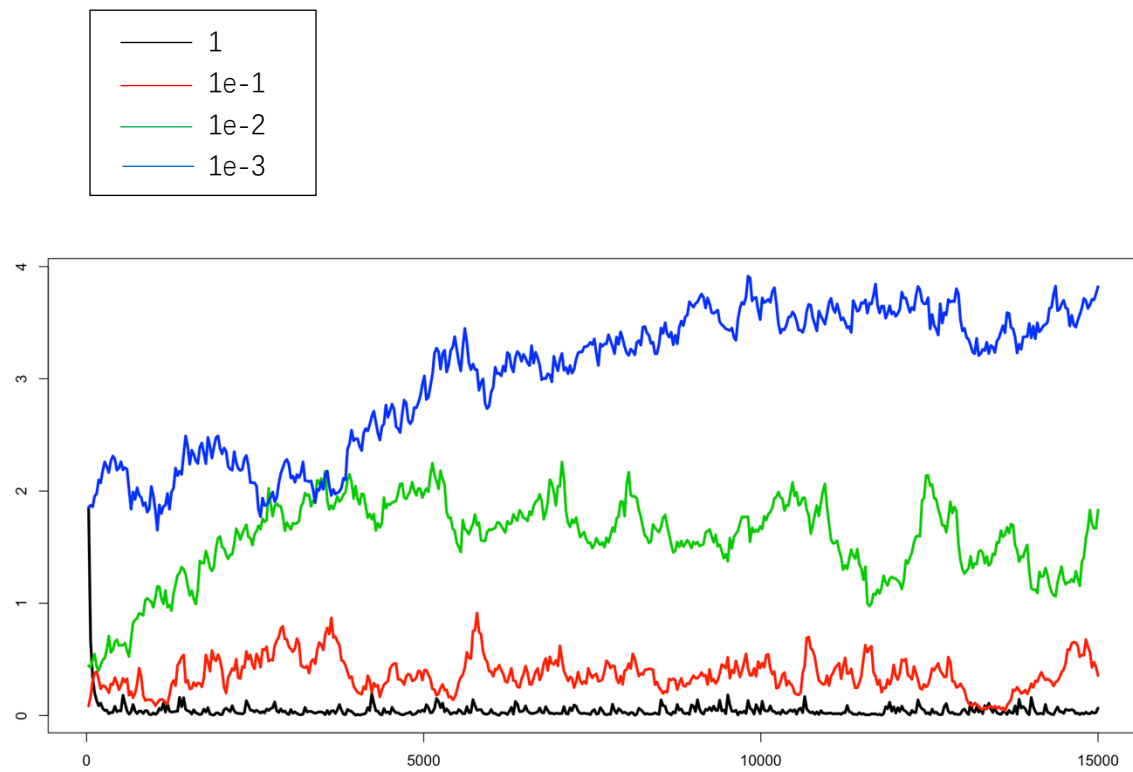


https://www.kaggle.com/c/aml-hw2/leaderboard							
	Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team
					My Submissions	Submit Predictions	
22	new	fancy doodle		0.80773	4	4h	
23	new	Tianyu Sun		0.80733	16	1d	
24	new	Andrew Jiang		0.80733	30	19h	
25	new	Kewei		0.80733	7	7h	
26	new	gchen46		0.80733	4	8h	
27	new	Mengdi Gao		0.80733	8	3h	
28	new	Anonymous		0.80733	7	3h	
29	new	Hengzhe Ding		0.80733	13	now	
Your Best Entry ↑ You advanced 8 places on the leaderboard! Your submission scored 0.80733, which is an improvement of your previous score of 0.80589. Great job! Tweet this!							
30	new	test_account		0.80692	16	2d	
31	new	Pan Zhang		0.80692	5	7h	
32	new	Shunuo Lai		0.80692	22	6h	

My best test dataset accuracy obtained on Kaggle is 80.733%.



The y axis is the accuracy and x axis is overall step count number.



The y axis is the magnitude of vector a and x axis is overall step count number.

The best value of the regularization constant is $1e-3$. The reason why I think it is a good value is that the accuracy result is the best with this constant. In addition, little regularization constant means hyperplane with little margin between the two class, "-1" and "1", which is good in this case, because the feature vector of two class is close to each other.

My choice of learning rate is $1/(0.01s+50)$.

Because in the equation of learning rate, $m/(s+n)$, m should be small and n should be large.

Library used & normalizing data:

```
1 library(caret)
2 # Take in data.
3 setwd("/Users/hengzhe/Library/Mobile Documents/com~apple~CloudDocs/UIUC/CS-498-Applied")
4 train_data = read.table("train.csv", sep=",")
5 test_data = read.table("test.csv", sep=",")
6
7 # Function that normalize data.
8 translate_data <- function(input_data){
9   for (i in c(1,3,5,11:13)){
10     input_data[[sprintf("V%d", i)]] <- scale(input_data[[sprintf("V%d", i)]])
11   }
12   return (input_data)
13 }
14
```

SVM training:

```
40 # Randomly split training data to 2 parts, 90% training part and 10% validation part.
41 ind <- createDataPartition(train_data$V15, times=1, p=0.9, list=F)
42 val_train <- train_data[ind,]
43 val_test <- train_data[-ind,]
44
45 # start loop: this is a three layer loop, the first layer is to choose which regularization constant has the highest
46 # accuracy rate, the second layer is season/epoch and the last layer is step.
47 for (m in 1:length(lambda_list)){
48   lambda <- lambda_list[m]
49   for (i in 1:Ns){
50     # Setting the learning rate.
51     g_s <- 1/(0.01*i*50)
52     for (j in 1:step){
53       index <- sample(1:nrow(val_train), 1)
54       r <- val_train[index,]
55       if (as.numeric(levels(r$V15)[r$V15])*(sum(a*r[c(1,3,5,11:13)]))+b >= 1){
56         a <- a - g_s*lambda*a
57         b <- b
58       } else {
59         a <- a - g_s*(lambda*a - as.numeric(levels(r$V15)[r$V15])*r[c(1,3,5,11:13)])
60         b <- b + g_s * as.numeric(levels(r$V15)[r$V15])
61       }
62       if (j %% 30 == 0){
63         val_sample <- val_test[sample(1:nrow(val_test), 50),]
64         pred_sample <- c()
65         for (q in 1:nrow(val_sample)){
66           if(sum(a*val_sample[q,][c(1,3,5,11:13)])+b >= 0)
67             pred_sample[q] <- 1
68           else
69             pred_sample[q] <- -1
70         }
71         pred_sample_table <- table(actual=val_sample$V15, predict=pred_sample)
72         acc <- sum(diag(pred_sample_table))/sum(pred_sample_table)
73         x[countxy] <- (i-1)*step + j
74         y[countxy] <- acc
75         mag_a[countxy] <- sum(a*a)
76         countxy <- countxy + 1
77       }
78     }
79   }
80
81 # Calculate accuracy in different regularization constant and
82 # choose the regularization constant with the highest accuracy.
83 pred <- c()
84 for (k in 1:nrow(val_test)){
85   if (sum(a*val_test[k,][c(1,3,5,11:13)])+b >= 0)
86     pred[k] <- 1
87   else
88     pred[k] <- -1
89 }
90 pred_table <- table(actual=val_test$V15, predict=pred)
91 print(sum(diag(pred_table))/sum(pred_table))
92 if (sum(diag(pred_table))/sum(pred_table) > accuracy){
93   accuracy <- sum(diag(pred_table))/sum(pred_table)
94   final_a <- a
95   final_b <- b
96   final_lambda <- lambda
97 }
98 }
99
```