SMM implementation 2

Chanwoo Lee, April 17, 2020

1 SMM implementation

1.1 Simulation 1

I generate random matrix $X_1, \ldots, X_{100} \in \mathbb{R}^{5\times 4}$ whose entries are from i.i.d.Unif$(-1, 1)$. I set ground truth matrix $B = UV^T \in \mathbb{R}^{5\times 4}$ such that $U \in \mathbb{R}^{5\times 3}, V \in \mathbb{R}^{4\times 3}$ whose entries are from i.i.d. Unif$(-1, 1)$. Our classification rule is

$$y = \text{sign}(\langle B, X\rangle + 0.1).$$

First, I perform five folded cross validation with the new algorithm (multiple initialization method). Table 1 shows the miss classification rate (MCR) of SVM method and SMM method at each test. Table 2 shows the loss function value at the last iteration at each test.

	1st	2nd	3rd	4th	5th	average
SVM	0.90	0.95	0.9	0.75	0.9	0.88
SMM	0.95	0.95	1.0	0.75	0.9	0.91

Table 1: Miss classification rate in Simulation 1

	1st	2nd	3rd	4th	5th	average
SVM	21.05690	21.48122	20.0643	13.47480	11.44244	17.50393
SMM	23.06968	22.65985	27.2126	13.93015	13.09969	19.99439

Table 2: Loss function value in Simulation 1

Secondly, I use PCA method to visualize matrix in 2 dimension. I vectorize feature matrix X_i and make the design matrix,

$$X = (\text{Vec}(X_1)^T, \cdots, \text{Vec}(X_N)^T)^T \in \mathbb{R}^{N\times mn}.$$

Through PCA method, I plot matrices $(X_1, \ldots, X_N)$ in terms of PC1 vs PC2 and PC1 vs PC3. In addition, I make grids as $a \cdot \text{PC1} + b \cdot \text{PC2}$ where a and b are some constant between minimum and maximum value of corresponding principal component of the data. I compare the true, SVM, and SMM boundary in the plots. Figure 1 shows the result.

1.2 Simulation 2

I generate random matrix and true label $(X_1, y_1), \ldots, (X_{100}, y_{100})\mathbb{R}^{5\times 4}$ with the following rule

$$\{(X_i, 1) : X_i = U_1V_1^T + E_i \quad i = 1, \ldots, 100\} \text{ and } \{(X_j, -1) : X_j = U_2V_2^T + E_i \quad j = 101, \ldots, 200\},$$

where $U_1, U_2 \in \mathbb{R}^{5\times 3}, V_1, V_2 \in \mathbb{R}^{4\times 3}$ whose entries are from i.i.d unif$(-1, 1)$, and E_i are Gaussian random matrix whose entries are from i.i.d $N(0, 2^2)$. In this case, there is no specific ground

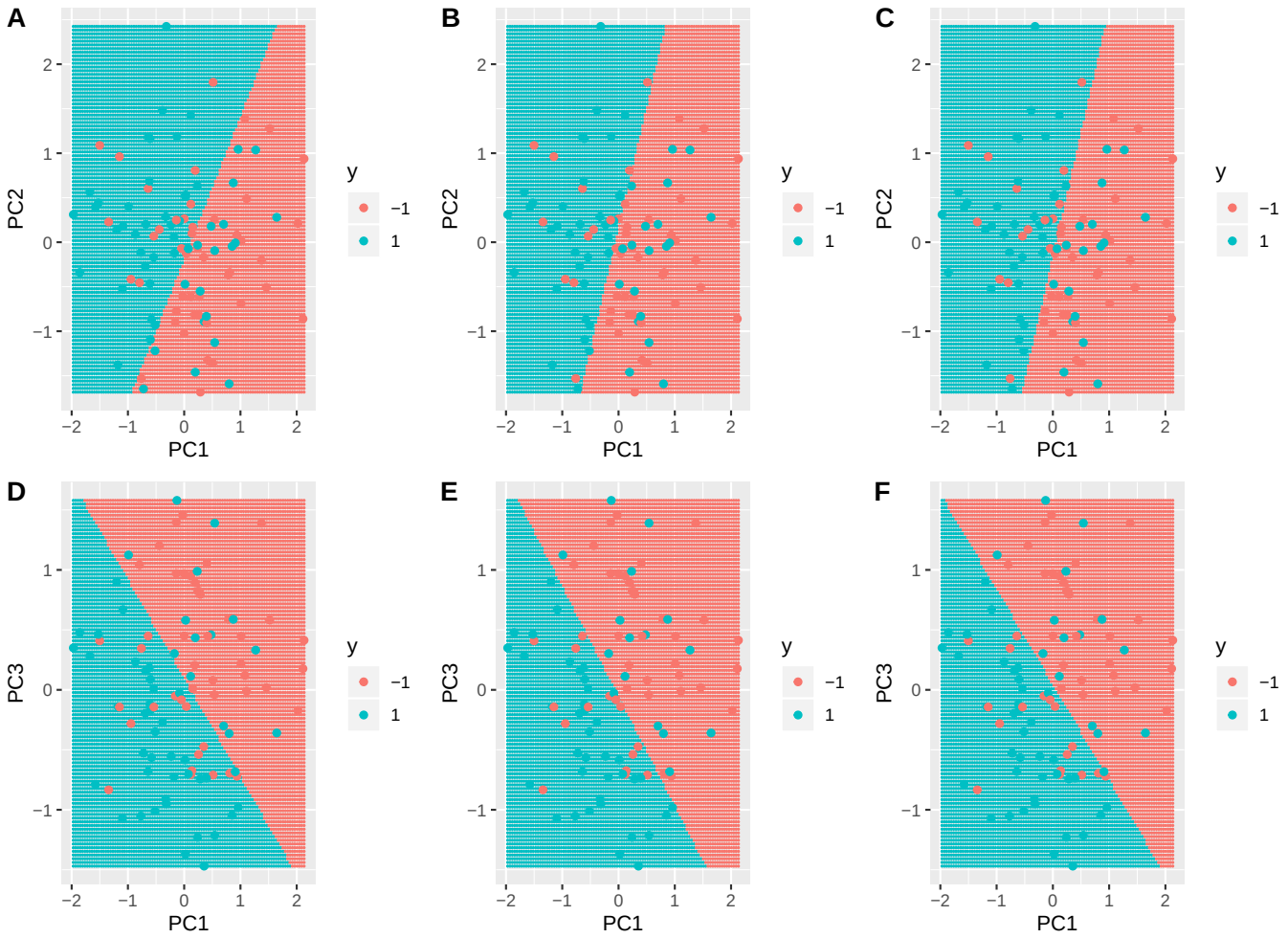

Figure 1: (A,D) are true boundary, (B,E) are SVM boundary, and (C,F) are SMM boundary. (A,B,C) are plotted with PC1 and PC2 axis and (D,E,F) are plotted with PC1 and PC3.

truth rank because this model does not have linear classification rule (It has different distribution according to label). I set rank as three to fit SMM.

I perform five folded cross validation with the new algorithm (multiple initialization method). Table 3 shows the miss classification rate (MCR) of SVM method and SMM method at each test. In SMM method, I set the rank as three. Table 4 shows the loss function value at the last iteration at each test.

	1st	2nd	3rd	4th	5th	average
SVM	0.55	0.65	0.75	0.6	0.75	0.66
SMM	0.50	0.80	0.75	0.7	0.75	0.70

Table 3: Miss classification rate in Simulation 2

Secondly, by the same way in the previous section, I plot matrices with PCA method and compare SVM and SMM classification boundary. Figure 2 shows the result.

	1st	2nd	3rd	4th	5th	average
SVM	194.3430	304.666	233.5439	275.6089	278.5445	257.3412
SMM	201.4427	322.057	233.1719	293.3010	293.5649	268.7075

Table 4: Loss function value in Simulation 2

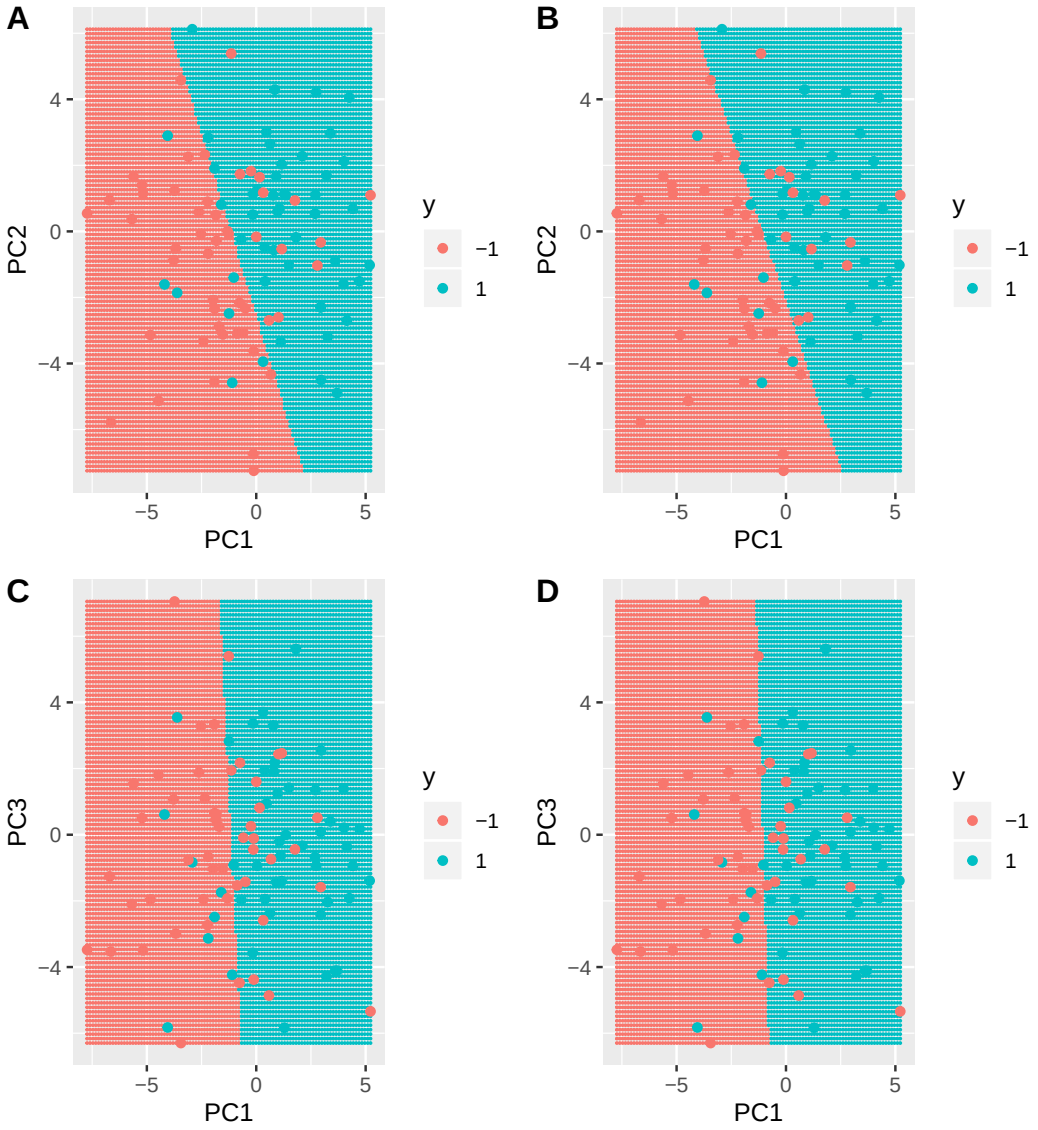

Figure 2: (A,B) are plotted with PC1 and PC2 axis and (C,D) are plotted with PC1 and PC2 axis. (A,C) and (B,D) show SVM and SMM classification boundary respectively

2 Simulation codes

```
source("SMMfunctions.R")

################################################
## CV
################################################
### simulation 1
### data generation
library(kableExtra)
set.seed(1818)
m = 5; n = 4; r = 3; N = 100; b0 = 0.1
result = gendat(m,n,r,N,b0)
X = result$X; y = result$y; dat = result$dat
B = result$B


### CV
ind = sample(100,100)

cvresult = matrix(nrow = 2, ncol = 5)
objresult = matrix(nrow = 2,ncol = 5)
for (i in 1:5) {
  testind = ind[((20*(i-1)+1):(20*i))]

  trX = X[-testind]
  result = svm(trX,y[-testind])
  cvresult[1,i] = length(which(unlist(lapply(X[testind],result$predict))==y[
    testind]))/20
  objresult[1,i] = result$obj
  result = smm(trX,y[-testind],3)
  cvresult[2,i] = length(which(unlist(lapply(X[testind],result$predict))==y[
    testind]))/20
  objresult[2,i] = objv(result$B,result$b0,trX,y[-testind])
  print(paste(i,"-th is done"))
}
cv = cbind(cvresult,apply(cvresult,1,mean))
colnames(cv) = c("1st","2nd","3rd","4th","5th","average")
rownames(cv) = c("SVM","SMM")
kable(cv,"latex")
cvobj = cbind(objresult,apply(objresult,1,mean))
colnames(cvobj) = c("1st","2nd","3rd","4th","5th","average")
rownames(cvobj) = c("SVM","SMM")
kable(cvobj,"latex")



################################################
## pca plotting
################################################

## data generation

result1 = svm(X,y)
result2 = smm(X,y,3)
x = dat[,-1]


```

```r
library(ggplot2)
coord = eigen(cov(x))$vectors
fcoef = t(coord)%*%t(x)
rownames(fcoef) = paste("PC",1:(m*n),sep = "")
ndat = as.data.frame(t(rbind(y,fcoef)))

xgrid1 = seq(min(fcoef[1,]),max(fcoef[1,]),length = 100)
xgrid2 = seq(min(fcoef[2,]),max(fcoef[2,]),length = 100)
xgrid3 = seq(min(fcoef[3,]),max(fcoef[3,]),length = 100)




#### TRUE pc1 vs pc2 and pc1 vs pc3
xgrid12 = expand.grid(X1 = xgrid1,X2 = xgrid2)
xgrid13 = expand.grid(X1 = xgrid1,X2 = xgrid3)


gridX12 = lapply(1:nrow(xgrid12),
                 function(i) matrix(coord[,c(1,2)]%*%t(xgrid12[i,]),nrow = m,ncol =
    n))
gridX13 = lapply(1:nrow(xgrid13),
                   function(i) matrix(coord[,c(1,3)]%*%t(xgrid13[i,]),nrow = m,ncol
    = n))


ygrid12 = unlist(lapply(gridX12,function(x) sign(sum(B*x)+b0)))
ygrid13 = unlist(lapply(gridX13,function(x) sign(sum(B*x)+b0)))

totgrid12 = as.data.frame(cbind(xgrid12,ygrid12))
totgrid13 = as.data.frame(cbind(xgrid13,ygrid13))

library(ggpubr)
gt12 = ggplot(data = totgrid12,aes(x = X1,y = X2,colour = as.factor(ygrid12)))+
    geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC2")+
  geom_point(data =ndat,aes(x = PC1,y = PC2,colour = as.factor(y)))
gt12

gt13 = ggplot(data = totgrid13,aes(x = X1,y = X2,colour = as.factor(ygrid13)))+
    geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC3")+
  geom_point(data =ndat,aes(x = PC1,y = PC3,colour = as.factor(y)))
gt13



## SVM method

Bhat1 = result1$B; b0hat1 = result1$b0
ygrid12 = unlist(lapply(gridX12,function(x) sign(sum(Bhat1*x)+b0hat1)))
ygrid13 = unlist(lapply(gridX13,function(x) sign(sum(Bhat1*x)+b0hat1)))

totgrid12 = as.data.frame(cbind(xgrid12,ygrid12))
totgrid13 = as.data.frame(cbind(xgrid13,ygrid13))

library(ggpubr)
g12 = ggplot(data = totgrid12,aes(x = X1,y = X2,colour = as.factor(ygrid12)))+geom
    _point(size = .1)+
```

```
  labs(colour = "y",x ="PC1",y = "PC2")+
  geom_point(data =ndat,aes(x = PC1,y = PC2,colour = as.factor(y)))
g12

g13 = ggplot(data = totgrid13,aes(x = X1,y = X2,colour = as.factor(ygrid13)))+geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC3")+
  geom_point(data =ndat,aes(x = PC1,y = PC3,colour = as.factor(y)))
g13



#### Linear SMM estimation pc1 vs pc2 and pc1 vs pc3

Bhat2 = result2$B; b0hat2 = result2$b0
ygrid12 = unlist(lapply(gridX12,function(x) sign(sum(Bhat2*x)+b0hat2)))
ygrid13 = unlist(lapply(gridX13,function(x) sign(sum(Bhat2*x)+b0hat2)))

totgrid12 = as.data.frame(cbind(xgrid12,ygrid12))
totgrid13 = as.data.frame(cbind(xgrid13,ygrid13))


gm12 = ggplot(data = totgrid12,aes(x = X1,y = X2,colour = as.factor(ygrid12)))+geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC2")+
  geom_point(data =ndat,aes(x = PC1,y = PC2,colour = as.factor(y)))
gm12

gm13 = ggplot(data = totgrid13,aes(x = X1,y = X2,colour = as.factor(ygrid13)))+geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC3")+
  geom_point(data =ndat,aes(x = PC1,y = PC3,colour = as.factor(y)))
gm13

ggarrange(gt12,g12, gm12,gt13,g13, gm13,
          labels = c("A", "B","C","D","E","F"),
          ncol = 3, nrow = 2)



###############simulation 2
############### new data set data generation

m = 5; n = 4; r = 3
u1 = matrix(runif(m*r,-1,1),nrow = m,ncol = r); v1 = matrix(runif(n*r,-1,1),nrow = n,ncol = r)
u2 = matrix(runif(m*r,-1,1),nrow = m,ncol = r); v2 = matrix(runif(n*r,-1,1),nrow = n,ncol = r)
X = list()
for (i in 1:50) {
  X[[i]] = u1%*%t(v1)+matrix(rnorm(m*n,0,2),nrow = m,ncol = n)
  X[[i+50]] = u2%*%t(v2)+matrix(rnorm(m*n,0,2),nrow = m,ncol = n)
}
y = c(rep(1,50),rep(-1,50))
x = matrix(nrow =100,ncol = m*n)
for(i in 1:100){
  x[i,] = as.vector(X[[i]])
}
dat = data.frame(y = factor(y), x)
```

```
164
165 ############cv############################
166
167 ind = sample(100,100)
168
169 cvresult = matrix(nrow = 2, ncol = 5)
170 objresult = matrix(nrow = 2,ncol = 5)
171 for (i in 1:5) {
172   testind = ind[((20*(i-1)+1):(20*i))]
173
174   trX = X[-testind]
175   result = svm(trX,y[-testind],minimization = TRUE)
176   cvresult[1,i] = length(which(unlist(lapply(X[testind],result$predict))==y[
      testind]))/20
177   objresult[1,i] = result$obj
178   result = smm(trX,y[-testind],r)
179   cvresult[2,i] = length(which(unlist(lapply(X[testind],result$predict))==y[
      testind]))/20
180   objresult[2,i] = objv(result$B,result$b0,trX,y[-testind])
181   print(paste(i,"th is done"))
182 }
183
184 cv = cbind(cvresult,apply(cvresult,1,mean))
185 colnames(cv) = c("1st","2nd","3rd","4th","5th","average")
186 rownames(cv) = c("SVM","SMM")
187 kable(cv,"latex")
188 cvobj = cbind(objresult,apply(objresult,1,mean))
189 colnames(cvobj) = c("1st","2nd","3rd","4th","5th","average")
190 rownames(cvobj) = c("SVM","SMM")
191 kable(cvobj,"latex")
192
193 ############### plotting ####################3
194
195
196 result1 = svm(X,y)
197 result2 = smm(X,y,3)
198 x = dat[,-1]
199
200
201 library(ggplot2)
202 coord = eigen(cov(x))$vectors
203 fcoef = t(coord)%*%t(x)
204 rownames(fcoef) = paste("PC",1:(m*n),sep = "")
205 ndat = as.data.frame(t(rbind(y,fcoef)))
206
207 xgrid1 = seq(min(fcoef[1,]),max(fcoef[1,]),length = 100)
208 xgrid2 = seq(min(fcoef[2,]),max(fcoef[2,]),length = 100)
209 xgrid3 = seq(min(fcoef[3,]),max(fcoef[3,]),length = 100)
210
211
212
213
214 #### TRUE pc1 vs pc2 and pc1 vs pc3
215 xgrid12 = expand.grid(X1 = xgrid1,X2 = xgrid2)
216 xgrid13 = expand.grid(X1 = xgrid1,X2 = xgrid3)
217
218
219 gridX12 = lapply(1:nrow(xgrid12),
220                  function(i) matrix(coord[,c(1,2)]%*%t(xgrid12[i,]),nrow = m,ncol
```

```
    = n))
gridX13 = lapply(1:nrow(xgrid13),
                 function(i) matrix(coord[,c(1,3)]%*%t(xgrid13[i,]),nrow = m,ncol
    = n))


## SVM method

Bhat1 = result1$B; b0hat1 = result1$b0
ygrid12 = unlist(lapply(gridX12,function(x) sign(sum(Bhat1*x)+b0hat1)))
ygrid13 = unlist(lapply(gridX13,function(x) sign(sum(Bhat1*x)+b0hat1)))

totgrid12 = as.data.frame(cbind(xgrid12,ygrid12))
totgrid13 = as.data.frame(cbind(xgrid13,ygrid13))

library(ggpubr)
g12 = ggplot(data = totgrid12,aes(x = X1,y = X2,colour = as.factor(ygrid12)))+geom
    _point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC2")+
  geom_point(data =ndat,aes(x = PC1,y = PC2,colour = as.factor(y)))
g12

g13 = ggplot(data = totgrid13,aes(x = X1,y = X2,colour = as.factor(ygrid13)))+geom
    _point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC3")+
  geom_point(data =ndat,aes(x = PC1,y = PC3,colour = as.factor(y)))
g13



#### Linear SMM estimation pc1 vs pc2 and pc1 vs pc3

Bhat2 = result2$B; b0hat2 = result2$b0
ygrid12 = unlist(lapply(gridX12,function(x) sign(sum(Bhat2*x)+b0hat2)))
ygrid13 = unlist(lapply(gridX13,function(x) sign(sum(Bhat2*x)+b0hat2)))

totgrid12 = as.data.frame(cbind(xgrid12,ygrid12))
totgrid13 = as.data.frame(cbind(xgrid13,ygrid13))


gm12 = ggplot(data = totgrid12,aes(x = X1,y = X2,colour = as.factor(ygrid12)))+
    geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC2")+
  geom_point(data =ndat,aes(x = PC1,y = PC2,colour = as.factor(y)))
gm12

gm13 = ggplot(data = totgrid13,aes(x = X1,y = X2,colour = as.factor(ygrid13)))+
    geom_point(size = .1)+
  labs(colour = "y",x ="PC1",y = "PC3")+
  geom_point(data =ndat,aes(x = PC1,y = PC3,colour = as.factor(y)))
gm13

ggarrange(g12, gm12,g13, gm13,
          labels = c("A", "B","C","D"),
          ncol = 2, nrow = 2)
```