# SVM method to classify the handwritten digits

Here we use the library “e1071” of R

R instruction of svm:

svm(x, y = NULL, scale = TRUE, type = NULL, kernel =

"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),

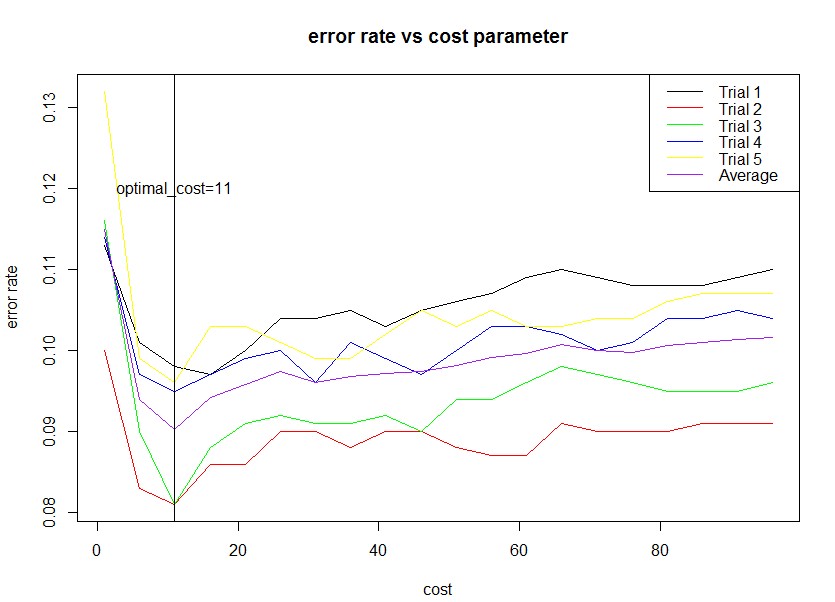
coef0 = 0, cost = 1, nu = 0.5,

class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,

shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,

..., subset, na.action = na.omit)

After trying some parameters, I find that the cost parameter affect the error rate a lot. Therefore, I decide to run the algorithm 5 times with the training data by ranging the c parameter from 1to 100. The plots of error rate vs cost parameter are in the following and we can find that the optimal cost is 11 with the average corresponding error rate 0.0902



With the optimal cost, I then train the whole training data and test on the testing data.

Related codes are in the following:

#deal with the data

training.data=as.data.frame(cbind(training.label,training.data))

training.data[,1]=as.factor(training.data[,1])

test.data=as.data.frame(cbind(test.label,test.data))

test.data[,1]=as.factor(test.data[,1])

colnames(training.data)=c("Y",paste("x.",1:400,sep=""))

colnames(test.data)=c("Y",paste("x.",1:400,sep=""))

svm.model1=svm(training.data$Y~.,data = training.data, cost = 11,method="class")

#calculate the time that used to predict and repot the errorrate

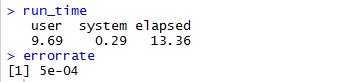
start\_time=proc.time()

svm.pred1=predict(svm.model1,newdata=test.data,type="class")

end\_time=proc.time()

run\_time=end\_time-start\_time

errorrate=sum(svm.pred1!=test.label)/nrow(test.data)



|  |  |
| --- | --- |
| Average time of one data stream(s) | Error rate |
| 1.3e(-3) | 5e-04 |

#code:

# use the SVM method to classify the handwritten digits

library("e1071")

data=load("C:/Users/Christina/Desktop/digitsdata.RData")

image(t(1 - training.data[3,1,,])[,20:1],col=gray(seq(0, 1, length.out=256)),axes=FALSE, asp=1)

num.class <- dim(training.data)[1] # Number of classes

num.training <- dim(training.data)[2] # Number of training data per class

d <- prod(dim(training.data)[3:4]) # Dimension of each training image (rowsxcolumns)

num.test <- dim(test.data)[2] # Number of test data

dim(training.data) <- c(num.class \* num.training, d) # Reshape training data to 2-dim matrix

dim(test.data) <- c(num.class \* num.test, d) # Same for test.

training.label <- rep(0:9, num.training) # Labels of training data.

test.label <- rep(0:9, num.test) # Labels of test data

# for the conveniece of dealing with data, we try to set training label as the rownames of the

#training data

rownames(training.data)=training.label

rownames(test.data)=test.label

error\_rate=matrix(0,nrow=5,ncol=20)

for(i in 1:5){

# we sample rows aaccording to their labels and get the corresponding traing data and test data.

#class 0

class0=training.data[rownames(training.data)=="0",]

train\_0=sample((nrow(class0)),size=400,replace = FALSE, prob = NULL)

train0=class0[train\_0, ]

test0=class0[-train\_0, ]

#class 1

class1=training.data[rownames(training.data)=="1",]

train\_1=sample(nrow(class1),size=400,replace = FALSE, prob = NULL)

train1=class1[train\_1, ]

test1=class1[-train\_1, ]

#class 2

class2=training.data[rownames(training.data)=="2",]

train\_2=sample(nrow(class2), size=400,replace = FALSE, prob = NULL)

train2=class2[train\_2, ]

test2=class2[-train\_2, ]

# class 3

class3=training.data[rownames(training.data)=="3",]

train\_3=sample(nrow(class3), size=400,replace = FALSE, prob = NULL)

train3=class3[train\_3, ]

test3=class3[-train\_3, ]

#class 4

class4=training.data[rownames(training.data)=="4",]

train\_4=sample(nrow(class4), size=400,replace = FALSE, prob = NULL)

train4=class4[train\_4, ]

test4=class4[-train\_4, ]

#class 5

class5=training.data[rownames(training.data)=="5",]

train\_5=sample(nrow(class5), size=400,replace = FALSE, prob = NULL)

train5=class5[train\_5, ]

test5=class5[-train\_5, ]

#class 6

class6=training.data[rownames(training.data)=="6",]

train\_6=sample(nrow(class6), size=400,replace = FALSE, prob = NULL)

train6=class6[train\_6, ]

test6=class6[-train\_6, ]

#class 7

class7=training.data[rownames(training.data)=="7",]

train\_7=sample(nrow(class7), size=400,replace = FALSE, prob = NULL)

train7=class7[train\_7, ]

test7=class7[-train\_7, ]

# class 8

class8=training.data[rownames(training.data)=="8",]

train\_8=sample(nrow(class8), size=400,replace = FALSE, prob = NULL)

train8=class8[train\_8, ]

test8=class8[-train\_8, ]

#class 9

class9=training.data[rownames(training.data)=="9",]

train\_9=sample(nrow(class9), size=400,replace = FALSE, prob = NULL)

train9=class9[train\_9, ]

test9=class9[-train\_9, ]

# we use the covariance mean as the whole same covarice

train=rbind(train0,train1,train2,train3,train4,train5,train6,train7,train8,train9)

test=rbind(test0,test1,test2,test3,test4,test5,test6,test7,test8,test9)

#svm

train\_label=as.numeric(rownames(train))

train=as.data.frame(cbind(train\_label,train))

train[,1]=as.factor(train[,1])

test\_label=as.numeric(rownames(test))

test=as.data.frame(cbind(test\_label,test))

test[,1]=as.factor(test[,1])

colnames(train)=c("Y",paste("x.",1:400,sep=""))

colnames(test)=c("Y",paste("x.",1:400,sep=""))

for (j in 1:20){#we choose cost from one to one hundred

m=seq(1,100,by=5)

c=m[j]

model.svm=svm(train$Y~.,method="class",data=train,cost=c)

prediction.SVM=predict(model.svm,newdata=test,type="class")

error\_rate[i,j]=sum(test$Y!=prediction.SVM)/nrow(test)

}

}

mean\_error\_rate=as.matrix(colMeans(error\_rate))

cost.optimal=3\*apply(mean\_error\_rate,2,which.min)+2

m=as.matrix(m)

matplot(m,cbind(t(error\_rate),mean\_error\_rate),type='l',col=c('black','red','green','blue','yellow','purple'),ylab='error rate',xlab='cost',lty =1,cex=2)

title(main='error rate vs cost parameter')

legend(legend=c('Trial 1','Trial 2','Trial 3','Trial 4','Trial 5','Average'),col=c('black','red','green','blue','yellow','purple'),'topright',lty=1)

abline(v=cost.optimal)

text(x=11,y=0.12,label='optimal\_cost=11')

# we get that the optimal cost is when cost=11 and the corresponding errorrate is 0.902

# do svm on the whole training set and do prediction on the test data

#deal with the data

training.data=as.data.frame(cbind(training.label,training.data))

training.data[,1]=as.factor(training.data[,1])

test.data=as.data.frame(cbind(test.label,test.data))

test.data[,1]=as.factor(test.data[,1])

colnames(training.data)=c("Y",paste("x.",1:400,sep=""))

colnames(test.data)=c("Y",paste("x.",1:400,sep=""))

svm.model1=svm(training.data$Y~.,data = training.data, cost = 11,method="class")

#calculate the time that used to predict and repot the errorrate

start\_time=proc.time()

svm.pred1=predict(svm.model1,newdata=test.data,type="class")

end\_time=proc.time()

run\_time=end\_time-start\_time

errorrate=sum(svm.pred1!=test.label)/nrow(test.data)