**###Unit 4: 進階分類**

**整合式學習ensemble learning (random forest, bagging & boosting)**

install.packages("rpart")

library(rpart)

install.packages("randomForest")

library(randomForest)

install.packages("adabag")

library(adabag)

\*\*\*\*\*Random Forest (iris/wine)

attach(iris)

ind<-sample(2, nrow(iris), replace=T, prob=c(0.6, 0.4))

tr.iris<-iris[ind==1, ]

ts.iris<-iris[ind==2, ]

iris.rf<-randomForest(Species~., data=tr.iris, ntree=50, importance=T, proximity=T, na.action=na.omit)

importance(iris.rf)

iris.rf$confusion

MDSplot(iris.rf, tr.iris$Species, pch=as.numeric(tr.iris$Species))

plot(iris.rf)

print(iris.rf)

iris.pred<-predict(iris.rf, newdata=ts.iris)

table(ts.iris$Species, iris.pred)

attach(redwine) #紅酒資料集

acc.red=0

for (k in 3:8)

{

red.rf<-randomForest(quality~., data=tr.red, ntree=75, mtry=k, importance=T, proximity=T, na.action=na.omit)

red.pred<-predict(red.rf, newdata=ts.red)

acc.red[k-2]<- sum( as.numeric(red.pred == ts.red[,12]) )/ nrow(ts.red)

}

plot(3:8, acc.red, xlab="mtry", ylab="acuracy")

red.rf<-randomForest(quality~., data=tr.red, ntree=75, mtry=4, importance=T, proximity=T, na.action=na.omit)

importance(red.rf)

plot(red.rf)

print(red.rf)

red.pred<-predict(red.rf, newdata=ts.red)

table(ts.red$quality, red.pred)

sum( as.numeric(red.pred == ts.red[,12]) )/ nrow(ts.red)

attach(wine) #白酒資料集

white.rf<-randomForest(quality~., data=tr.white, ntree=100, mtry=5, importance=T, proximity=T, na.action=na.omit)

importance(white.rf)

plot(white.rf)

print(white.rf)

white.pred<-predict(white.rf, newdata=ts.white)

table(ts.white$quality, white.pred)

sum( as.numeric(white.pred == ts.white[,12]) )/ nrow(ts.white)

#心臟病資料集

heart <-read.table("heart.dat", header=T, sep=" ")

colnames(heart)=c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "y") #更改欄位名稱

heart$y<-as.factor(heart$y)

id=sample(1:nrow(heart), round(nrow(heart)/3))

tr.heart=heart[-id,]

ts.heart=heart[id,]

heart.rf<-randomForest(y~., data= tr.heart, ntree=50, mtry=4, importance=T, proximity=T)

importance(heart.rf)

plot(heart.rf)

print(heart.rf)

rf.ht<-predict(heart.rf, newdata= ts.heart)

table(ts.heart$y, rf.ht)

sum( as.numeric(rf.ht == ts.heart[, 14]) )/ nrow(ts.heart)

bag.ht<-bagging(y~., tr.heart, mfinal=10, control=rpart.control(maxdepth=5))

names(bag.ht)

bag.ht$class[11:20]

bag.ht$importance

ht.bg<-predict(bag.ht, ts.heart)

names(ht.bg)

ht.bg$confusion

sum( as.numeric(ht.bg$class == ts.heart[, 14]) )/ nrow(ts.heart)

boo.ht<-boosting(y~., tr.heart, mfinal=10)

boo.ht$importance

bt.ht<-predict(boo.ht, ts.heart)

names(bt.ht)

bt.ht$confusion

sum( as.numeric(bt.ht$class == ts.heart[, 14]) )/ nrow(ts.heart)

bank<- read.table("bank.csv", header=T, sep=";") #銀行資料集

attach(bank)

id=sample(1:nrow(bank), round(nrow(bank)/3))

tr.bank=bank[-id,]

ts.bank=bank[id,]

bank.rf<-randomForest(y~., data=tr.bank, ntree=100, mtry=6, importance=T, proximity=T)

importance(bank.rf)

plot(bank.rf)

print(bank.rf)

rf.bk<-predict(bank.rf, newdata= ts.bank)

table(ts.bank$y, rf.bk)

sum( as.numeric(rf.bk == ts.bank[, 17]) )/ nrow(ts.bank)

bag.bk<-bagging(y~., tr.bank, mfinal=10) #mfinal:基礎分類器個數

names(bag.bk)

bag.bk$importance

bg.bk<-predict(bag.bk, ts.bank)

names(bg.bk)

bg.bk$confusion

sum( as.numeric(bg.bk$class == ts.bank[, 17]) )/ nrow(ts.bank)

boo.bk<-boosting(y~., tr.bank, mfinal=10)

boo.bk$importance

bt.bk<-predict(boo.bk, ts.bank)

names(bt.bk)

bt.bk$confusion

sum( as.numeric(bt.bk$class == ts.bank[, 17]) )/ nrow(ts.bank)

install.packages("cluster")

install.packages("clusterSim")

library(cluster)

library(clusterSim) #呼叫指令進行資料正規化

attach(wine) #白酒資料集

white.nm<-data.Normalization (wine[, -12], type="n4", normalization="column")

white.id=sample(nrow(wine), 3000)

tr.white=white.nm[white.id,]

tr.white$quality= wine[white.id, 12]

ts.white= white.nm[-white.id,]

ts.white$quality=wine[-white.id, 12]

acc.white=0

for (i in 8:16)

{

bag.white<-bagging(quality~., tr.white, mfinal=i)

white.bg<-predict(bag.white, ts.white)

acc.white[i-7]<- sum( as.numeric(white.bg$class == ts.white[,12]) )/ nrow(ts.white)

}

plot(8:16, acc.white, xlab="mfinal", ylab="acuracy")

bag.white<-bagging(quality~., tr.white, mfinal=8)

bag.white$importance

white.bg<-predict(bag.white, ts.white)

white.bg$confusion

sum( as.numeric(white.bg$class == ts.white[,12]) )/ nrow(ts.white)

attach(redwine) #紅酒資料集

red.nm<-data.Normalization (redwine[, -12], type="n4", normalization="column")

red.id=sample(nrow(redwine), 1000)

tr.red=red.nm[red.id, ]

tr.red$quality= redwine[red.id, 12]

ts.red=red.nm[-red.id, ]

ts.red$quality=redwine[-red.id, 12]

acc.red=0

for (j in 6:15)

{

boo.red<-boosting(quality~., tr.red, mfinal=j, boos=T)

red.bt<-predict(boo.red, ts.red)

acc.red[j-5]<- sum( as.numeric(red.bt$class == ts.red[,12]) )/ nrow(ts.red)

}

plot(6:15, acc.red, xlab="mfinal", ylab="acuracy")

boo.red<-boosting(quality~., tr.red, mfinal=12, boos=T)

boo.red$importance

red.bt<-predict(boo.red, ts.red)

red.bt$confusion

sum( as.numeric(red.bt$class == ts.red[,12]) )/ nrow(ts.red)

**###類神經網路 (neuralnet- breast\_cancer)**

install.packages("neuralnet")

library(neuralnet)

cancer\_data <-read.csv("breast\_cancer.csv", header=T, sep=",") #乳癌資料集

cancer<- cancer\_data[cancer\_data$Bare.Nuclei!= 9999, 2:11]

attach(cancer)

cancer1<-data.frame(cancer[,1:9], as.numeric(cancer$Class))

colnames(cancer1)=c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "Y1")

trnet1<-neuralnet(Y1~ x1+x2+x3+x4+x5+x6+x7+x8+x9, hidden=6, data=cancer1)

pred1=compute(trnet1, cancer1[, -10])$net.result

name=c("benign", "malignant")

table(cancer$Class, name[round(pred1)] )

sum( as.numeric(name[round(pred1)] == cancer$Class) )/ nrow(cancer)

**\*\*\*\*\*類神經網路 (nnet- 紅酒/白酒)**

install.packages("nnet")

library(nnet)

attach(redwine)

red.id=sample(nrow(redwine), 1000)

tr.red=redwine[red.id, ]

ts.red=redwine[-red.id, ]

cor.tr=0

cor.ts=0

rd=1/max(abs(redwine[red.id,-12]))

for(i in 1:20) #尋找隱藏層適合的神經元個數

{

model=nnet(quality~., data= tr.red, maxit=500, rang=rd, size=i, decay=5e-4)

#模型建立,計算在不同隱藏層神經元個數下的error

pred.tr=predict(model, tr.red[,-12], type="class") #訓練集

table(tr.red[,12], pred.tr)

cor.tr[i]<- sum( as.numeric(pred.tr == tr.red[, 12]) )/ nrow(tr.red)

pred.ts=predict(model, ts.red[,-12], type="class") #測試集

table(ts.red[,12], pred.ts)

cor.ts[i]<- sum( as.numeric(pred.ts == ts.red[, 12]) )/ nrow(ts.red)

}

plot(1:20, cor.tr, 'l', col=1, lty=1, ylab="Accuracy", xlab="Hidden Neurons", ylim=

c(min(min(cor.tr), min(cor.ts)), max(max(cor.tr), max(cor.ts))))

lines(1:20, cor.ts, col=2, lty=3)

points(1:20, cor.tr, col=1, pch="+")

points(1:20, cor.ts, col=2, pch="o")

legend(15, 0.7, "train", bty="n", cex=1)

legend(15, 0.6, "test", bty="n", cex=1)

ann=nnet(quality~., data=redwine, subset=red.id, size=19, rang=rd, decay=5e-4, maxit=500)

summary(ann)

pred.tr=predict(ann, tr.red[,-12], type="class")

table(tr.red[,12], pred.tr)

sum( as.numeric(pred.tr == tr.red[, 12]) )/ nrow(tr.red)

pred.ts=predict(ann, ts.red[,-12], type="class")

table(ts.red[,12], pred.ts)

sum( as.numeric(pred.ts == ts.red[, 12]) )/ nrow(ts.red)

**\*\*\*ANN (whitewine)**

install.packages("nnet")

library(nnet)

attach(wine)

white.id=sample(nrow(wine), 3000)

tr.white=wine[white.id, ]

ts.white=wine[-white.id, ]

cor.tr=0

cor.ts=0

rw=1/max(abs(wine[white.id,-12]))

for(i in 1:20) #尋找隱藏層適合的神經元個數

{

model=nnet(quality~., data= tr.white, maxit=500, rang=rw, size=i, decay=5e-4) #模型建立,並計算在不同隱藏層神經元個數下的error

pre.tr=predict(model, tr.white[,-12], type="class")

table(tr.white[,12], pre.tr)

cor.tr[i]<- sum( as.numeric(pre.tr == tr.white[, 12]) )/ nrow(tr.white)

pre.ts=predict(model, ts.white[,-12], type="class")

table(ts.white[,12], pre.ts)

cor.ts[i]<- sum( as.numeric(pre.ts == ts.white[, 12]) )/ nrow(ts.white)

}

plot(1:20, cor.tr, 'l', col=1, lty=1, ylab="Accuracy", xlab="Hidden Neurons", ylim=

c(min(min(cor.tr), min(cor.ts)), max(max(cor.tr), max(cor.ts))))

lines(1:20, cor.ts, col=2, lty=3)

points(1:20, cor.tr, col=1, pch="+")

points(1:20, cor.ts, col=2, pch="o")

ann1=nnet(quality~., data=tr.white, subset=white.id, size=17, rang=rw, decay=5e-4, maxit=500)

summary(ann1)

pre.tr1=predict(ann1, tr.white[,-12], type="class") #訓練集

table(tr.white[,12], pre.tr1)

sum( as.numeric(pre.tr1 == tr.white[, 12]) )/ nrow(tr.white)

pre.ts1=predict(ann1, ts.white[,-12], type="class") #測試集

table(ts.white[,12], pre.ts1)

sum( as.numeric(pre.ts1 == ts.white[, 12]) )/ nrow(ts.white)

y<- class.ind(tr.white[, 12])

ann2=nnet(tr.white[,-12], y, decay=5e-4, maxit=300, size=17, rang=r )

summary(ann2)

name=c("bad", "good", "medium")

pre.tr2=predict(ann2, tr.white[,-12])

table(tr.white[,12], name[max.col(pre.tr2)] )

sum( as.numeric(name[max.col(pre.tr2)] == tr.white[, 12]) )/ nrow(tr.white)

pre.ts2=predict(ann2, ts.white[,-12])

table(ts.white[,12], name[max.col(pre.ts2)] )

sum( as.numeric(name[max.col(pre.ts2)] == ts.white[, 12]) )/ nrow(ts.white)

**#####ANN (neuralnet)**

install.packages("neuralnet")

library(neuralnet)

attach(redwine)

tr.red$y<- as.numeric(tr.red$quality)

ts.red$y<- as.numeric(ts.red$quality)

ann.red<- neuralnet(y~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11, tr.red, hidden=16, rep=3, threshold=2.5, learningrate=0.1)

summary(ann.red)

name=c("bad", "excellent", "good", "medium") #還原原始類別

pre.tr<- compute(ann.red, tr.red[,1:11])$net.result #訓練集

table(tr.red[,12], name[round(pre.tr)] )

sum( as.numeric(name[round(pre.tr)] == tr.red[, 12]) )/ nrow(tr.red)

pre.ts<- compute(ann.red, ts.red[,1:11])$net.result #測試集

pre.ts[which(round(pre.ts)>4)]<-4 #檢查是否over shooting

pre.ts[which(round(pre.ts)<1)]<-1 #檢查是否over shooting

table(ts.red[,12], name[round(pre.ts)] )

sum( as.numeric(name[round(pre.ts)] == ts.red[, 12]) )/ nrow(ts.red)

**\*\*\*\*\*SVM (iris)**

install.packages("e1071")

library(e1071)

attach(iris)

x <- subset(iris, select = -Species)

y <- Species

tr.iris=rbind(iris[1:30,], iris[51:80,], iris[101:130,])

ts.iris=rbind(iris[31:50,], iris[81:100,], iris[131:150,])

svm1 <- svm(Species ~ ., data = iris, cost=10, gamma=10)

summary(svm1)

pred1 <- predict(svm1, x)

table(pred=pred1, true=y)

svm2 <- svm(x, y, cost=10, gamma=10)

summary(svm2)

pred2 <- fitted(svm2)

table(pred=pred2, true=y)

# Seeking the optimal parameters (尋找最佳參數)

tuned<-tune.svm(Species~., data=tr.iris, gamma=10^(-3:0), cost=10^(-1:2))

summary(tuned)

svm3 <- svm(Species ~ ., data =tr.iris, cost=10, gamma=0.1)

pred3 <- predict(svm3, tr.iris)

table(pred3, tr.iris[,5])

pred4<-predict(svm3, ts.iris)

table(pred4, ts.iris[,5])

\*\*\*SVM (gender)

gender\_size <-read.csv("gender\_size.csv")

attach(gender\_size)

tuned<-tune.svm(Gender~., data= gender\_size, gamma=10^(-3:0), cost=10^(-1:2))

summary(tuned)

svm4 <- svm(Gender~., data= gender\_size, kernel="radial", cost=1, gamma=1)

summary(svm4)

pred\_svm <- predict(svm4, gender\_size[,-4])

table(pred\_svm, gender\_size[,4])

sum( as.numeric(pred\_svm == gender\_size[,4]) )/ nrow(gender\_size)

#visualize (classes by color, SV by crosses):

plot(svm4, gender\_size, Height~Weight, fill=FALSE, symbolPalette=c("red", "blue"), svSymbol="+", dataSymbol="o")

plot(svm4, gender\_size, Waist~Height, fill=FALSE, symbolPalette=c("red", "blue"), svSymbol="+", dataSymbol="o")

plot(svm4, gender\_size, Waist~Weight, fill=FALSE, symbolPalette=c("red", "blue"), svSymbol="+", dataSymbol="o")

**\*\*\*\*\* Selection of Multi-SVM parameters**

attach(iris)

x <- subset(iris, select = -Species)

y <- Species

tt=c("C-classification", "nu-classification")

kk=c("linear", "polynomial", "radial", "sigmoid")

fitSVM=array(0, dim=c(150,2,4))

error=matrix(0,2,4)

z=as.integer(y)

for (i in 1:2)

{ for (j in 1:4)

{

model=svm(x, y, type=tt[i], kernel=kk[j])

fitSVM[, i ,j]=predict(model, x)

error[i, j]=sum(fitSVM[ ,i ,j]!=z)

}

}

dimnames(error)=list(tt, kk)

error

**\*\*\*\*\*SVM (cancer/wine)**

cancer\_data <-read.csv("breast\_cancer.csv", header=T, sep=",") #乳癌資料集

cancer<- cancer\_data[cancer\_data$Bare.Nuclei!= 9999, 2:11]

tuned1<-tune.svm(Class~., data= cancer, gamma=10^(-3:-1), cost=10^(-1:2))

summary(tuned1)

svm5 <- svm(Class~., data= cancer, kernel="radial", cost=1, gamma=0.01)

summary(svm5)

cancer.svm <- predict(svm5, cancer[, -10])

table(cancer[,10], cancer.svm)

sum( as.numeric(cancer.svm == cancer[, 10]) )/ nrow(cancer)

attach(redwine) #紅酒資料集

red.id=sample(nrow(redwine), 1000)

tr.red=redwine[red.id,]

ts.red=redwine[-red.id,]

tuned2<-tune.svm(quality~., data= tr.red, gamma=10^(-3:0), cost=10^(-1:2))

summary(tuned2)

svm6 <- svm(quality~., data= tr.red, cross=5, cost=100, gamma=0.1)

summary(svm6)

svm6$index

red.svm <- predict(svm6, ts.red[, -12])

table(ts.red [,12], red.svm)

sum( as.numeric(red.svm == ts.red[,12]) )/ nrow(ts.red)

attach(wine) #白酒資料集

white.id =sample(nrow(wine), 3000)

tr.white=wine[white.id,]

ts.white=wine[-white.id,]

tuned3<-tune.svm(quality~., data= tr.white, gamma=10^(-3:0), cost=10^(-1:2))

summary(tuned3)

svm7 <- svm(quality~., data= tr.white, cross=5, cost=1, gamma=1)

summary(svm7)

svm7$index

white.svm <- predict(svm7, ts.white[, -12])

table(ts.white[,12], white.svm)

sum( as.numeric(white.svm == ts.white[,12]) )/ nrow(ts.white)