《数据挖掘与商务智能》课程实验指导

#####关联规则#####

library(Matrix)

library(lattice)

library(arules) ##加载程序包

data(Groceries) ##加载Groceries数据集

typeof(Groceries) ##Groceries数据集的类型

cc <- as(Groceries,'data.frame') ##将S4类型转换为data.frame类型

##setwd("G:\\myProject\\RDoc\\Unit1") ##设置RGui的工作路径

write.csv(cc,"d://Groceries.csv") ##导出Groceries数据集

items<-Groceries@data

dd<-as.matrix(items)

inspect(Groceries[1:10])

rules0=apriori(Groceries)

rules0=apriori(Groceries,parameter=list(support=0.001,confidence=0.5))

rules0=apriori(Groceries,parameter=list(support=0.001,confidence=0.5))

rules0

inspect(rules0[1:10])

rules1=apriori(Groceries,parameter=list(support=0.005,confidence=0.5))

rules1

rules2=apriori(Groceries,parameter=list(support=0.005,confidence=0.6))

rules2

rules3=apriori(Groceries,parameter=list(support=0.005,confidence=0.64))

rules3

inspect(rules3)

rules.sorted\_sup=sort(rules0,by="support")

inspect(rules.sorted\_sup[1:5])

rules.sorted\_con=sort(rules0,by="confidence")

inspect(rules.sorted\_con[1:5])

rules.sorted\_lift=sort(rules0,by="lift")

inspect(rules.sorted\_lift[1:5])

rules4=apriori(Groceries,parameter=list(maxlen=2,supp=0.001,conf=0.1),appearance=list(rhs="mustard",default="lhs"))

inspect(rules4)

itemsets\_apr=apriori(Groceries,parameter=list(supp=0.001,target="frequent itemsets"),control=list(sort=-1))

itemsets\_apr

inspect(itemsets\_apr[1:5])

itemsets\_ecl=eclat(Groceries,parameter=list(minlen=1,maxlen=3,supp=0.001,target="frequent itemsets"),control=list(sort=-1))

itemsets\_ecl

inspect(itemsets\_ecl[1:5])

library(arulesViz)

rules5=apriori(Groceries,parameter=list(support=0.002,confidence=0.5))

rules5

plot(rules5)

plot(rules5,measure=c("support","lift"),shading="confidence")

rules5=apriori(Groceries,parameter=list(support=0.002,confidence=0.5))

rules5

plot(rules5)

plot(rules5,measure=c("support","lift"),shading="confidence")

plot(rules5,interactive=TRUE)

plot(rules5,shading="order",control=list(main="Two-key plot"))

plot(rules5,method="grouped")

plot(rules5[1:50],method="matrix",measure="lift")

plot(rules5[1:50],method="matrix3D",measure="lift")

plot(rules5[1:50],method="paracoord")

#####聚类#####

countries=read.csv("d:/R\_ws/dmData/data\_countries/countries.csv")

dim(countries)

head(countries)

names(countries)=c("country","birth","death")

var=countries$country

var=as.character(var)

head(var)

for(i in 1:224)row.names(countries)[i]=var[i]

head(countries)

plot(countries$birth,countries$death)

C=which(countries$country=="CHINA")

T=which(countries$country=="TAIWAN")

H=which(countries$country=="HONG KONG")

I=which(countries$country=="INDIA")

J=which(countries$country=="JAPAN")

U=which(countries$country=="UNITED STATES")

M=which.max(countries$birth)

points(countries[c(C,T,H,I,U,J,M),-1],pch=16)

legend(countries$birth[C],countries$death[C],"CHINA",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[T],countries$death[T],"TAIWAN",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[H],countries$death[H],"HONG KONG",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[I],countries$death[I],"INDIA",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[U],countries$death[U],"UNITED STATES",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[J],countries$death[J],"JAPAN",bty="n",xjust=0.5,cex=0.8)

legend(countries$birth[M],countries$death[M],countries$country[M],bty="n",xjust=0.5,cex=0.8)

fit\_kml=kmeans(countries[,-1],center=3)

print(fit\_kml)

fit\_kml$centers

fit\_kml$totss;fit\_kml$tot.withinss;fit\_kml$betweenss

fit\_kml$betweenss+fit\_kml$tot.withinss

plot(countries[,-1],pch=(fit\_kml$cluster-1))

points(fit\_kml$centers,pch=8)

legend(fit\_kml$centers[1,1],fit\_kml$centers[1,2],"Center\_1",bty="n",xjust=1,yjust=0,cex=0.8)

legend(fit\_kml$centers[2,1]-2,fit\_kml$centers[2,2],"Center\_2",bty="n",xjust=1,yjust=0,cex=0.8)

legend(fit\_kml$centers[3,1],fit\_kml$centers[3,2],"Center\_3",bty="n",xjust=1,yjust=0.5,cex=0.8)

result=rep(0,60)

for(k in 1:60)

{

fit\_km=kmeans(countries[,-1],center=k)

result[k]=fit\_km$betweenss/fit\_km$totss

}

round(result,2)

plot(1:60,result,type="b",main="Choosing the Optimal Number of Cluster",

xlab="number of cluster:1 to 60",ylab="betweenss/totss")

points(10,result[10],pch=16)

legend(10,result[10],paste("(10,",sprintf("%.1f%%",result[10]\*100),")",sep=""),bty="n",xjust=0.3,cex=0.8)

fit\_km2=kmeans(countries[,-1],center=10)

cluster\_CHINA=fit\_km2$cluster[which(countries$country=="CHINA")]

which(fit\_km2$cluster==cluster\_CHINA)

library(cluster)

fit\_pam=pam(countries[ ,-1],3)

print(fit\_pam)

head(fit\_pam$data)

fit\_pam$call

fit\_pam1=pam(countries[,-1],3,keep.data=FALSE)

fit\_pam1$data

fit\_pam2=pam(countries[,-1],3,cluster.only=TRUE)

print(fit\_pam2)

which(fit\_kml$cluster!=fit\_pam$cluster)

plot(countries[,-1],pch=(fit\_pam$cluster-1))

c1=which(rownames(countries)==rownames(fit\_pam$medoids)[1])

c2=which(rownames(countries)==rownames(fit\_pam$medoids)[2])

c3=which(rownames(countries)==rownames(fit\_pam$medoids)[3])

for(i in 1:3)

{ var=c(c1,c2,c3)

points(countries[var[i],-1],pch=fit\_pam$cluster[var[i]]+14) }

legend(fit\_pam$medoids[1,1],fit\_pam$medoids[1,2],paste("Center\_1:",rownames(fit\_pam$medoids)[1]),bty="n",xjust=0.5,yjust=0,cex=0.8)

legend(fit\_pam$medoids[2,1]-1.2,fit\_pam$medoids[2,2],paste("Center\_2:",rownames(fit\_pam$medoids)[2]),bty="n",xjust=0,yjust=0.5,cex=0.8)

legend(fit\_pam$medoids[3,1],fit\_pam$medoids[3,2]+3.5,paste("Center\_3:",rownames(fit\_pam$medoids)[3]),bty="n",xjust=0.5,yjust=0,cex=0.8)

points(countries[c(21,23,33),-1],pch=12)

legend(countries$birth[21],countries$death[21],"MONGOLIA",bty="n",xjust=0.5,yjust=0,cex=0.8)

legend(countries$birth[23]-1.2,countries$death[23],"SYRIA",bty="n",xjust=0,yjust=0.5,cex=0.8)

legend(countries$birth[33]-1.2,countries$death[33],"PANAMA",bty="n",xjust=0,yjust=0.5,cex=0.8)

fit\_hc=hclust(dist(countries[,-1]))

print(fit\_hc)

plot(fit\_hc)

group\_k3=cutree(fit\_hc,k=3)

group\_k3

table(group\_k3)

group\_h18=cutree(fit\_hc,h=18)

group\_h18

table(group\_h18)

sapply(unique(group\_k3),function(g)countries$country[group\_k3==g])

plot(fit\_hc)

rect.hclust(fit\_hc,k=4,border="light grey")

rect.hclust(fit\_hc,k=3,border="dark grey")

rect.hclust(fit\_hc,k=7,which=c(2,6),border="dark grey")

library(mclust)

fit\_EM=Mclust(countries[,-1])

summary(fit\_EM)

summary(fit\_EM,parameters=TRUE)

plot(fit\_EM)

countries\_BIC=mclustBIC(countries[,-1])

countries\_BICsum=summary(countries\_BIC,data=countries[,-1])

countries\_BICsum

countries\_BIC

plot(countries\_BIC,G=1:7,col="black")

names(countries\_BICsum)

mclust2Dplot(countries[,-1], classification=countries\_BICsum$classification,

parameters=countries\_BICsum$parameters, col ="black")

countries\_Dens=densityMclust(countries[,-1])

plot(countries\_Dens,countries[,-1],col="grey",nlevels=55)

plot(countries\_Dens,type = "persp",col = grey(0.8))

##DBSCAN##

library(fpc)

ds1=dbscan(countries[,-1],eps=1,MinPts=5)

ds2=dbscan(countries[,-1],eps=4,MinPts=5)

ds3=dbscan(countries[,-1],eps=4,MinPts=2)

ds4=dbscan(countries[,-1],eps=8,MinPts=2)

par(mfcol=c(2,2))

plot(ds1,countries[,-1],main="1: MinPts=5 eps=1")

plot(ds3,countries[,-1],main="3: MinPts=2 eps=4")

plot(ds2,countries[,-1],main="2: MinPts=5 eps=4")

plot(ds4,countries[,-1],main="4: MinPts=2 eps=8")

d=dist(countries[,-1])

max(d);min(d)

library(ggplot2)

interval=cut\_interval(d,30)

table(interval)

which.max(table(interval))

for(i in 3:5)

{ for(j in 1:10)

{ ds=dbscan(countries[,-1],eps=i,MinPts=j)

print(ds)

}

}

ds5=dbscan(countries[,-1],eps=3,MinPts=2)

ds6=dbscan(countries[,-1],eps=4,MinPts=5)

ds7=dbscan(countries[,-1],eps=5,MinPts=9)

par(mfcol=c(1,3))

plot(ds5,countries[,-1],main="1: MinPts=2 eps=3")

plot(ds6,countries[,-1],main="3: MinPts=5 eps=4")

plot(ds7,countries[,-1],main="2: MinPts=9 eps=5")

#####决策树#####

library(mvpart) ##安装了Rtools

data(car.test.frame)

head(car.test.frame)

car.test.frame$Mileage=100\*4.546/(1.6\*car.test.frame$Mileage)

names(car.test.frame)=c("价格","产地","可靠性","油耗","类型","车重",

"发动机功率","净马力")

head(car.test.frame)

str(car.test.frame)

summary(car.test.frame)

Group\_Mileage=matrix(0,60,1)

Group\_Mileage[which(car.test.frame$"油耗">=11.6)]="A"

Group\_Mileage[which(car.test.frame$"油耗"<=9)]="C"

Group\_Mileage[which(Group\_Mileage==0)]="B"

car.test.frame$"分组油耗"=Group\_Mileage

car.test.frame[1:10,c(4,9)]

library(sampling)

a=round(1/4\*sum(car.test.frame$"分组油耗"=="A"))

b=round(1/4\*sum(car.test.frame$"分组油耗"=="B"))

c=round(1/4\*sum(car.test.frame$"分组油耗"=="C"))

a;b;c

sub=strata(car.test.frame,stratanames="分组油耗",size=c(c,b,a),method="srswor")

sub

Train\_Car=car.test.frame[-sub$ID\_unit,]

Test\_Car=car.test.frame[sub$ID\_unit,]

nrow(Train\_Car);nrow(Test\_Car)

library(rpart)

library(rpart.plot)

library(maptree)

formula\_Car\_Reg=油耗~价格+产地+可靠性+类型+车重+发动机功率+净马力

rp\_Car\_Reg=rpart(formula\_Car\_Reg,Train\_Car,method="anova")

print(rp\_Car\_Reg)

printcp(rp\_Car\_Reg)

summary(rp\_Car\_Reg)

rp\_Car\_Reg1=rpart(formula\_Car\_Reg,Train\_Car,method="anova",minsplit=10)

print(rp\_Car\_Reg1)

printcp(rp\_Car\_Reg1)

rp\_Car\_Reg2=rpart(formula\_Car\_Reg,Train\_Car,method="anova",cp=0.1)

print(rp\_Car\_Reg2)

printcp(rp\_Car\_Reg2)

rp\_Car\_Reg3=prune.rpart(rp\_Car\_Reg,cp=0.1)

print(rp\_Car\_Reg3)

printcp(rp\_Car\_Reg3)

rp\_Car\_Reg4=rpart(formula\_Car\_Reg,Train\_Car,method="anova",maxdepth=1)

print(rp\_Car\_Reg4)

printcp(rp\_Car\_Reg4)

rp\_Car\_Plot=rpart(formula\_Car\_Reg,Train\_Car,method="anova",minsplit=10)

print(rp\_Car\_Plot)

rpart.plot(rp\_Car\_Plot)

rpart.plot(rp\_Car\_Plot,type=4)

rpart.plot(rp\_Car\_Plot,type=4,branch=1)

rpart.plot(rp\_Car\_Plot,type=4,fallen.leaves=TRUE)

draw.tree(rp\_Car\_Plot,col=rep(1,7),nodeinfo=TRUE)

plot(rp\_Car\_Plot,uniform=TRUE,main="plot: Regression Tree")

text(rp\_Car\_Plot,use.n=TRUE,all=TRUE)

post(rp\_Car\_Plot,file="",title.="post: Regression Tree")

formula\_Car\_Cla=分组油耗~价格+产地+可靠性+类型+车重+发动机功率+净马力

rp\_Car\_Cla=rpart(formula\_Car\_Cla,Train\_Car,method="class",minsplit=5)

print(rp\_Car\_Cla)

rpart.plot(rp\_Car\_Cla,type=4,fallen.leaves=TRUE)

pre\_Car\_Cla=predict(rp\_Car\_Cla,Test\_Car,type="class")

pre\_Car\_Cla

table(Test\_Car$分组油耗,pre\_Car\_Cla)

(p=sum(as.numeric(pre\_Car\_Cla!=Test\_Car$分组油耗))/nrow(Test\_Car))

# C4.5 #

library(RWeka)

library(partykit)

names(Train\_Car)=c("Price","Country","Reliability","Mileage",

"Type","Weight","Disp.","HP","Oil\_Consumption")

Train\_Car$Oil\_Consumption=as.factor(Train\_Car$Oil\_Consumption)

formula=Oil\_Consumption~Price+Country+Reliability+Type+Weight+Disp.+HP

C45\_0=J48(formula,Train\_Car)

C45\_0

summary(C45\_0)

C45\_1=J48(formula,Train\_Car,control=Weka\_control(M=3))

C45\_1

plot(C45\_1)

#####贝叶斯网络#####

library(kknn)

data(miete)

head(miete)

dim(miete)

summary(miete)

library(sampling)

n=round(2/3\*nrow(miete)/5)

n

sub\_train=strata(miete,stratanames="nmkat",size=rep(n,5),method="srswor")

head(sub\_train)

miete2<-miete[,c(-1,-3,-12)]

data\_train=getdata(miete2,sub\_train)[,c(-15,-16,-17)]

data\_test=miete2[-sub\_train$ID\_unit,]

dim(data\_train);dim(data\_test)

head(data\_test)

library(MASS)

fit\_lda1=lda(nmkat~.,data\_train)

names(fit\_lda1)

fit\_lda1$prior

fit\_lda1$counts

fit\_lda1$means

fit\_lda1$scaling

fit\_lda1$lev

fit\_lda1$svd

fit\_lda1$N

fit\_lda1$call

fit\_lda1$terms

fit\_lda1$xlevels

fit\_lda1

fit\_lda2=lda(data\_train[,-12],data\_train[,12])

fit\_lda2

plot(fit\_lda1)

plot(fit\_lda1,dimen=1)

plot(fit\_lda1,dimen=2)

pre\_lda1=predict(fit\_lda1,data\_test)

pre\_lda1$class

pre\_lda1$posterior

table(data\_test$nmkat,pre\_lda1$class)

error\_lda1=sum(as.numeric(as.numeric(pre\_lda1$class)!=as.numeric(data\_test$nmkat)))/nrow(data\_test)

error\_lda1

library(klaR)

fit\_Bayes1=NaiveBayes(nmkat~.,data\_train)

names(fit\_Bayes1)

fit\_Bayes1$apriori

fit\_Bayes1$tables

fit\_Bayes1$levels

fit\_Bayes1$call

fit\_Bayes1$usekernel

fit\_Bayes1$varnames

fit\_Bayes2=NaiveBayes(data\_train[,-12],data\_train[,12])

fit\_Bayes2

plot(fit\_Bayes1,vars="wfl",n=50,col=c(1,"darkgrey",1,"darkgrey",1)) # 占地面积

plot(fit\_Bayes1,vars="mvdauer",n=50,col=c(1,"darkgrey",1,"darkgrey",1)) # 租赁期

plot(fit\_Bayes1,vars="nmqm",n=50,col=c(1,"darkgrey",1,"darkgrey",1)) # 每平方米净租金

pre\_Bayes1=predict(fit\_Bayes1,data\_test)

pre\_Bayes1

table(data\_test$nmkat,pre\_Bayes1$class)

error\_Bayes1=sum(as.numeric(as.numeric(pre\_Bayes1$class)!=as.numeric(data\_test$nmkat)))/nrow(data\_test)

error\_Bayes1

library(class)

fit\_pre\_knn=knn(data\_train[,-12],data\_test[,-12],cl=data\_train[,12])

fit\_pre\_knn

table(data\_test$nmkat,fit\_pre\_knn)

error\_knn=sum(as.numeric(as.numeric(fit\_pre\_knn)!=as.numeric(data\_test$nmkat)))/nrow(data\_test)

error\_knn

error\_knn=rep(0,20)

for(i in 1:20)

{ fit\_pre\_knn=knn(data\_train[,-12],data\_test[,-12],cl=data\_train[,12],k=i)

error\_knn[i]=sum(as.numeric(as.numeric(fit\_pre\_knn)!=as.numeric(data\_test$nmkat)))/nrow(data\_test)}

error\_knn

plot(error\_knn,type="l",xlab="K")

library(kknn)

fit\_pre\_kknn=kknn(nmkat~.,data\_train,data\_test[,-12])

fit\_pre\_kknn

summary(fit\_pre\_kknn)

fit=fitted(fit\_pre\_kknn)

fit

table(data\_test$nmkat,fit)

error\_kknn=sum(as.numeric(as.numeric(fit)!=as.numeric(data\_test$nmkat)))/nrow(data\_test)

error\_kknn

error\_kknn=rep(0,20)

for(i in 1:20)

{ fit\_pre\_kknn=kknn(nmkat~.,data\_train,data\_test[,-12],k=i)

error\_kknn[i]=sum(as.numeric(as.numeric(fitted(fit\_pre\_kknn))!=as.numeric(data\_test$nmkat)))/nrow(data\_test)}

error\_kknn

plot(error\_kknn,type="l",xlab="K")

sub=matrix(0,4,30)

for(i in 1:4) sub[i,]=sample(which(miete$nmkat==i),30)

SUB=sample(which(miete$nmkat=="5"),200)

subb=matrix(0,5,20)

for(i in 1:5) subb[i,]=sample(which(miete$nmkat==i),20)

data\_train=miete[c(sub,SUB),c(-1,-3,-12)]

data\_test=miete[subb,c(-1,-3,-12)]

dim(data\_train);dim(data\_test)

# setwd("D://book")

# data=read.table("u.data.txt")

data=read.table("D:\\dmData\\data\_movielens\\ml-100k\\u.data")

data=data[,-4]

names(data)=c("userid","itemid","rating")

MovieLens\_KNN(Userid=1,Itemid=61,n=50,K=10)

Userid=1;Itemid=61;n=50;K=10

MovieLens\_KNN=function(Userid,Itemid,n,K) {

sub=which(data$userid==Userid)

if(length(sub)>=n) sub\_n=sample(sub,n)

if(length(sub)<n) sub\_n=sample(sub,length(sub))

known\_itemid=data$itemid[sub\_n]

unknown\_itemid=Itemid

unknown\_sub=which(data$itemid==unknown\_itemid)

user=data$userid[unknown\_sub[-1]]

data\_all=matrix(0,1+length(user),2+length(known\_itemid))

data\_all=data.frame(data\_all)

names(data\_all)=c("userid",paste("unknown\_itemid\_",Itemid),paste("itemid\_",known\_itemid,sep=""))

item=c(unknown\_itemid,known\_itemid)

data\_all$userid=c(Userid,user)

for (i in 1:nrow(data\_all))

{

data\_temp=data[which(data$userid==data\_all$userid[i]),]

for (j in 1:length(item))

{ if(sum(as.numeric(data\_temp$itemid==item[j]))!=0)

{data\_all[i,j+1]=data\_temp$rating[which(data\_temp$itemid==item[j])]}

} }

data\_test\_x=data\_all[1,c(-1,-2)]

data\_test\_y=data\_all[1,2]

data\_train\_x=data\_all[-1,c(-1,-2)]

data\_train\_y=data\_all[-1,2]

fit=knn(data\_train\_x,data\_test\_x,cl=data\_train\_y,k=K)

list("data\_all:"=data\_all,"True Rating:"=data\_test\_y,"Predict Rating:"=fit,"User ID:"=Userid,"Item ID:"=Itemid)

}

user1=NULL

for(Item in 1:20)

user1=c(user1,MovieLens\_KNN(Userid=1,Itemid=Item,n=50,K=10)$`True Rating:`)

user1

which(user1==5)

data=read.table("D:\\dmData\\data\_movielens\\ml-100k\\u.data")

data=data[,-4]

names(data)=c("userid","itemid","rating")

Userid=1;Itemid=61;n=50;K=10

MovieLens\_KNN=function(Userid,Itemid,n,K) {

sub=which(data$userid==Userid)

if(length(sub)>=n) sub\_n=sample(sub,n)

if(length(sub)<n) sub\_n=sample(sub,length(sub))

known\_itemid=data$itemid[sub\_n]

unknown\_itemid=Itemid

unknown\_sub=which(data$itemid==unknown\_itemid)

user=data$userid[unknown\_sub[-1]]

data\_all=matrix(0,1+length(user),2+length(known\_itemid))

data\_all=data.frame(data\_all)

names(data\_all)=c("userid",paste("unknown\_itemid\_",Itemid),paste("itemid\_",known\_itemid,sep=""))

item=c(unknown\_itemid,known\_itemid)

data\_all$userid=c(Userid,user)

for (i in 1:nrow(data\_all))

{

data\_temp=data[which(data$userid==data\_all$userid[i]),]

for (j in 1:length(item))

{ if(sum(as.numeric(data\_temp$itemid==item[j]))!=0)

{data\_all[i,j+1]=data\_temp$rating[which(data\_temp$itemid==item[j])]}

} }

data\_test\_x=data\_all[1,c(-1,-2)]

data\_test\_y=data\_all[1,2]

data\_train\_x=data\_all[-1,c(-1,-2)]

data\_train\_y=data\_all[-1,2]

fit=knn(data\_train\_x,data\_test\_x,cl=data\_train\_y,k=K)

list("data\_all:"=data\_all,"True Rating:"=data\_test\_y,"Predict Rating:"=fit,"User ID:"=Userid,"Item ID:"=Itemid)

}

user1=NULL

for(Item in 1:20)

user1=c(user1,MovieLens\_KNN(Userid=1,Itemid=Item,n=50,K=10)$`True Rating:`)

user1

which(user1==5)

#####人工神经网络#####

###第一种建模格式

library(nnet)

scale01=function(x){

ncol=dim(x)[2]-1

nrow=dim(x)[1]

new=matrix(0,nrow,ncol)

for(i in 1:ncol){

max=max(x[,i])

min=min(x[,i])

for(j in 1:nrow){

new[j,i]=(x[j,i]-min)/(max-min)

}

}

new

}

wine=read.table("d:\\dmdata\\data\_wine\\winequality-white.csv",header=TRUE,sep=";") # 本文默认数据以记事本格式存储于电脑D盘中

names(wine)=c("fixed","volatile","citric","residual","chlorides","free","total","density","PH","sulphates","alcohol","quality") # 为每一个变量命名

set.seed(71)

samp=sample(1:4898,3000) # 从总样本集中抽取3000个样本作为训练集

wine[samp,1:11]=scale01(wine[samp,]) # 对样本进行预处理

r=1/max(abs(wine[samp,1:11])) # 确定参数rang的变化范围

set.seed(101)

model1=nnet(quality~.,data=wine,subset=samp,size=4,rang=r,decay=5e-4,maxit=200) # 建立神经网络模型

###第二种建模格式

x=subset(wine,select=-quality) # 提取wine数据中除quality列以外的数据作为自变量

y=wine[,12] # 提取wine数据中的quality列数据作为响应变量

y=class.ind(y) # 对响应变量进行预处理，将其变为类指标矩阵

set.seed(101)

model2=nnet(x,y,decay=5e-4,maxit=200,size=4,rang=r) # 建立神经网络模型

###针对第一种格式进行预测

x=wine[,1:11] # 确认需要进行预测的样本特征矩阵

pred=predict(model1,x) ##,type="class") # 根据模型model1对xt数据进行预测

set.seed(110)

pred[sample(1:4898,8),] # 随机挑选8个预测结果进行展示

###针对第二种格式进行预测

xt=wine[,1:11] # 确认需要进行预测的样本特征矩阵

pred=predict(model2,xt) # 根据模型model2对xt数据进行预测

dim(pred) # 查看预测结果的维度

pred[sample(1:4898,4),] # 随机挑选4个预测结果进行展示

name=c("bad","good","mid","bad","good","mid","bad") # 为三个类别确定名称

# prednew=max.col(pred) # 确定每行中最大值所在列

prednew<-apply(pred, 1, function(t) colnames(pred)[which.max(t)])

prednewn=name[as.numeric(prednew)-2] # 根据预测结果将其变为相对应的类别名称

set.seed(201)

prednewn[sample(1:4898,8)] # 随机挑选8个预测结果进行展示

true=max.col(y) # 确定真实值的每行中最大值所在列

table(true,prednewn) # 模型预测精度展示

###nnet函数使用过程中特别注意

data("iris")

samp=sample(1:150,100)

iris2<-iris

iris2[samp,1:4]=scale01(iris2[samp,])

r<-1/max(abs(iris2[samp,1:4]))

model1=nnet(Species~.,data=iris2,rang=r,size=4,maxit=500,decay=5e-4) # 建立模型model1

model2=nnet(Species~.,data=iris2,rang=r,size=4,maxit=500,decay=5e-4) # 建立模型model2

name=c("setosa","versicolor","virginica") # 为三个类别确定名称

pred1<-predict(model1,iris)

pred2<-predict(model2,iris)

# pred1=name[max.col(predict(model1,x))]

pred1\_new<-apply(pred1, 1, function(t) colnames(pred1)[which.max(t)])

# 利用第二种模型的预测方法对模型model1进行预测

# pred2\_new=name[max.col(predict(model2,x))]

pred2\_new<-apply(pred2, 1, function(t) colnames(pred2)[which.max(t)])

# 利用第二种模型的预测方法对模型model2进行预测

table(iris[,5],pred1\_new) # 模型model1预测精度展示

###实际建模操作

###确定隐藏层节点数

wine=read.csv("d:\\dmdata\\data\_wine\\winequality-red.csv",sep=";",head=TRUE) # 本文默认数据以记事本格式存储于电脑D盘中

names(wine)=c("fixed","volatile","citric","residual","chlorides","free","total","density","PH","sulphates","alcohol","quality") # 为每一个变量命名

set.seed(71)

# wine=wine[sample(1:4898,3000),]

nrow.wine=dim(wine)[1]

###原始数据归一化程序

scale01=function(x){

ncol=dim(x)[2]-1

nrow=dim(x)[1]

new=matrix(0,nrow,ncol)

for(i in 1:ncol){

max=max(x[,i])

min=min(x[,i])

for(j in 1:nrow){

new[j,i]=(x[j,i]-min)/(max-min) }

}

new

}

cha<-rep(0,times=nrow.wine) # 设置中间变量对处理后的向量进行临时存储

cha=0

for(i in 1: nrow.wine) # 针对每一个样本进行调整

{

if(wine[i,12]>6)

{

cha[i]="good" # 将品质大于6的样本品质定义为“good”

}

else if(wine[i,12]>5)

{

cha[i]="mid" # 将品质大于5却不大于6的样本品质定义为“mid”

}

else

{

cha[i]="bad" # 将品质不大于5的样本品质定义为“bad”

}

}

wine[,12]=factor(cha) # 将字符型变量转化为含有因子的变量并复制给数据集wine

set.seed(444)

samp=sample(1:nrow.wine, nrow.wine\*0.7) # 从总样本集中抽取70%的样本作为训练集

wine[samp,1:11]=scale01(wine[samp,]) # 对训练集样本进行预处理

wine[-samp,1:11]=scale01(wine[-samp,]) # 对测试集样本进行预处理

r=1/max(abs(wine[samp,1:11])) # 确定参数rang的变化范围

n=length(samp)

err1=0

err2=0

for(i in 1:17)

{

set.seed(111)

model=nnet(quality~.,data=wine,maxit=400,rang=r,size=i,subset=samp,decay=5e-4)

err1[i]=sum(predict(model,wine[samp,1:11],type='class')!=wine[samp,12])/n

err2[i]=sum(predict(model,wine[-samp,1:11],type='class')!=wine[-samp,12])/(nrow.wine -n)

}

plot(1:17,err1,'l',col=1,lty=1,ylab="模型误判率",xlab="隐藏层节点个数",ylim=c(min(min(err1),min(err2)),max(max(err1),max(err2))))

lines(1:17,err2,col=1,lty=3)

points(1:17,err1,col=1,pch="+")

points(1:17,err2,col=1,pch="o")

legend(1,0.53,"测试集误判率",bty="n",cex=1.5)

legend(1,0.35,"训练集误判率",bty="n",cex=1.5)

###确定训练周期

err11=0

err12=0

for(i in 1:500)

{

set.seed(111)

model=nnet(quality~.,data=wine,maxit=i,rang=r,size=3,subset=samp)

err11[i]=sum(predict(model,wine[samp,1:11],type='class')!=wine[samp,12])/n

err12[i]=sum(predict(model,wine[-samp,1:11],type='class')!=wine[-samp,12])/(nrow.wine-n)

}

plot(1:length(err11),err11,'l',ylab="模型误判率",xlab="训练周期",col=1,ylim=c(min(min(err11),min(err12)),max(max(err11),max(err12))))

lines(1:length(err11),err12,col=1,lty=3)

legend(250,0.47,"测试集误判率",bty="n",cex=1.2)

legend(250,0.425,"训练集误判率",bty="n",cex=1.2)

###最终模型

set.seed(111)

model=nnet(quality~.,data=wine,maxit=300,rang=r,size=3,subset=samp)

x=wine[-samp,1:11] # 确认需要进行预测的样本特征矩阵

pred=predict(model,x,type="class") # 根据模型model1对xt数据进行预测

table(wine[-samp,12],pred)

#####支持向量机#####

library(e1071) # 加载e1071软件包

###第一种格式建立模型

data(iris) # 获取数据集iris

model=svm(Species~.,data=iris) # 建立svm模型

###第二种格式建立模型

x=iris[,-5] # 提取iris数据中除第5列以外的数据作为特征变量

y=iris[,5] # 提取iris数据中的第5列数据作为结果变量

model=svm(x,y,kernel ="radial",gamma =if(is.vector(x)) 1 else 1/ncol(x)) # 建立svm模型

###对模型进行预测

x=iris[,1:4] # 确认需要进行预测的样本特征矩阵

pred=predict(model,x) # 根据模型model对x数据进行预测

pred[sample(1:150,8)] # 随机挑选8个预测结果进行展示

table(pred,y) # 模型预测精度展示

###实际建模过程中完整操作

attach(iris) # 将数据iris按列单独确认为向量

x=subset(iris,select=-Species) # 确定特征变量为数据iris中除去Species的其他项

y=Species # 确定结果变量为数据iris中的Species项

type=c("C-classification","nu-classification","one-classification")# 确定将要适用的分类方式

kernel=c("linear","polynomial","radial","sigmoid") #确定将要适用的核函数

pred=array(0,dim=c(150,3,4)) #初始化预测结果矩阵的三维长度分别为150，3，4

accuracy=matrix(0,3,4) #初始化模型精准度矩阵的两维分别为3，4

yy=as.integer(y) #为方便模型精度计算，将结果变量数量化为1，2，3

for(i in 1:3) { #确认i影响的维度代表分类方式

for(j in 1:4) { #确认j影响的维度代表核函数

pred[,i,j]=predict(svm(x,y,type=type[i],kernel=kernel[j]),x) #对每一模型进行预测

if(i>2){

accuracy[i,j]=sum(pred[,i,j]!=1) }

else{

accuracy[i,j]=sum(pred[,i,j]!=yy)}

}

}

dimnames(accuracy)=list(type,kernel) #确定模型精度变量的列名和行名

table(pred[,1,3],y) # 模型预测精度展示

###模型可视化

plot(cmdscale(dist(iris[,-5])),col=c("lightgray","black","gray")[as.integer(iris[,5])],pch= c("o","+")[1:150 %in% model$index + 1]) # 绘制模型分类散点图

legend(2,-0.8,c("setosa","versicolor","virginica"),col=c("lightgray","black","gray"),lty=1) # 标记图例

data(iris) #读入数据iris

model=svm(Species~., data = iris) #利用公式格式建立模型

plot(model,iris,Petal.Width~Petal.Length,fill=FALSE,symbolPalette=c("lightgray","black","grey"),svSymbol="+")

#绘制模型类别关于花萼宽度和长度的分类情况

legend(1,2.5,c("setosa","versicolor","virginica"),col=c("lightgray","black","gray"),lty=1) #标记图例

###模型进一步优化

wts=c(1,1,1) # 确定模型各个类别的比重为1：1：1

names(wts)=c("setosa","versicolor","virginica") #确定各个比重对应的类别

model1=svm(x,y,class.weights=wts) #建立模型

wts=c(1,100,100) # 确定模型各个类别的比重为1：100：100

names(wts)=c("setosa","versicolor","virginica") #确定各个比重对应的类别

model2=svm(x,y,class.weights=wts) #建立模型

pred2=predict(model2,x) #根据模型进行预测

table(pred2,y) #展示预测结果

wts=c(1,500,500) # 确定模型各个类别的比重为1：500：500

names(wts)=c("setosa","versicolor","virginica") #确定各个比重对应的类别

model3=svm(x,y,class.weights=wts) #建立模型

pred3=predict(model3,x) #根据模型进行预测

table(pred3,y) #展示预测结果