Shared Nearest Neighbours (sNN)

**Algorithm**

matchCol <- function(x,y,nbr,t){

j=0

for(i in 1:ncol(nbr)){

if((nbr[x,i] == nbr[y,i]) && nbr[x,i]==TRUE){

j = (j+1)

}

}

if(j>=t){

return (TRUE)

}

else{

return (FALSE)

}

}

mySNN <- function(x,eps,minPts,srdPts){

#no. of rows cols

row <- nrow(x)

# all distances

distance <- as.matrix(dist(x,"euclidean"))

# neighbour[i,j] contains 1 if j is in neighborhood of i, OR 0

neighbour <- matrix(rep(FALSE,row\*row),nrow=row,ncol=row)

# cluster[i] contains in which cluster record i belongs

cluster <- rep(1,row)

# status[i] contains 0,1,2 if i is..

# 1 if Core point

# 2 if Border point

# 0 if Noise point

status <- rep(0,row)

# find core points

# core point vector

corepts<-c()

for(i in 1:row){

count = 0

for(j in 1:row){

if(i != j && distance[i,j] <= eps){

neighbour[i,j] = TRUE

count = count + 1

}

}

if(count >= minPts){

status[i] <- 1

corepts = union(corepts,i)

}

}

if(length(corepts)==0){

return (cat("No core point found!!\nChoose eps,minPts properly."))

}

# mark border point

for(i in corepts){

for(j in 1:row){

if(neighbour[i,j] && status[j] != 1){

status[j] = 2

}

}

}

# merge clusters

stack = c()

unvisited = corepts

cl=2

while(length(unvisited)!=0){

#push 1st of unvisited

stack = union(stack,unvisited[1])

unvisited = unvisited[-1]

while(length(stack)!=0){

#pop last into temp1

temp1 = stack[length(stack)]

stack = stack[-length(stack)]

for(i in unvisited){

if(neighbour[temp1,i] && i!=temp1 && matchCol(temp1,i,neighbour,srdPts)){

stack = union(stack,i)

unvisited = unvisited[-match(i,unvisited)]

# 1st point

cluster[temp1] = cl

for(j in 1:length(cluster)){

if(status[j]==2 && neighbour[temp1,j]){

cluster[j] = cl

}

}

# 2nd point

cluster[i] = cl

for(j in 1:length(cluster)){

if(status[j]==2 && neighbour[i,j]){

cluster[j] = cl

}

}

}

}

}

cl = cl + 1

}

#print(distance)

#print(neighbour)

cat("\nSTATUS\n")

print(status)

cat("\nCore Points\n")

print(corepts)

cat("\nCLUSTER\n")

print(cluster)

return (cluster)

}

**Time Complexity**

O(n2)

**Space Complexity**

O(n2)

Where, n = total records

**Example**

y<-iris[,c(1,2)]

res=mySNN(y,0.3,6,4)

plot(y,col=res,pch=res,main="sNN for eps=0.3,minPts=6,srdPts=4")

**Output**

