Codes

March 24, 2023

1 An Example Run on the Libras Dataset

Load the libraries

[2]:

```
# Libraries
library(FastKNN)
library(dynamicTreeCut)
library(sparcl)
library(igraph)
```

Run the code

[3]:

```
#Code
BFfzero=function (f, a, b, num = 10, eps = 1e-05)
{
  h = abs(b - a)/num
  i = 0
  j = 0
  a1 = b1 = 0
  while (i <= num) {
    a1 = a + i * h
    b1 = a1 + h
    if (f(a1) == 0) {
      return(a1)
      #print(f(a1))
    }
    else if (f(b1) == 0) {
      return(b1)
      #print(f(b1))
    }
    else if (f(a1) * f(b1) < 0) {
      repeat {
        if (abs(b1 - a1) < eps)
          break
        x <- (a1 + b1)/2
        if (f(a1) * f(x) < 0)
          b1 <- x
        else a1 <- x
      }
      #print(j + 1)
      j = j + 1
      return((a1 + b1)/2)
      #print(f((a1 + b1)/2))
    }
    i = i + 1
  }
  if (j == 0)
    print("finding root is fail")
  else print("finding root is successful")
}
S=function(x,y){
  if(x>y){
    return(x-y)
  }else{return(0)}
}
diff_sq_calc=function(x,W){
  n=length(x)
  s=0
  for( i in 1:n){
    for( j in 1:n){
      s=s+W[i,j]*(x[i]-x[j])^2
    }
  }
  return(s)
}
d_w=function(x,y,w){
  return(sum(w*(x-y)^2))
}
solve_alpha=function(A,lambda){
  f=function(alpha){
    s=0
    p=length(A)
    for(i in 1:p){
      s=s+S(alpha/A[i],lambda)
    }
    return(s/2-1)
  }
  a=BFfzero(f,0.01,1000)
  return(a)
}
bcc=function(X,M,beta=2,lambda,gamma=1000,k=5,tmax=30){
  n=dim(X)[1]
  p=dim(X)[2]
  #w=rep(1/p,p)
  w=runif(p)
```

```
w=w/sum(w)
A=numeric(p)
B=numeric(p)
C=numeric(p)
for(t in 1:tmax){
  D=matrix(0,n,n)
  for(i in 1:n){
    for(j in 1:n){
      D[i,j]=d_w(X[i,],X[j,],w^2+lambda*w)
    }
  }
  nn = matrix(0,n,k) # n x k
  for (i in 1:n)
    nn[i,] = k.nearest.neighbors(i, D, k )
  W=matrix(0,n,n)
  for(i in 1:n){
    for(j in nn[i,]){
      W[i,j]=exp(-0.5*D[i,j]/p)
    }
  }
  for(i in 1:n){


    for(l in 1:p){
      if(w[l]>0){
       ␣
↪divider=(sum(W[i,nn[i,]])+sum(W[nn[i,],i]))*gamma+w[l]^beta+lambda*abs(w[l])
       ␣
↪M[i,l]=(gamma*(sum(W[i,nn[i,]]*M[nn[i,],l])+sum(W[nn[i,],i]*M[nn[i,],l]))
                +w[l]^beta*X[i,l]+lambda*abs(w[l])*X[i,l])/dividor
        #    cat(c(i,l))
        #   cat('\n')
      }else{
        M[i,l]=0
      }
    }
  }

  for( l in 1:p){
    A[l]=sum((X[,l]-M[,l])^2)
    #B[l]=gamma*diff_sq_calc(M[,l],W)
    #C[l]=A[l]
  }
  alpha=solve_alpha(A,lambda)
  for(l in 1:p){
    w[l]=(S(alpha/A[l],lambda)/beta)^(1/(beta-1))
  }
```

```
    #cat(t)
    #cat('\t')
    #cat(w)
    #cat('\n')
    #    points(M,pch=19,col=t+1)
  }

  return(list(M,w))
}
label_orientation=function(label){
  m=length(label)
  u=unique(label)
  u=sort(u)
  n=length(u)
  u1=numeric(m)
  for(i in 1:n){
    I=which(label==u[i])
    u1[I]=i
  }
  return(u1)
}
```

load the data

[4]:

```
X=read.csv('movement_libras.csv',head=FALSE)
X=data.matrix(X)
toss=X[,91]
X=X[,-91]
I3=which(toss==3)
I4=which(toss==4)
I5=which(toss==5)
I7=which(toss==7)
I11=which(toss==11)
I12=which(toss==12)
I=c(I3,I4,I5,I7,I11,I12)
X=X[I,]
toss=toss[I]
p=dim(X)[2]
for(i in 1:p){
  X[,i]=(X[,i]-mean(X[,i]))/sd(X[,i])
}
```

execute the bcc code

[5]:

```
l=bcc(X,X,lambda=0.002,gamma=10,tmax=100)
```

[6]:

```
h=hclust(dist(l[[1]]),method = 'average')
```

```
[7]: c=cutreeDynamic(h,distM = as.matrix(dist(l[[1]])))
```

```
 ..cutHeight not given, setting it to 11.6  ===>  99% of the (truncated) height
range in dendro.
 ..done.
```

Compare with ground truth

```
[8]: compare(toss,c,'adjusted.rand')
```

0.782284779108184

```
[9]: compare(toss,c,'nmi')
```

0.894883438728016

```
[ ]:
```