MATH 277 PROJECT - Comparison of Spectral Clustering and Kernighan Lin Algorithm in Planted Bisection over their Consistency

Hao, Luyan

2019-3-19

Problem Part 1.

Generating a Planted Bisection Model by a function PlantedBisection

```
# 2n <- the size of the total vertices, n for each class
# p <- the probability of an edge existing within same class
# q <- the probability of an edge existing between classes
PlantedBisection <- function(n, p_within, q_between) {
# generating a random sorted sequence of vertices
  v = 1: (2*n)
  v = sample(v,size = 2*n,replace = FALSE,prob = NULL)
  v_class = matrix(data = NA,nrow = n, ncol = 2)
# put our vertices into two classes
  for (i in 1:n) {
  v_class[i,1] = v[i]
  v_class[i,2] = v[i+n]
  }
# Adding edges to vertices based on p and q, storing information into a matrix w
  w = matrix(0,nrow = (2*n), ncol = (2*n))
  for (i in 1:(2*n-1)) {
    for (j in (i+1):(2*n)) {
      for (k in 1:2) {
        if ((i %in% v_class[,k]) && (j %in% v_class[,k])) {
          w[i,j]= runif(1) < p_within
        }
      }

      if(w[i,j] == 0){
        w[i,j] <- (runif(1) < q_between)
      }else
      {
        w[i,j] <- (runif(1) < p_within)
      }
    }
}

   w = w + t(w)
    result = list(v = v, v_class = v_class, w = w)
    return(result)
}
```

Solving a Planted Bisection Problem by a function SpectralClustering

```
SpectralClustering <- function(w,n) {

  D = matrix(data = 0,nrow = 2*n,ncol = 2*n)
  inverseOfSqrtD = matrix(data = 0,nrow = 2*n,ncol = 2*n)
  L = matrix(data = 0,nrow = 2*n,ncol = 2*n)

  for(i in 1:(2*n)){
   for(j in 1:(2*n)){
     D[i,i] = D[i,i] + w[i,j]}
    inverseOfSqrtD[i,i] =  sqrt(1/D[i,i])
    }

  L = inverseOfSqrtD %*% w %*% inverseOfSqrtD
  L = diag(2*n) - L

  lamda = eigen(L,symmetric = TRUE)$values[(2*n)-1]
  eigenVector = eigen(L,symmetric = TRUE)$vectors[,(2*n)-1]
  classedV1 = vector()
  classedV2 = vector()
  middlePoint = median(eigenVector)
    for (i in 1:(2*n)) {
      if (eigenVector[i] < middlePoint){
        classedV1 = c(classedV1,i)
      } else {
        classedV2 = c(classedV2,i)
      }
    }
  result = list(c1 = classedV1,c2 = classedV2,eigenValue = lamda,eigenVector = eigenVector)
  return(result)
}
```

Plotting our results of the function SpectralClustering

```
SpectralClusteringPlotting <- function(w,n){
  par(mfrow=c(1,1), pty = "s")
  image(x=c(1:(2*n)), y=c(1:(2*n)), w,col=c("white","pink"), xlab = 'Index of Verticies', ylab = 'Index of
  lines(x = c(1:(2*n)),y=c(1:(2*n)), col ='grey')
  result = SpectralClustering(w,n)

  v2Small = result$eigenVector
  v2SmallSort =  sort(v2Small)

  or = order(v2Small)
  reOrderMatrix = w[or,or]
  par(mfrow=c(1,1), pty = "s")
  image(x=c(1:(2*n)), y=c(1:(2*n)), reOrderMatrix,col=c("white","pink"), xlab = 'Index of Verticies', ylab
  lines(x = c(1:(2*n)),y=c(1:(2*n)), col ='grey')
  par(mfrow=c(1,1), pty = "s")
  plot(v2SmallSort,pch=16, ylab = 'Second Smallest Eigenvector Plot')
}
```

Define a function to calculate the distance between two partitions

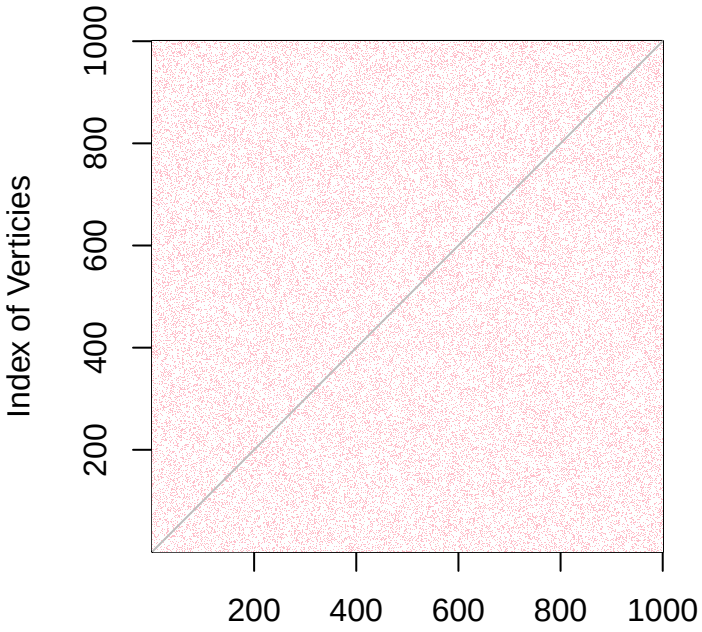
```
dist <- function(P,Q){
  k <- length(P)
  n <- sum(lengths(P))
  m <- length(Q)
  d <- 0
  library(dplyr)
  t <- matrix(0,nrow=n,ncol=n)
  for (j in 1:m){
    q <- data.frame(Q[[j]])
    colnames(q) <- c(1)
    for (i in 1:k) {
      p <- data.frame(P[[i]])
      colnames(p) <- c(1)
      t[i,j] <- count(inner_join(p, q))[[1]]
      if(i==1){
        u=i
      } else if(t[i,j] >= t[i-1,j]){
        u = i
      }
    }
    n <- sum(lengths(Q[[j]]))
    d <- t[u,j]/n + d
  }
  d/m
}
```

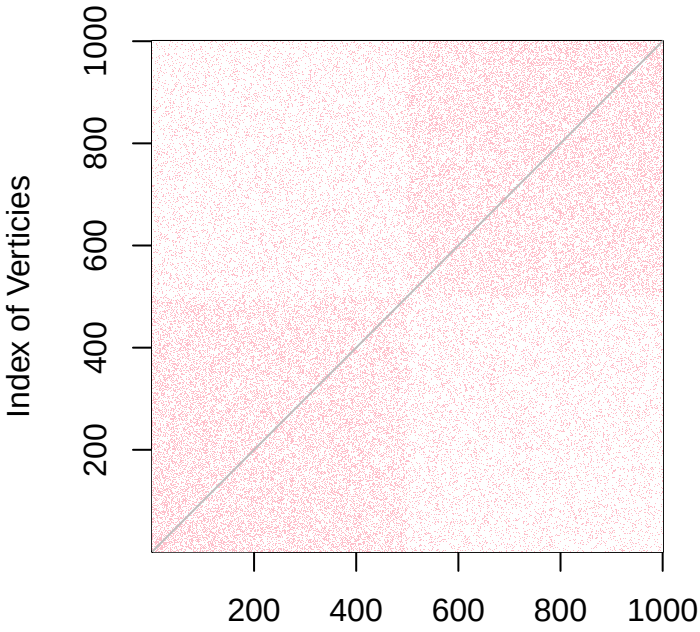
Testing the performance of our algorithm

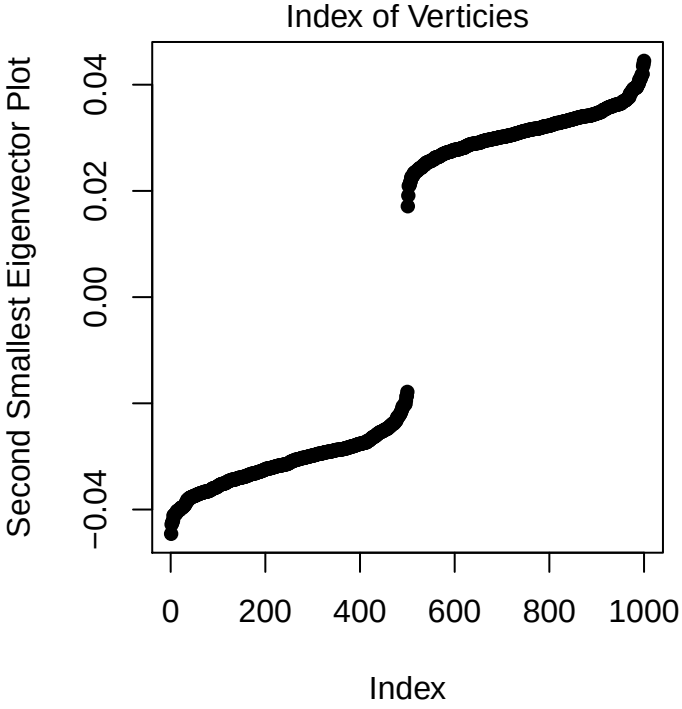
```
performanceTest <- function(n,p_within, q_between,rep){
  d = 0
  for(i in 1:rep)
  {
    plantedBisection = PlantedBisection(n,p_within,q_between)
    w = plantedBisection$w
    v_class = plantedBisection$v_class
    output = SpectralClustering(w,n)
    P = list(output$c1,output$c2)
    Q = list(v_class[,1],v_class[,2])
    d = d + dist(P,Q)
  }
  d = d/rep
  return(d)
}
```

Testing the PLantedBisection

```
n=500
p_within = 0.5
q_between = 0.15
w = PlantedBisection(n, p_within, q_between)$w
SpectralClusteringPlotting(w,n)
```








```
#rep = 20
#d = performanceTest(n,p_within,q_between,rep)
#d
```