Hw4_code

Eric Chuu

November 30, 2015

Problem 2: Serial Ranking Algorithm

```
# read in file for 2009, 2010 names, to matrix
names <- as.matrix(read.csv("England_2009_2010_TeamNames.csv", header = FALSE))

# read in file for 2009, 2010 scores, to matrix
C <- as.matrix(read.csv("England_2009_2010.csv", header = FALSE))

# team loses both games => C_ij = -1
# team wins both games  => C_ij =  1
# otherwise             => C_ij =  0

# implement Serial-Rank algorithm
# diagonal of C is all 1s
diag(C) <- 1
S <- matrix(NA, 20, 20)

# final similarity pairwise similarity matrix S, dimensions: 20x20
one_mat <- matrix(1, 20, 20)
S <- 1/2 * (20 * one_mat %*% t(one_mat) + C %*% t(C))

# Laplacian matrix: L = D - S
D_ii <- diag(S %*% one_mat)
D <- D_ii * diag(20)

L <- matrix(NA, 20, 20)
L <- D - S

# get eigenvalues from L
L_evals <- eigen(L)$values
L_evecs <- eigen(L)$vectors

# compute fiedler vector of S
# we take the 19th eigenvector since the 20th eigenvalue is ~ 0
fiedler_vector <- L_evecs[,19]
# sort fiedlier vector in increasing order
fiedler_vector_inc <- sort(fiedler_vector)

# minimize upsets in the final ranking
team_rank <- matrix(NA, 20, 1)

for(i in 1:20){
  for(j in 1:20){
    if(fiedler_vector_inc[i] == fiedler_vector[j])
      team_rank[i,1] <- names[j]
  }
```

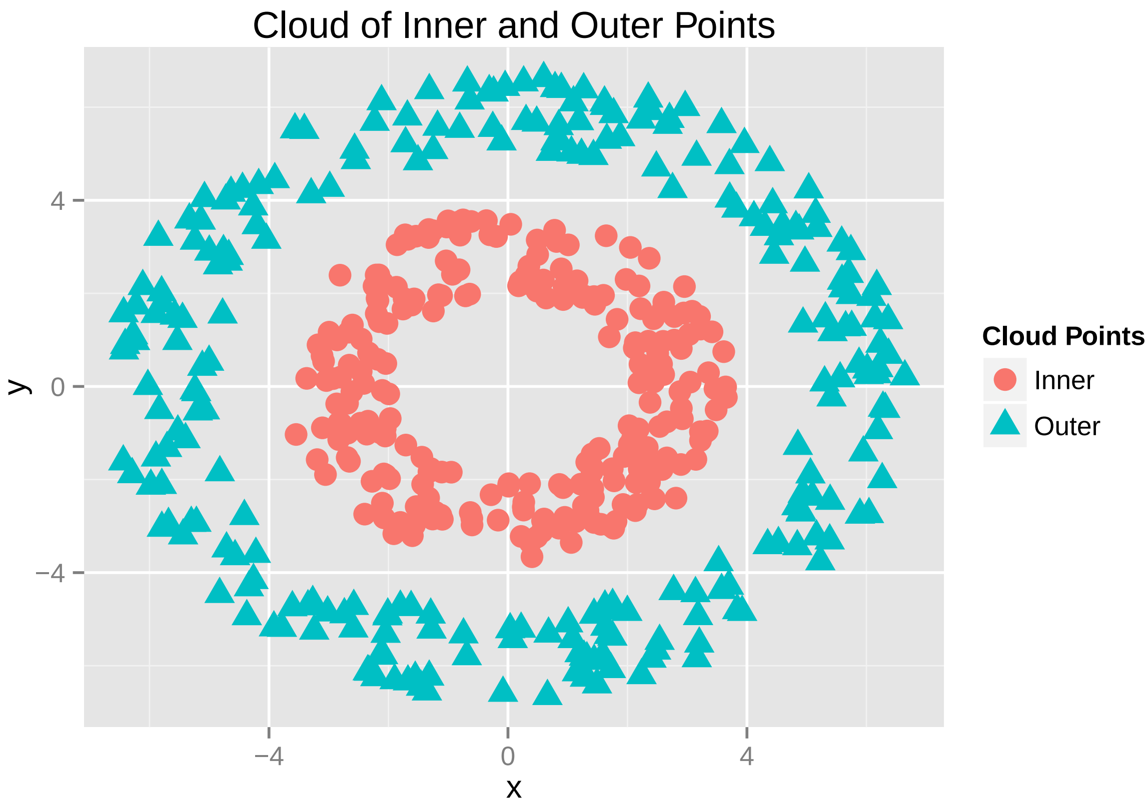
```r
}

# team ranking result
team_rank
```

```
##       [,1]
##  [1,] "'Man City'"
##  [2,] "'Chelsea'"
##  [3,] "'Man United'"
##  [4,] "'Tottenham'"
##  [5,] "'Arsenal'"
##  [6,] "'Everton'"
##  [7,] "'Aston Villa'"
##  [8,] "'Liverpool'"
##  [9,] "'Blackburn'"
## [10,] "'Fulham'"
## [11,] "'Stoke'"
## [12,] "'Bolton'"
## [13,] "'Birmingham'"
## [14,] "'West Ham'"
## [15,] "'Sunderland'"
## [16,] "'Wolves'"
## [17,] "'Portsmouth'"
## [18,] "'Wigan'"
## [19,] "'Hull'"
## [20,] "'Burnley'"
```

Problem 3: Diffusion Maps

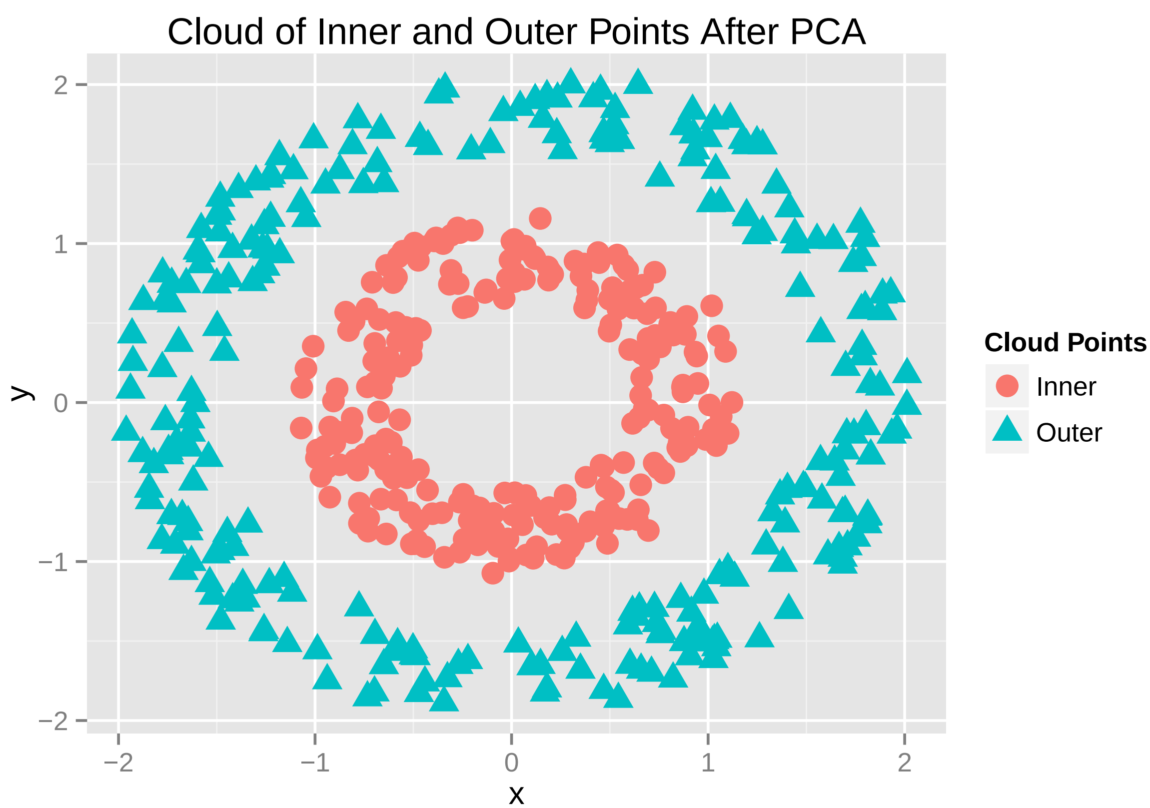
```r
library(ggplot2)
load("twoCircles.data")
# object A contains n = 500 points of 2-d coordinates
# inner points: first 250 points
# outer points: second 250 points

### (a) plot point cloud, mark 2 different kind of plots with two different symbols
inner <- data.frame(A[1:250,], point = rep("inner", 250))
outer <- data.frame(A[251:500,], point = rep("outer", 250))

in_out <- rbind(inner, outer)

pt_cloud <- ggplot(in_out, aes(x = x, y = y, group = point,
                               shape = point, colour = point)) +
  geom_point(size = 4) +
  scale_colour_discrete(name = "Cloud Points",
                        breaks = c("inner", "outer"),
                        labels = c("Inner", "Outer")) +
  scale_shape_discrete(name = "Cloud Points",
                       breaks = c("inner", "outer"),
                       labels = c("Inner", "Outer")) +
  ggtitle("Cloud of Inner and Outer Points")
```

```
# inner points are red circles, outer points are blue triangles
pt_cloud
```



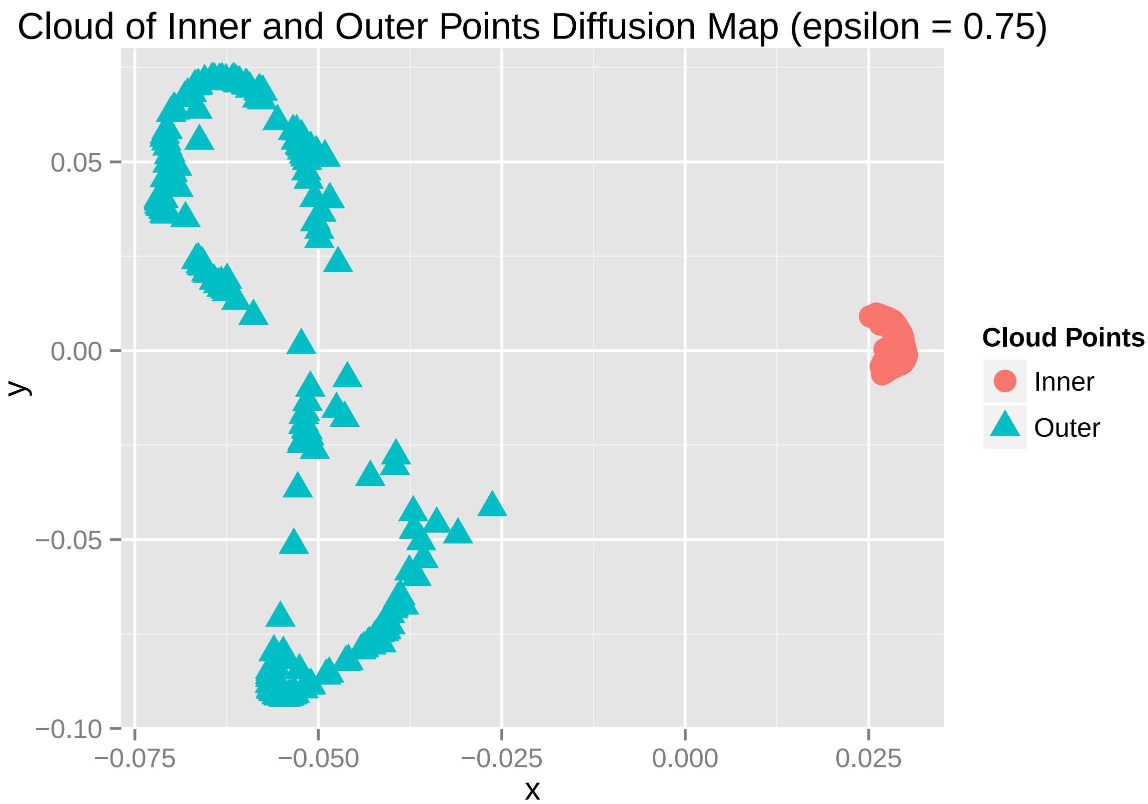

```
### (b) perform PCA on this data set
# show a 2-d plot of your data points using the 2 principal components
# plot inner and outer points again with distinct symbols

# performs PCA and returns top 2 eigen vectors
run_PCA = function(stock_mat){
  myPCA = prcomp(stock_mat, scale. = TRUE)
  # top 5 eigen vectors
  e_vecs = myPCA$x[,1:2]

  # print variance capture
  #pca_summ <- summary(myPCA)
  #variance_capture <- pca_summ$importance[3,5]

  return(e_vecs)
}

eigen_vectors <- run_PCA(in_out[-3])
# store the top 2 eigenvectors back into the inner, outer data frame
in_out[,1:2] <- eigen_vectors

# replot the points
```

```r
pt_cloud <- ggplot(in_out, aes(x = x, y = y, group = point,
                               shape = point, colour = point)) +
  geom_point(size = 4) +
  scale_colour_discrete(name = "Cloud Points",
                        breaks = c("inner", "outer"),
                        labels = c("Inner", "Outer")) +
  scale_shape_discrete(name = "Cloud Points",
                       breaks = c("inner", "outer"),
                       labels = c("Inner", "Outer")) +
  ggtitle("Cloud of Inner and Outer Points After PCA")

pt_cloud
```




```r
### (c) implement diffusion maps algorithm

# compute matrix of all pairwise distances
# 500x500 matrix
DIST <- as.matrix(dist(A, method = "euclidean", diag = FALSE,
                       upper = TRUE, p = 2))
# L = D_inverse * W
epsilon = 0.75 # experiment with epsilon - 0.75, 1
W = exp(-DIST^2 / epsilon)
# D is a diagonal matrix with D_ii = sum(row i of W matrix)
D_ii = rowSums(W)
# make D_ii the diagonal entires of a 500x500 matrix
```

```r
D <- D_ii * diag(500)
D_inv <- solve(D)
L = D_inv %*% W

e_values <- eigen(L)$values   # 500 eigenvalues
e_vectors <- eigen(L)$vectors # 500x500 matrix, each column is an eigenvector

# take first two nontrivial eigenvectors
e_vec_1 <- e_values[2] * e_vectors[,2]
e_vec_2 <- e_values[3] * e_vectors[,3]

in_out[,1] <- e_vec_1
in_out[,2] <- e_vec_2
new_cloud <- ggplot(in_out, aes(x = x, y = y, group = point,
                                shape = point, colour = point)) +
  geom_point(size = 4) +
  scale_colour_discrete(name = "Cloud Points",
                        breaks = c("inner", "outer"),
                        labels = c("Inner", "Outer")) +
  scale_shape_discrete(name = "Cloud Points",
                       breaks = c("inner", "outer"),
                       labels = c("Inner", "Outer")) +
  ggtitle("Cloud of Inner and Outer Points Diffusion Map (epsilon = 0.75)")

# replot points after implementing diffusion map algorithm
new_cloud
```




```
# for epsilon = 1
# L = D_inverse * W
epsilon = 1 # experiment with epsilon - 0.75, 1
W = exp(-DIST^2 / epsilon)
# D is a diagonal matrix with D_ii = sum(row i of W matrix)
D_ii = rowSums(W)
# make D_ii the diagonal entires of a 500x500 matrix
D <- D_ii * diag(500)
D_inv <- solve(D)
L = D_inv %*% W

e_values <- eigen(L)$values   # 500 eigenvalues
e_vectors <- eigen(L)$vectors # 500x500 matrix, each column is an eigenvector

# take first two nontrivial eigenvectors
e_vec_1 <- e_values[2] * e_vectors[,2]
e_vec_2 <- e_values[3] * e_vectors[,3]

in_out[,1] <- e_vec_1
in_out[,2] <- e_vec_2

new_cloud <- ggplot(in_out, aes(x = x, y = y, group = point,
                                shape = point, colour = point)) +
  geom_point(size = 4) +
  scale_colour_discrete(name = "Cloud Points",
                        breaks = c("inner", "outer"),
```

```
                              labels = c("Inner", "Outer")) +
  scale_shape_discrete(name = "Cloud Points",
                       breaks = c("inner", "outer"),
                       labels = c("Inner", "Outer")) +
  ggtitle("Cloud of Inner and Outer Points Diffusion Map (epsilon = 1)")

# replot points after implementing diffusion map algorithm
new_cloud
```