STAT-S675

Homework 7

John Koo

Link to assignment

```
library(ggplot2)
dp <- loadNamespace('dplyr')
import::from(magrittr, `%>%`, `%<>%`)
import::from(ggrepel, geom_text_repel)
import::from(viridis, scale_colour_viridis)
import::from(readr, read_table)

theme_set(ggthemes::theme_base())

source('http://pages.iu.edu/~mtrosset/Courses/675/stress.r')
source('http://pages.iu.edu/~mtrosset/Courses/675/manifold.r')
```

Problem 1

Figure ??(a)

```
# parameters/constants
a <- sqrt(2 - 2 * cos(pi / 3))
b <- sqrt(2 + 2 * cos(pi / 3))

# construct the data
input.df <- dplyr::data_frame(i = seq(6)) %>%
  dp$mutate(theta = (i - 1) * pi / 3) %>%
  dp$mutate(x = cos(theta), y = sin(theta)) %>%
  # attach i+1, i+2, and i+3 for the edge weights
  dp$mutate(x.next = lead(x), y.next = lead(y),
            x.next.2 = lead(x, 2), y.next.2 = lead(y, 2),
            x.next.3 = lead(x, 3), y.next.3 = lead(y, 3))

# plot
figure.a <- ggplot(input.df) +
  coord_fixed() +
  scale_colour_distiller(palette = 'Spectral') +
  labs(x = NULL, y = NULL, colour = 'dissimilarity') +
  # edge weights
  geom_segment(aes(x = x, y = y,
                   xend = x.next, yend = y.next,
                   colour = a)) +
  geom_segment(aes(x = x, y = y,
                   xend = x.next.2, yend = y.next.2,
                   colour = b)) +
  geom_segment(aes(x = x, y = y,
                   xend = x.next.3, yend = y.next.3,
```

```
                    colour = 2)) +
  # vertices
  geom_point(aes(x = x, y = y)) +
  geom_text_repel(aes(x = x, y = y, label = i))

figure.a
```

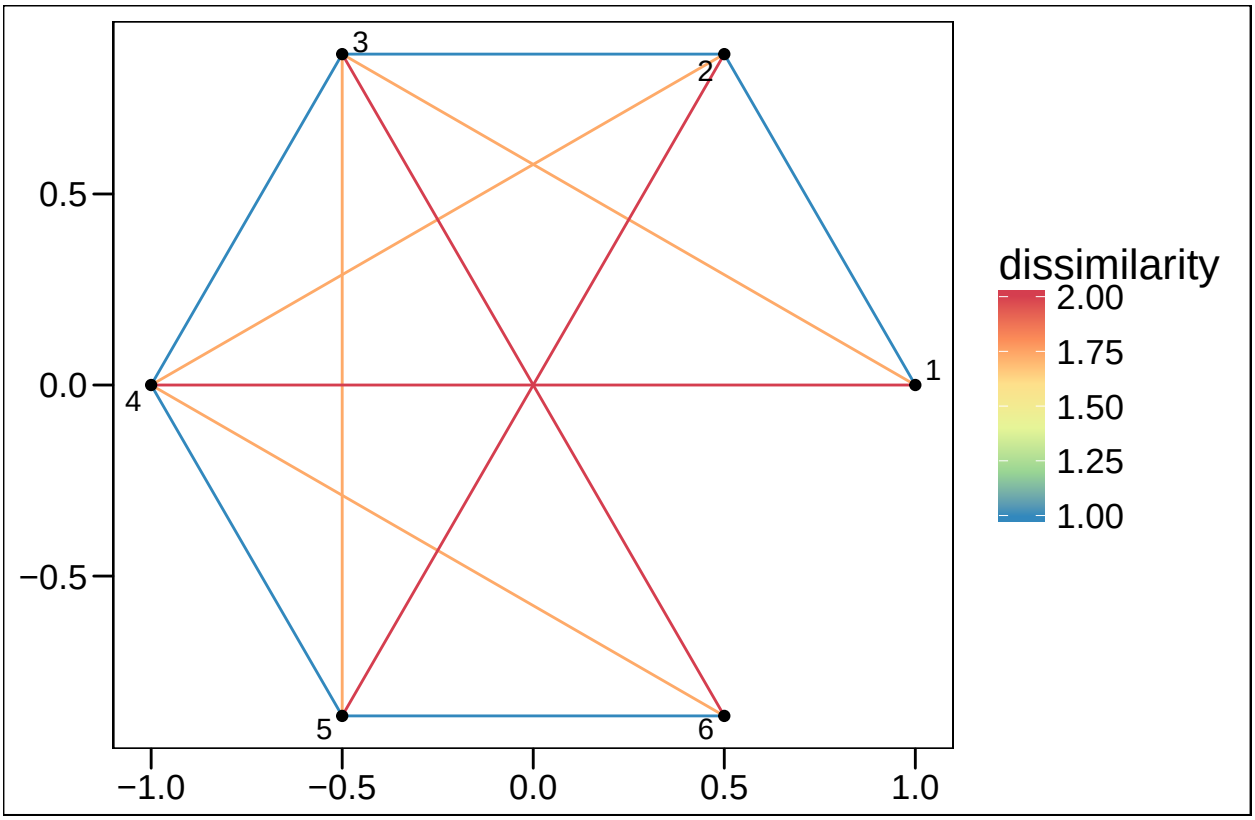
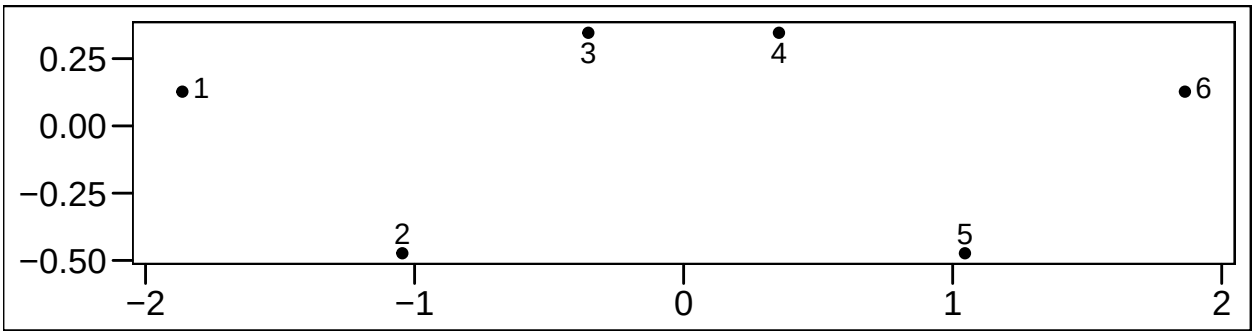


Figure ??(b)

```
# paramters/constants
N.iter <- 20

# specified dissimilarity matrix
Delta <- rbind(c(0, a, b, 2, a + 2, b + 2),
               c(a, 0, a, b, 2, a + 2),
               c(b, a, 0, a, b, 2),
               c(2, b, a, 0, a, b),
               c(a + 2, 2, b, a, 0, a),
               c(b + 2, a + 2, 2, b, a, 0))

# initialize the configuration using CMDS
X <- cmdscale(as.dist(Delta))
stress <- mds.stress.raw.eq(X, Delta)
gma.df <- as.data.frame(X) %>%
  dp$transmute(id = seq(6), x = V1, y = V2, iter = 0, stress)
```

```
# GMA optimization
for (i in seq(N.iter)) {
  # iterate
  X <- mds.guttman.eq(X, Delta)
  # compute stress
  stress <- mds.stress.raw.eq(X, Delta)
  # compile into data frame
  temp.df <- as.data.frame(X) %>%
  dp$transmute(id = seq(6), x = V1, y = V2, iter = i, stress)
  # attach data frame to original
  gma.df %<>% dp$bind_rows(temp.df)
}

# plot final configuration
gma.df %>%
  dp$filter(iter == N.iter) %>%
  ggplot() +
  labs(x = NULL, y = NULL) +
  coord_fixed() +
  geom_point(aes(x = x, y = y)) +
  geom_text_repel(aes(x = x, y = y, label = id))
```

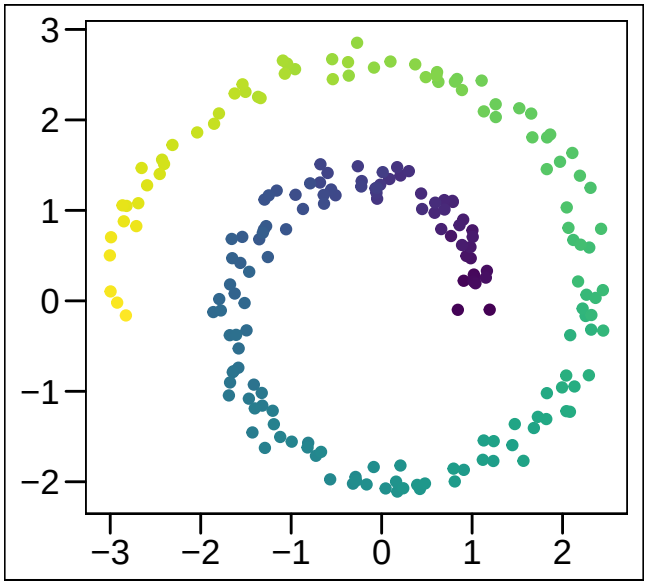


Problem 2

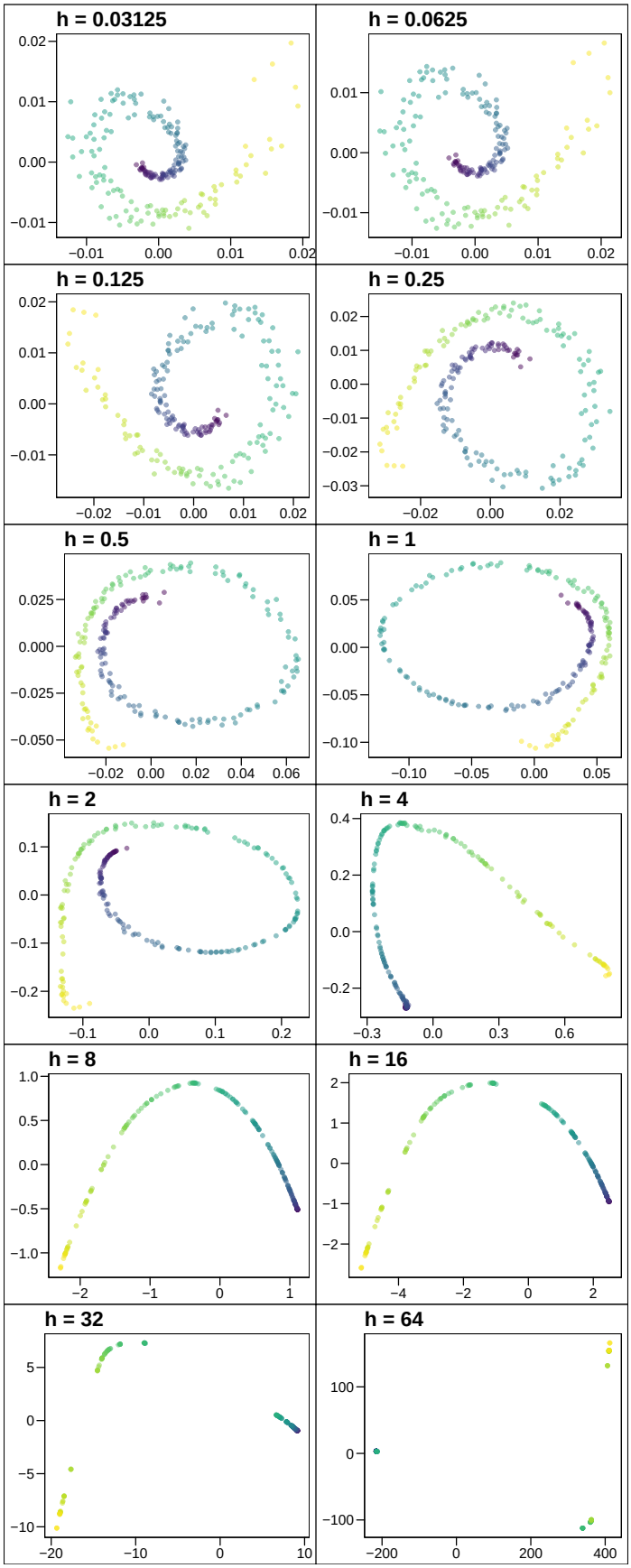
[Exercise 6.8.1 from the text]

```
# load data
spiral.df <- read_table('http://pages.iu.edu/~mtrosset/Courses/675/X.spiral',
                        col_names = FALSE)

# save the number of rows of the data
# for when we want the 2 smallest eigenvalues
n <- nrow(spiral.df)

# plot the data
spiral.df %>%
  dp$mutate(id = as.numeric(rownames(.))) %>%
  ggplot() +
  viridis::scale_colour_viridis() +
  geom_point(aes(x = X1, y = X2, colour = id)) +
  coord_fixed() +
```

```
  theme(legend.position = 'none') +
  labs(x = NULL, y = NULL)
```




```
# values of h to try
h.vector <- 2 ** seq(-5, 6)

# construct plots
eigenmaps <- lapply(h.vector, function(h) {
  # similarity matrix
  W <- exp(-h * as.matrix(dist(spiral.df)) ** 2)

  L <- graph.laplacian(W)

  L.eigen <- eigen(L)

  eigenmap <- cbind(L.eigen$vectors[, n - 1] / sqrt(L.eigen$values[n - 1]),
                    L.eigen$vectors[, n - 2] / sqrt(L.eigen$values[n - 2])) %>%
    as.data.frame() %>%
    dp$mutate(id = as.numeric(rownames(.))) %>%
    ggplot() +
    geom_point(aes(x = V1, y = V2, colour = id),
               alpha = .5) +
    viridis::scale_colour_viridis() +
    labs(x = NULL, y = NULL, title = paste('h =', h)) +
    theme(legend.position = 'none')
  return(eigenmap)
})

# construct plot
.gridarrange <- function(...) gridExtra::grid.arrange(..., ncol = 2)
do.call(.gridarrange, eigenmaps)
```


h = 0.03125
h = 0.0625
h = 0.125
h = 0.25
h = 0.5
h = 1
h = 2
h = 4
h = 8
h = 16
h = 32
h = 64

We can see that very small values of h result in a Laplacian eigenmap that preserves the original structure, and increasing values of h unravels the spiral up to a point. Once the spiral is unraveled, the points along the curve start to separate into three clusters, which seem to correspond to the "weak points" of the original spiral.

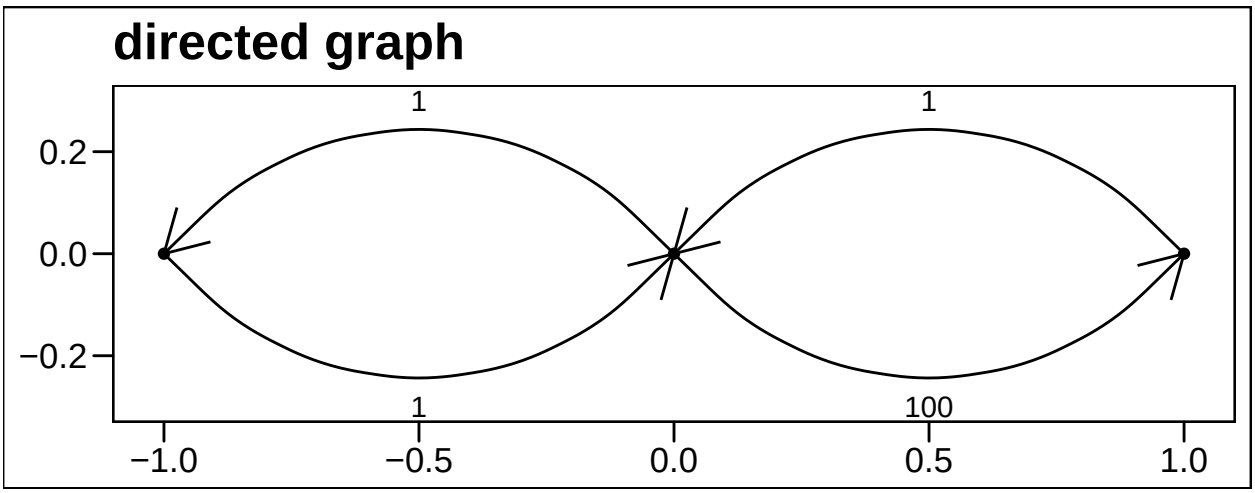
If we want an approximate representation of the original data in one dimension, $h \approx 8$ seems like the best choice.

An attempt at explaining what's going on

When h is very small, the similarity between any two points is close to 1 since $lim_{h \to 0} \exp(-h d_{ij}^2) = \exp(0) = 1$. On the other hand, when $h \to \infty$, the similarities go to 0. In addition, $\exp(-x)$ is approximately linear around $x = 0$. So when h is very small, we just have $\gamma_{ij} \sim -h d_{ij}^2 \ \forall i, j$. On the other hand, when h is very large, we have this relationship for only when d_{ij}^2 is very small, with $\gamma_{ij} \approx 0$ otherwise. So it would seem that large choices for h are the most appropriate. However, since there is some noise in the data, not all points are equally spaced along the spiral. The gaps along the spiral, which are really just random errors, become more pronounced as h increases.

Problem 3

```
# construct the graph
epsilon <- .3
x1 <- c(1, 0)
x2 <- c(0, 0)
x3 <- c(-1, 0)
ggplot() +
  labs(x = NULL, y = NULL, title = 'directed graph') +
  geom_point(aes(x = x1[1], y = x1[2])) +
  geom_point(aes(x = x2[1], y = x2[2])) +
  geom_point(aes(x = x3[1], y = x3[2])) +
  geom_curve(aes(x = x1[1], y = x1[2], xend = x2[1], yend = x2[2]),
               arrow = arrow()) +
  geom_text(aes(x = .5, y = epsilon, label = 1)) +
  geom_curve(aes(x = x2[1], y = x2[2], xend = x1[1], yend = x1[2]),
             arrow = arrow()) +
  geom_text(aes(x = .5, y = -epsilon, label = 100)) +
  geom_curve(aes(x = x2[1], y = x2[2], xend = x3[1], yend = x3[2]),
             arrow = arrow()) +
  geom_text(aes(x = -.5, y = epsilon, label = 1)) +
  geom_curve(aes(x = x3[1], y = x3[2], xend = x2[1], yend = x2[2]),
             arrow = arrow()) +
  geom_text(aes(x = -.5, y = -epsilon,
                label = 1)) +
  coord_fixed()
```



We can find the ECT matrix for this graph using the method outlined in the text:

```
diag.matrix <- function(m) {
  # diagonal matrix of a matrix
  as.matrix(Matrix::Diagonal(nrow(m), diag(m)))
}

ECT <- function(W) {
  # find the matrix of expected commute times for weight matrix W

  # size of W (i.e., number of vertices)
  n <- nrow(W)

  # 1 vector
  e <- rep(1, n)

  # volume of graph
  w.tilde <- sum(W)

  # transition probabilities
  pi. <- (W %*% e) / w.tilde

  # total weight (and inverse)
  T <- Matrix::Diagonal(n, W %*% e)
  T.inv <- Matrix::Diagonal(n, (W %*% e) ** -1)

  # matrix of transition probabilities
  P <- T.inv %*% W

  # fundamental matrix of kemeny and snell
  Z <- solve(diag(n) - P + e %*% t(pi.))
  D <- w.tilde * T.inv

  # matrix of first passage times
  M <- as.matrix((diag(n) - Z + e %*% t(e) %*% diag.matrix(Z)) %*% D)

  # matrix of expected commute times
  C. <- (M - diag.matrix(M)) + t(M - diag.matrix(M))
```

```
  return(C.)
}

# construct W for the graph
W <- rbind(c(0, 1, 0),
           c(100, 0, 1),
           c(0, 1, 0))

# find ECT for the graph
ECT(W)
```

```
         [,1]     [,2] [,3]
[1,]   0.0000  52.5198  206
[2,]  52.5198   0.0000  103
[3,] 206.0000 103.0000    0
```

Expected commute distances are guaranteed to be EDM-2. On the other hand, we know that this can't be EDM-1. If we think of the vertices as vertices of a triangle, since the length of the side from vertex 1 to vertex 3 is greater than the sum of the lengths of the sides from 1 to 2 and from 2 to 3, the triangle can't quite be formed.