Project 1 - Kernel PCA

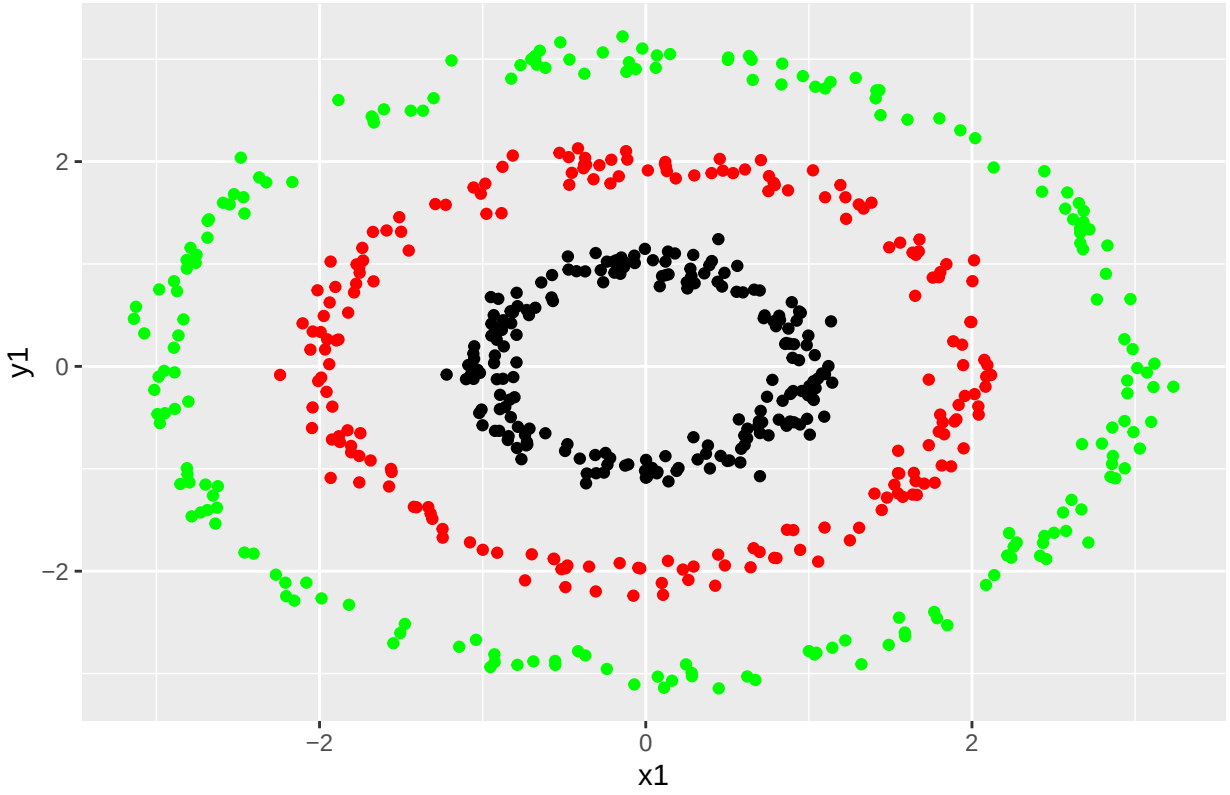
August Jonasson

2024-10-04

Task A

The first task is to simulate three concentric circles. This is done sampling from a uniform distribution in combination with some normally distributed noise. The (mean) radius of the circles are 1, 2 and 3 respectively, which is evident from Figure 1.

Figure 1: Concentric circles plot (simulated)



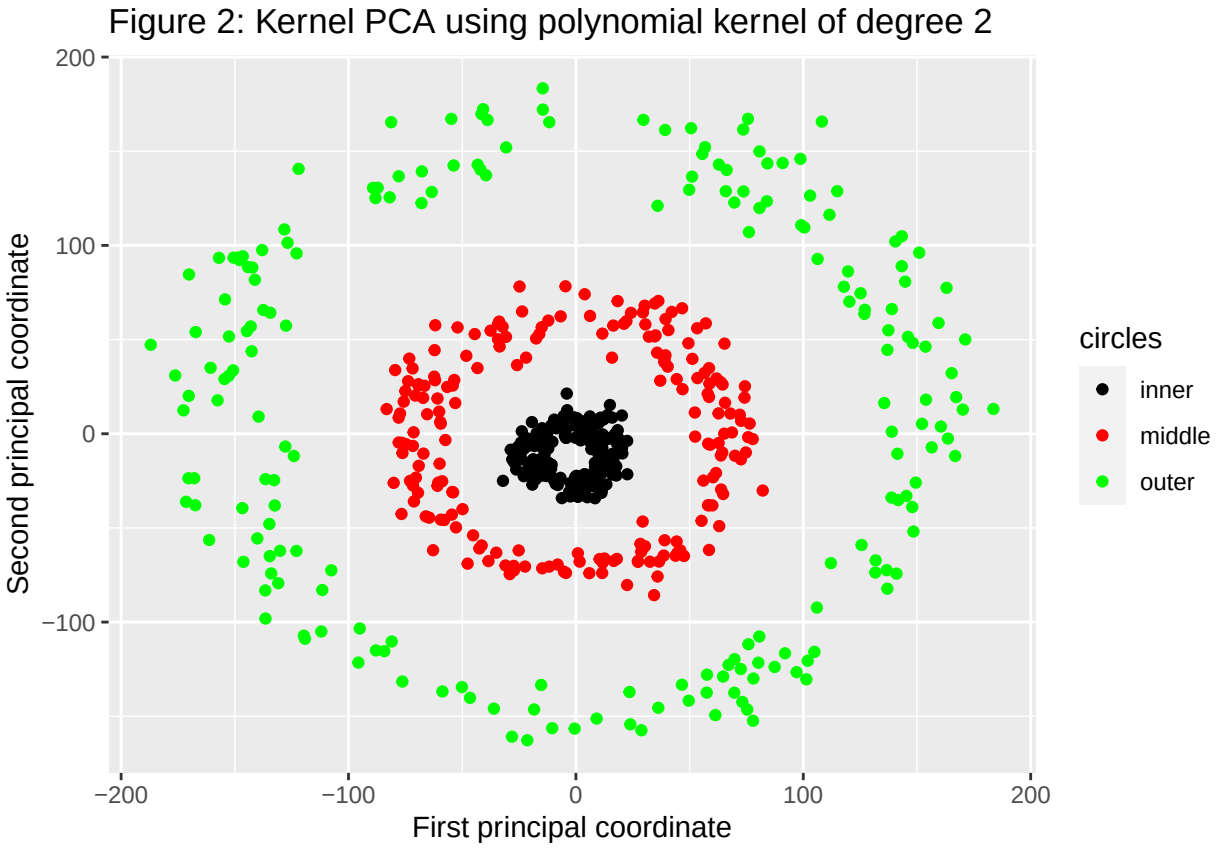
Task B

In the second task we are asked to perform kernel PCA on the dataset, using a polynomial kernel of degree 2. We define this kernel in the following way

$$< \phi(y_i) \cdot \phi(y_j) >= (1 + < y_i \cdot y_j >)^2 , \tag{1}$$

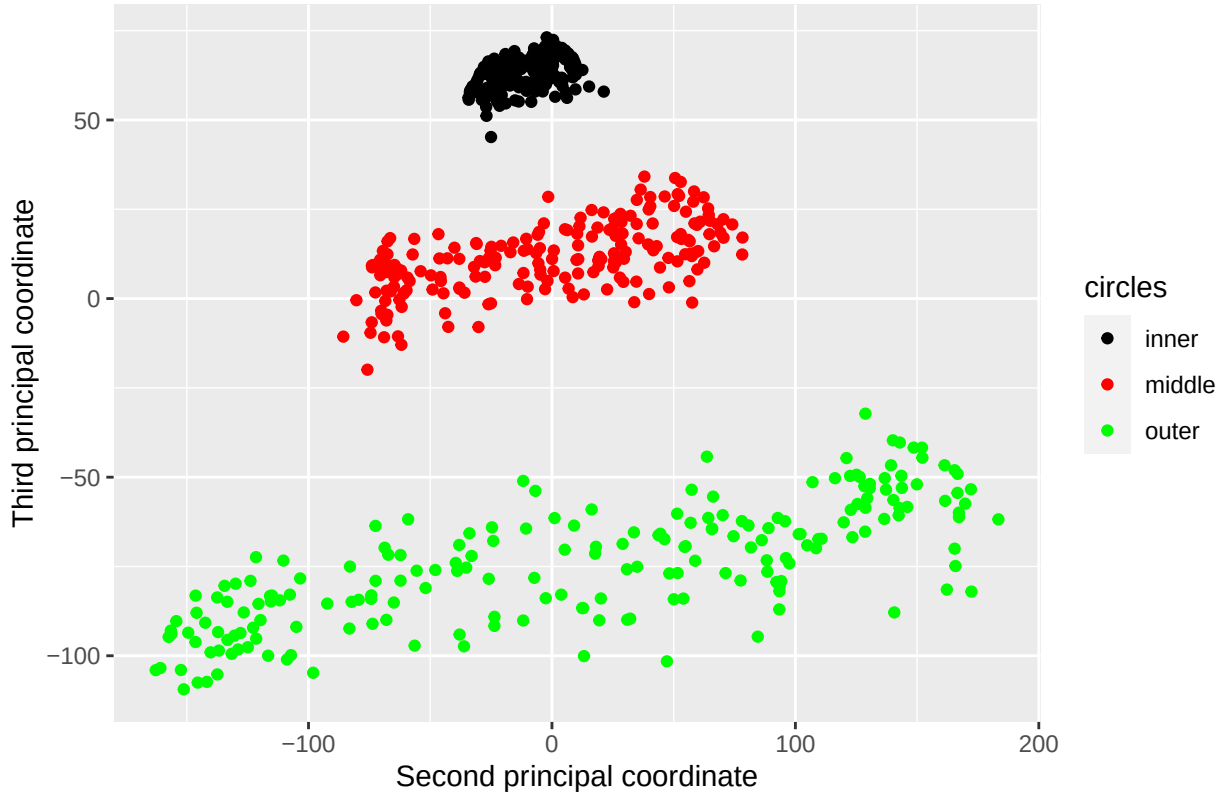
where $< y_i \cdot y_j >$ denotes the scalar product between observation i and observation j. In our case, the two-dimensional data gets projected up into five dimensions using this kernel. By then performing multidimensional

scaling (MDS) on this projected space, we hope to find some pair of principal coordinates which let us (linearly) separate the concentric circles into three distinct clusters. This is what you would call a "brute force" method. There is no guarantee that the data will become separable, and there is generally no way of knowing beforehand which pair of principal coordinates might yield the data as separable.



In Figure 2, we see the first two principal coordinates resulting from the MDS. As is evident, the circles have not become linearly separable. Were we to look at the second and the third principal coordinates, as can be seen from Figure 3, the data actually becomes linearly separable. Since the task description states that we are to look at the first two principal coordinates - going forward, we shall keep to the non-separable result.

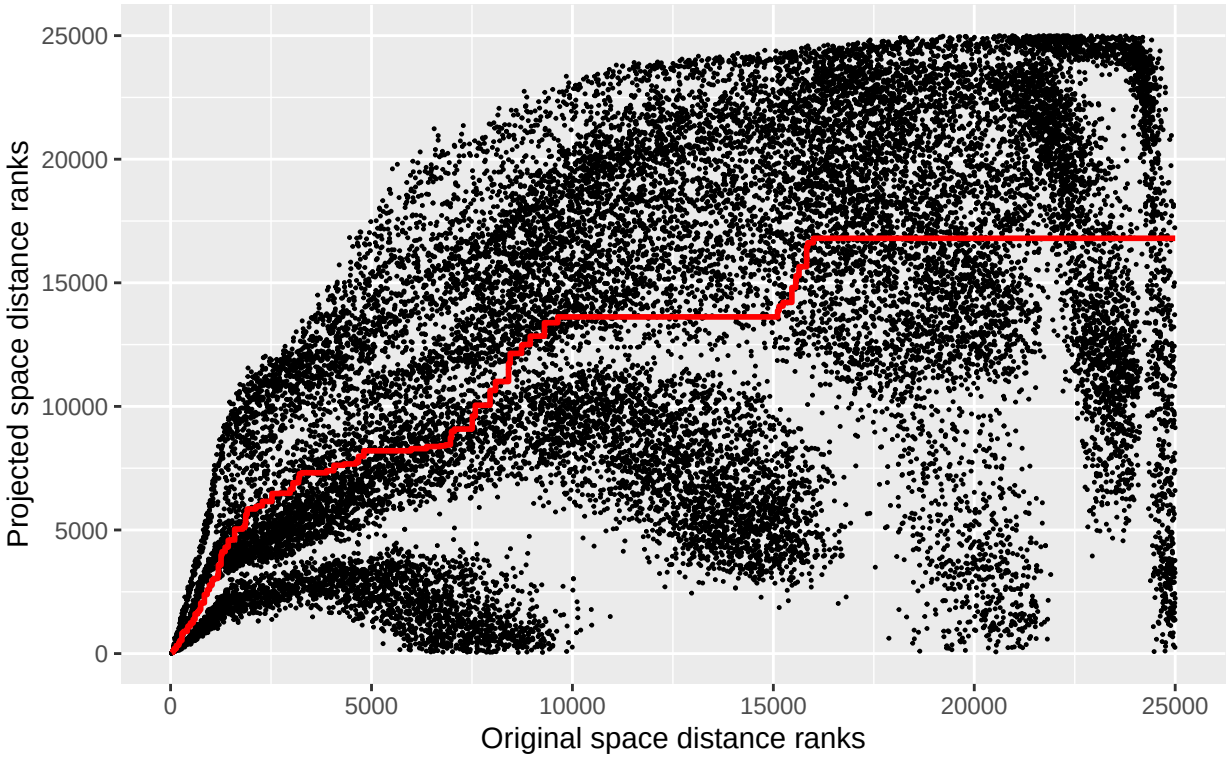
Figure 3: Kernel PCAusing polynomial kernel of degree 2



Task C

The third task asks us to perform validation on the results from the previous task, in the form of a Shepard diagram depicting the ranked distances from the original space against the ranked distances from the projected space. Were we to find that the distance ranks are not well preserved, this tells us that the projected space might not be a fair representation of the original data. As such, we want the ranks to be preserved.

Figure 4: Shepard diagram of distance ranks after using polynomial kernel (n_sample = 25000)

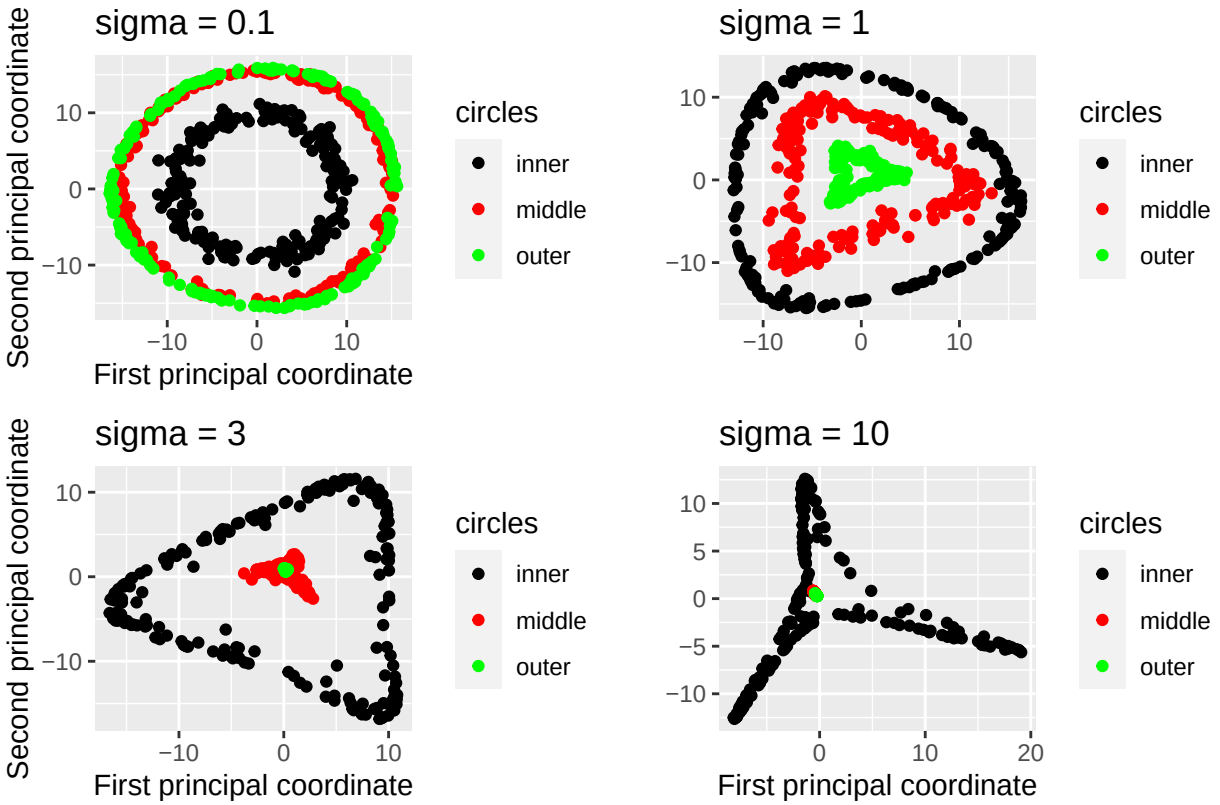


From Figure 4 it is evident that the distance ranks are not very well preserved through the polynomial kernel PCA of degree 2. There seems to be three different "streams" of water that each collapse the high ranks of the original space into the low ranks of the projected space. Looking back at Figure 1 and Figure 2 this seems reasonable, as we can tell from the latter that the points do in fact seem to collapse inwards, towards the center. Thereby somewhat ruining the previous "inter-circle" relationships of the original space.

Task D

We now shift to studying the first two principal coordinates using a Gaussian kernel, paying extra special attention to the hyperparameter σ an letting this shift between four different values 0.1, 1, 3 and 10.

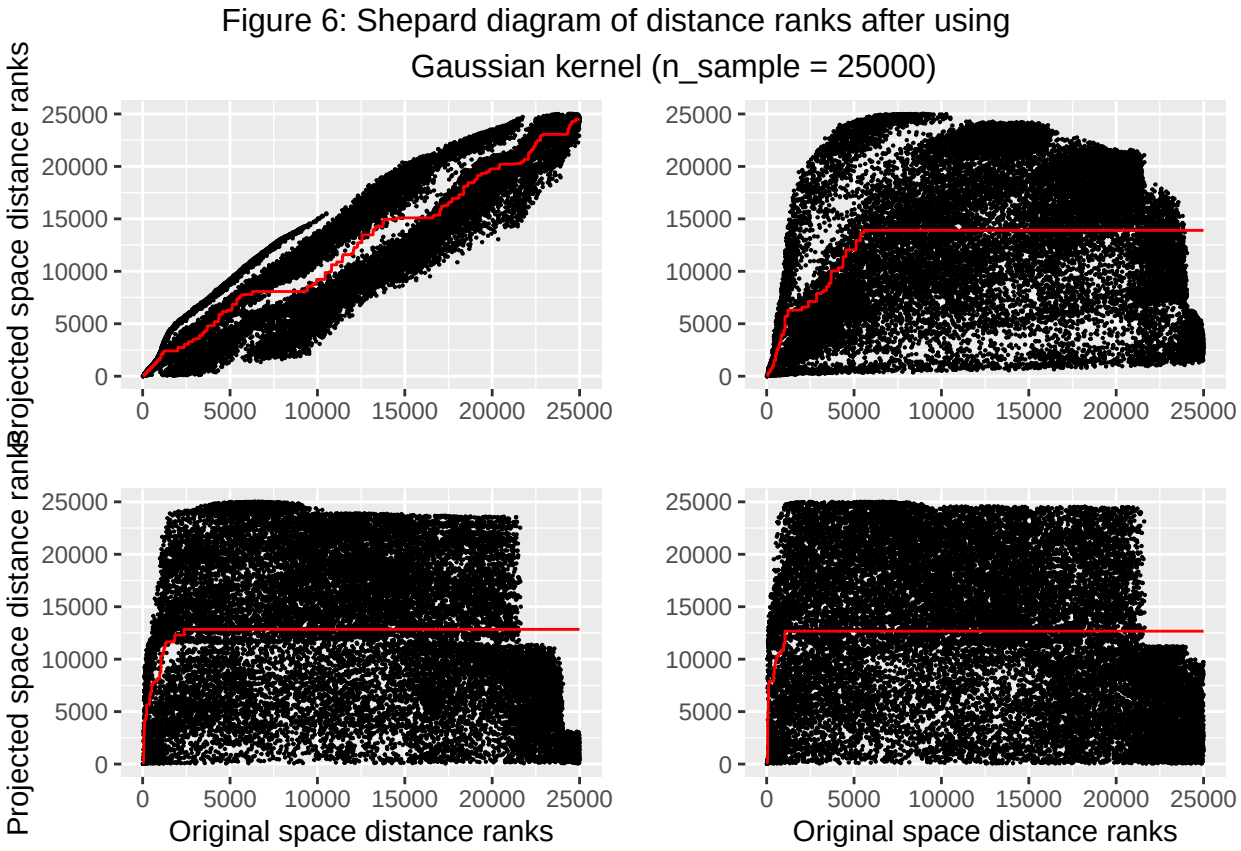
Figure 5: Kernel PCA using a Gaussian kernel for different sigmas



What we deduce from Figure 5 and our experiments is that the higher the value on σ, the lower the regard is for preserving distances. This can be specified further to mean that as we increase σ, we are also increasing the "radius" for what we would consider as locally similar, hence making the transformation less sensitive to subtle differences. Regarding how the structure of the data is preserved this couldn't possibly happen for large enough σ, since there is no way for the function to distinguish between the three circles, as they are all three considered as part of the same locality and thereby the same.

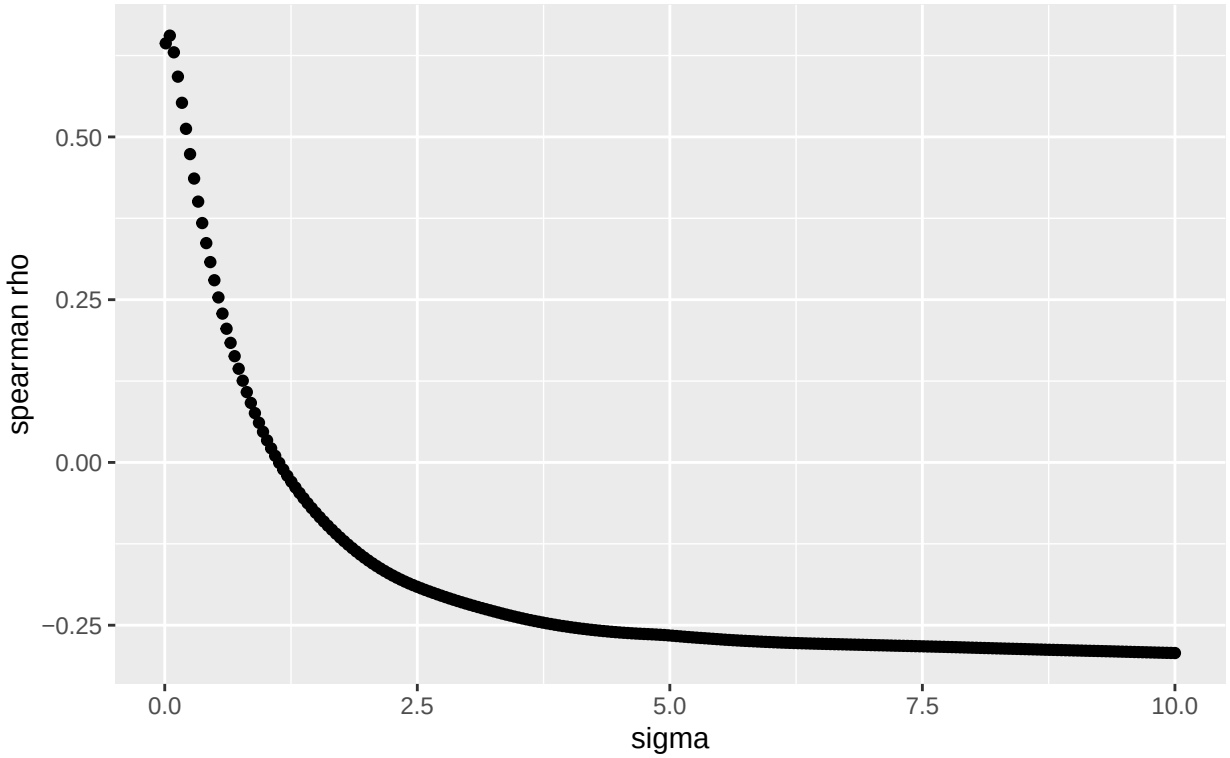
Task E

For the last task we once again want to use a Shepard diagram in order to validate our distance rank preservation from the original space, to the projected space. We especially want to study the Spearman correlation for before and after projection against different values on σ.



Studying the results from Figure 6, we confirm what we hinted at in the previous task - that higher values on σ disregards distance rank preservation even more. We can also confirm the structural preservation is also disregarded as the plots for the higher values on σ now seem to produce more random looking Shepard diagrams. The points seem to be more uniformly distributed in those cases than for the lower values.

Figure 7: Kernel PCA Gaussian kernel different values on sigma against the Spearman correlation coefficient



Finally, looking at Figure 7 we can confirm that the distance preservation seems to get more random (as the correlation decreases). It is harder to find patterns in the different ranges of ranks in the Shepard diagrams of the higher values on σ (as we have already speculated). Note also that the correlation changes sign from positive to negative. This is due to the fact that the circles have changed order. The innermost circle has become the outermost structure and vice versa. In conclusion the correlation seems to deteriorate as we increase σ.

Appendix

```
library(tidyverse)
library(ggpubr)
library(gridExtra)
library(ggisotonic)
library(kernlab)
set.seed(4)
theta1 <- runif(200, 0, 2*pi)
x1 <- cos(theta1) + rnorm(200, 0, 0.1)
y1 <- sin(theta1) + rnorm(200, 0, 0.1)

theta2 <- runif(200, 0, 2*pi)
x2 <- 2*cos(theta1) + rnorm(200, 0, 0.1)
y2 <- 2*sin(theta1) + rnorm(200, 0, 0.1)

theta3 <- runif(200, 0, 2*pi)
x3 <- 3*cos(theta1) + rnorm(200, 0, 0.1)
y3 <- 3*sin(theta1) + rnorm(200, 0, 0.1)
```

```r
data <- data.frame(x1, y1, x2, y2, x3, y3)

data %>%
  ggplot() +
  geom_point(aes(x1, y1), color = "black") +
  geom_point(aes(x2,y2), color = "red") +
  geom_point(aes(x3,y3), color = "green") +
  ggtitle("Figure 1: Concentric circles plot (simulated)")
# observations matrix Y_DxN
x <- c(x1,x2,x3)
y <- c(y1,y2,y3)
Y <- cbind(x,y)
kpc <- kpca(x = Y, kernel = "polydot", kpar = list(degree = 2))
kpc <- as.data.frame(rotated(kpc))
kpc <- kpc
circles <- cbind(c(rep("inner", 200), rep("middle", 200), rep("outer", 200)))
kpc <- kpc %>% cbind(circles)
kpc %>%
  ggplot(aes(x = V1, y = V2, color = circles)) +
  geom_point() +
  ggtitle("Figure 2: Kernel PCA using polynomial kernel of degree 2") +
  xlab("First principal coordinate") +
  ylab("Second principal coordinate") +
  scale_color_manual(values=c("black", "red", "green"))
kpc %>%
  ggplot(aes(x = V2, y = V3, color = circles)) +
  geom_point() +
  ggtitle("Figure 3: Kernel PCAusing polynomial kernel of degree 2") +
  xlab("Second principal coordinate") +
  ylab("Third principal coordinate") +
  scale_color_manual(values=c("black", "red", "green"))

# distances of the original data
orig_dist <- Y %>% dist() %>% as.matrix()

# distances after projection
pc_dist <- kpc %>% select(V1, V2) %>% dist() %>% as.matrix()

n_sample <- 25000
i_sample <- sample(1:length(orig_dist), n_sample)

orig_dist_sample_rank <- rank(orig_dist[i_sample])
pc_dist_sample_rank <- rank(pc_dist[i_sample])
sample_distances <- data.frame(orig_dist_sample_rank, pc_dist_sample_rank)
sample_distances %>%
  ggplot(aes(x = orig_dist_sample_rank, y = pc_dist_sample_rank)) +
  geom_point(size = 0.2) +
  stat_isotonic(color = "red", size = 1) +
  xlab("Original space distance ranks") +
  ylab("Projected space distance ranks") +
  ggtitle("Figure 4: Shepard diagram of distance ranks after using polynomial
          kernel (n_sample = 25000)")
# for sigma = 0.1
```

```r
kpc <- kpca(x = Y, kernel = "rbfdot", kpar = list(sigma = 0.1))
kpc1 <- as.data.frame(rotated(kpc))
circles <- cbind(c(rep("inner", 200), rep("middle", 200), rep("outer", 200)))
kpc <- kpc %>% cbind(circles)

plot1 <- kpc1 %>%
  ggplot(aes(x = V1, y = V2, color = circles)) +
  geom_point() +
  ggtitle("sigma = 0.1") +
  xlab("First principal coordinate") +
  ylab("Second principal coordinate") +
  scale_color_manual(values=c("black", "red", "green"))
# for sigma = 1
kpc <- kpca(x = Y, kernel = "rbfdot", kpar = list(sigma = 1))
kpc2 <- as.data.frame(rotated(kpc))
circles <- cbind(c(rep("inner", 200), rep("middle", 200), rep("outer", 200)))
kpc <- kpc %>% cbind(circles)

plot2 <- kpc2 %>%
  ggplot(aes(x = V1, y = V2, color = circles)) +
  geom_point() +
  ggtitle("sigma = 1") +
  xlab("") +
  ylab("") +
  scale_color_manual(values=c("black", "red", "green"))
# for sigma = 3
kpc <- kpca(x = Y, kernel = "rbfdot", kpar = list(sigma = 3))
kpc3 <- as.data.frame(rotated(kpc))
circles <- cbind(c(rep("inner", 200), rep("middle", 200), rep("outer", 200)))
kpc <- kpc %>% cbind(circles)

plot3 <- kpc3 %>%
  ggplot(aes(x = V1, y = V2, color = circles)) +
  geom_point() +
  ggtitle("sigma = 3") +
  xlab("First principal coordinate") +
  ylab("Second principal coordinate") +
  scale_color_manual(values=c("black", "red", "green"))
# for sigma = 10
kpc <- kpca(x = Y, kernel = "rbfdot", kpar = list(sigma = 10))
kpc4 <- as.data.frame(rotated(kpc))
circles <- cbind(c(rep("inner", 200), rep("middle", 200), rep("outer", 200)))
kpc <- kpc %>% cbind(circles)

plot4 <- kpc4 %>%
  ggplot(aes(x = V1, y = V2, color = circles)) +
  geom_point() +
  ggtitle("sigma = 10") +
  xlab("First principal coordinate") +
  ylab("") +
  scale_color_manual(values=c("black", "red", "green"))
grid.arrange(plot1, plot2, plot3, plot4,
                         top = "Figure 5: Kernel PCA using a Gaussian kernel for different sigmas")
```

```r
# distances after projection
pc_dist1 <- kpc1 %>% select(V1, V2) %>% dist() %>% as.matrix()
pc_dist2 <- kpc2 %>% select(V1, V2) %>% dist() %>% as.matrix()
pc_dist3 <- kpc3 %>% select(V1, V2) %>% dist() %>% as.matrix()
pc_dist4 <- kpc4 %>% select(V1, V2) %>% dist() %>% as.matrix()

n_sample <- 25000
i_sample <- sample(1:length(orig_dist), n_sample)

orig_dist_sample_rank <- rank(orig_dist[i_sample])
pc_dist_sample_rank1 <- rank(pc_dist1[i_sample])
pc_dist_sample_rank2 <- rank(pc_dist2[i_sample])
pc_dist_sample_rank3 <- rank(pc_dist3[i_sample])
pc_dist_sample_rank4 <- rank(pc_dist4[i_sample])
sample_distances <- data.frame(orig_dist_sample_rank, pc_dist_sample_rank1,
                               pc_dist_sample_rank2, pc_dist_sample_rank3,
                               pc_dist_sample_rank4)

splot1 <- sample_distances %>%
  ggplot(aes(x = orig_dist_sample_rank, y = pc_dist_sample_rank1)) +
  geom_point(size = 0.1) +
  stat_isotonic(color = "red") +
  xlab("") +
  ylab("Projected space distance ranks")
splot2 <- sample_distances %>%
  ggplot(aes(x = orig_dist_sample_rank, y = pc_dist_sample_rank2)) +
  geom_point(size = 0.1) +
  stat_isotonic(color = "red") +
  xlab("") +
  ylab("")
splot3 <- sample_distances %>%
  ggplot(aes(x = orig_dist_sample_rank, y = pc_dist_sample_rank3)) +
  geom_point(size = 0.1) +
  stat_isotonic(color = "red") +
  xlab("Original space distance ranks") +
  ylab("Projected space distance ranks")
splot4 <- sample_distances %>%
  ggplot(aes(x = orig_dist_sample_rank, y = pc_dist_sample_rank4)) +
  geom_point(size = 0.1) +
  stat_isotonic(color = "red") +
  ylab("") +
  xlab("Original space distance ranks")

grid.arrange(splot1, splot2, splot3, splot4,
             top = "Figure 6: Shepard diagram of distance ranks after using
             Gaussian kernel (n_sample = 25000)")
sigmas <- seq(0.01,10, length.out = 250)
spearmen <- map_dbl(sigmas, function(.x){
    kpc <- kpca(Y, kernel = "rbfdot", kpar = list(sigma = .x))
    kpc <- as.data.frame(rotated(kpc))
    pc_dist <- kpc %>% select(V2, V3) %>% dist() %>% as.matrix()
    pc_dist_sample_ranking <- rank(pc_dist[i_sample])
    return(cor(orig_dist_sample_rank, pc_dist_sample_ranking,
```

```
              method = "pearson"))
  })
sigmas_spearmen <- data.frame(sigmas, spearmen)
sigmas_spearmen %>%
  ggplot(aes(sigmas, spearmen)) +
  geom_point() +
  ggtitle("Figure 7: Kernel PCA Gaussian kernel different values on sigma against
          the Spearman correlation coefficient") +
  xlab("sigma") +
  ylab("spearman rho")
```