title: Charles Zheng EE 378b HW 6 output: html_document: mathjax: default —

Charles Zheng EE 378b HW 6

Setup

```
library(knitr)
opts_knit$set(root.dir=normalizePath('../'))
```

Loading the data:

```
library(Rcpp)
library(pracma)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following objects are masked from 'package:pracma':
##
##     and, mod, or
```

```
library(RcppArmadillo)
library(grid)
getwd()
```

```
## [1] "/home/snarles/github/misc"
```

```
load("ee378b/images.RData")
imgs[1, ] %>% matrix(64, 64) %>% t %>% fliplr %>% image(col = gray(0:20/20), axes = FALSE)
```

Euclidean distance matrix

```
dm <- distmat(imgs, imgs)
diag(dm) <- 0
```

ISOMAP

Functions

```
cppFunction('
NumericMatrix floydWarshall(NumericMatrix Ar) {
  int n = Ar.nrow();
  for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        double x = Ar(i, k) + Ar(k, j);
        if (x < Ar(i, j)) {
          Ar(i, j) = x;
        }
      }
    }
  }
  return Ar;
}
')

knn_dist_matrix <- function(a, k) {
  for (i in 1:dim(a)[1]) {
    kthres <- sort(a[i, ])[k + 1]
    a[i, a[i, ] > kthres] <- Inf
  }
  a
}
```

With k = 10, form the k-nearest neighbors distance matrix and check that it is connected.

```
dm2 <- knn_dist_matrix(dm, k = 10)
dm3 <- floydWarshall(dm2)
sum(dm3 == Inf)
```

```
## [1] 0
```

Run MDS

```
mds <- function(a, d = 2) {
  n <- dim(a)[1]
  u <- t(t(rep(1, n)))
  tu <- t(u)
  b <- a - (1/n) * u %*% (tu %*% a) - (1/n) * (a %*% u) %*% tu + mean(a)
  res <- svd(b)
```

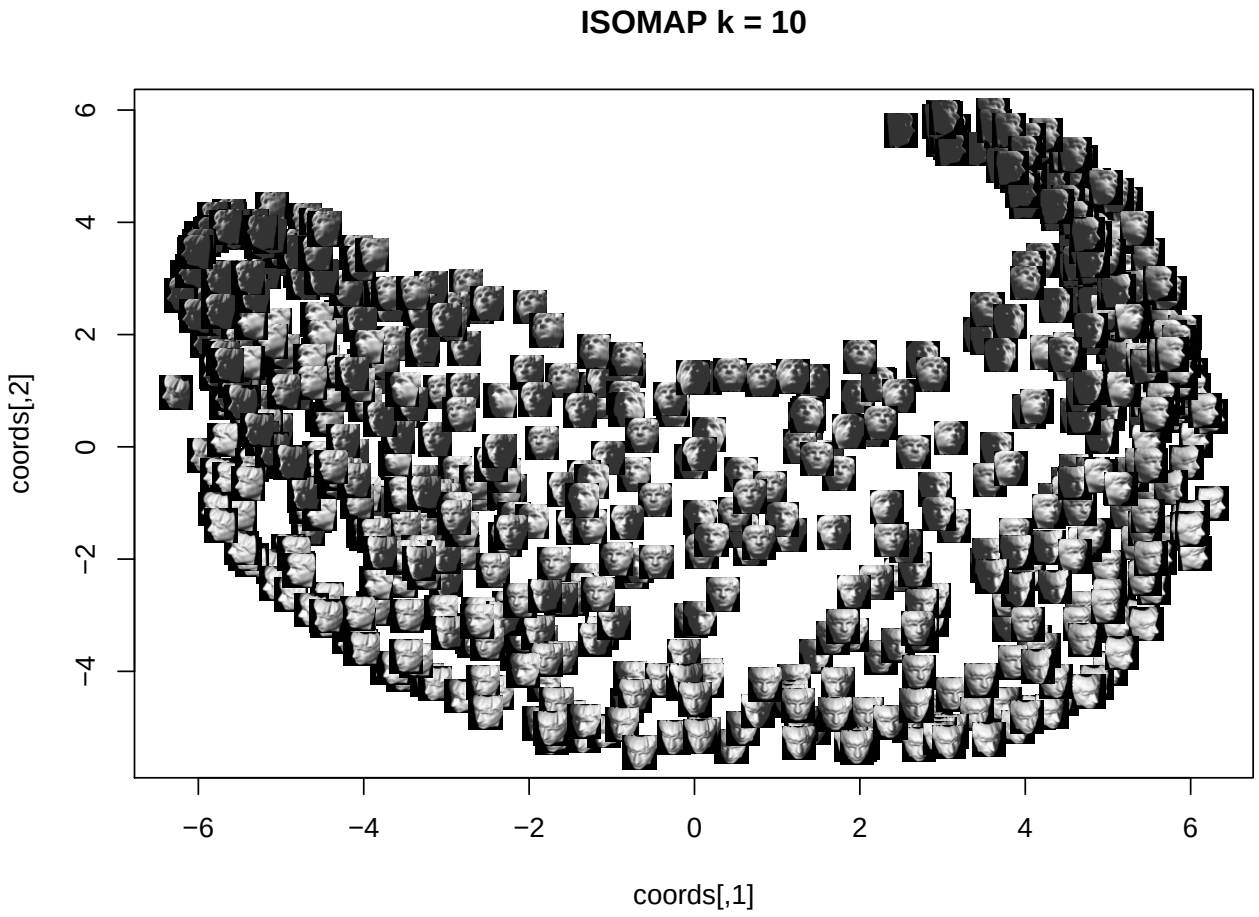
```
  coords <- res$u[, 1:d] %*% diag(sqrt(res$d[1:d]))
  Re(coords)
}

coords <- mds(dm3, 2)
dim(coords)
```

```
## [1] 698   2
```

Plot coords with thumbnails

```
pic_raster <- function(i) {
  imgs[i, ] %>% matrix(64, 64) %>% as.raster
}
plot(coords, main = "ISOMAP k = 10")
scale_x <- .2
scale_y <- .3
i <- 1
for (i in 1:698) {
  rasterImage(pic_raster(i), coords[i, 1] - scale_x, coords[i, 2] - scale_y,
              coords[i, 1] + scale_x, coords[i, 2] + scale_y)
}
```



Local Linear Embedding

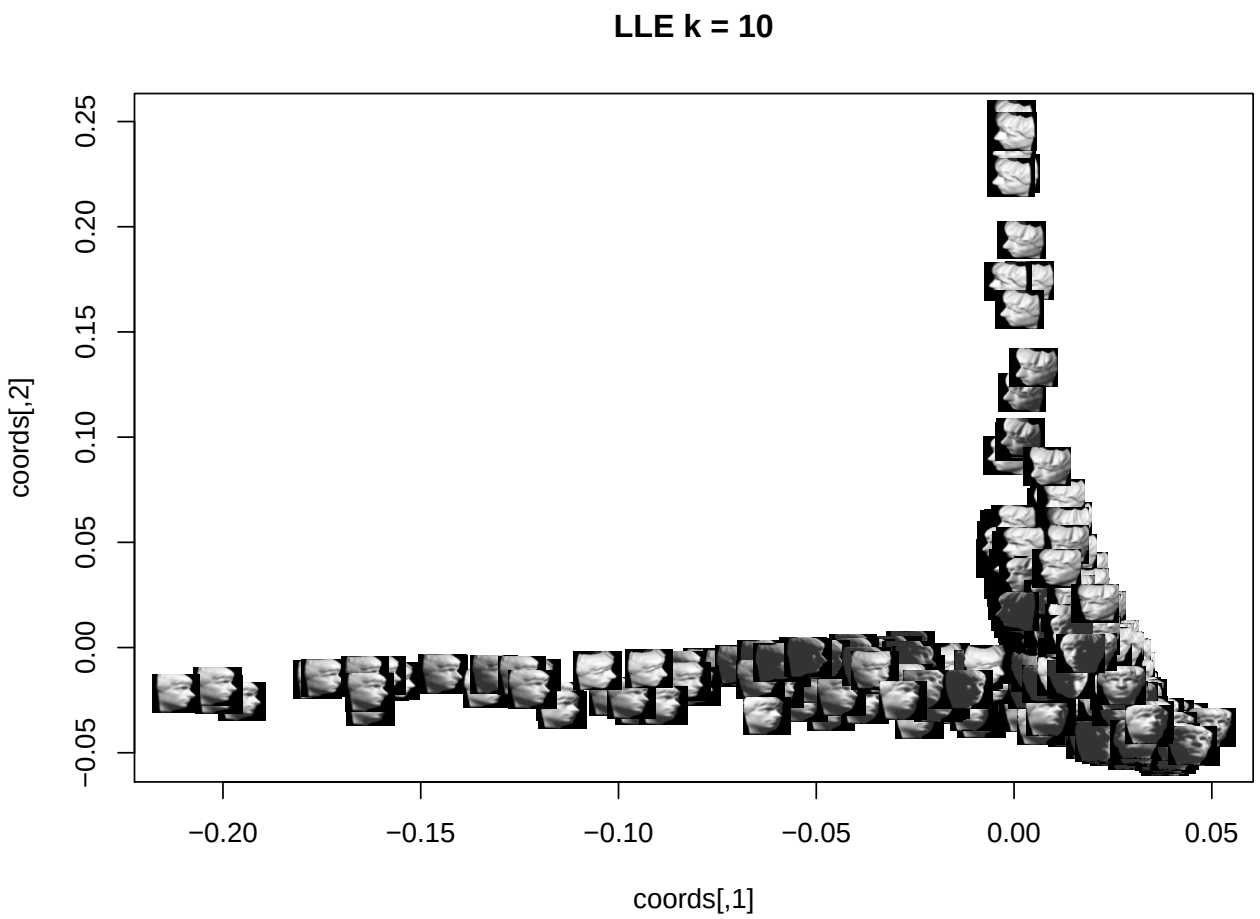
Code

```
get_ws <- function(x, k = 10) {
  n <- dim(x)[1]
  dm <- as.matrix(dist(x))
  diag(dm) <- Inf
  ords <- t(apply(dm, 2, order))
  nn_inds <- ords[, 1:k]
  ws <- matrix(0, n, n)
  for (i in 1:n) {
    xi <- x[i, ]
    xs <- t(x[nn_inds[i, ], ])
    xi <- xi - xs[, 1]
    xs <- xs[, -1] - xs[, 1]
    v <- lm(xi ~ xs + 0)$coefficients
    w <- c(1 - sum(v), v)
    ws[i, nn_inds[i, ]] <- w
  }
  ws
}

get_ys <- function(ws, d = 2) {
  n <- dim(ws)[1]
  mm <- -ws
  diag(mm) <- 1 + diag(mm)
  #lmax <- norm(mm, type = "2")^2
  #res <- svd(mm)
  #y <- res$v[, n - (1:d)]
  mmat <- diag(rep(1, n)) - (t(mm) %*% mm)
  res <- eigen(mmat)
  res$values[1:3]
  y <- res$vectors[, 1 + (1:d)]
  y
}
```

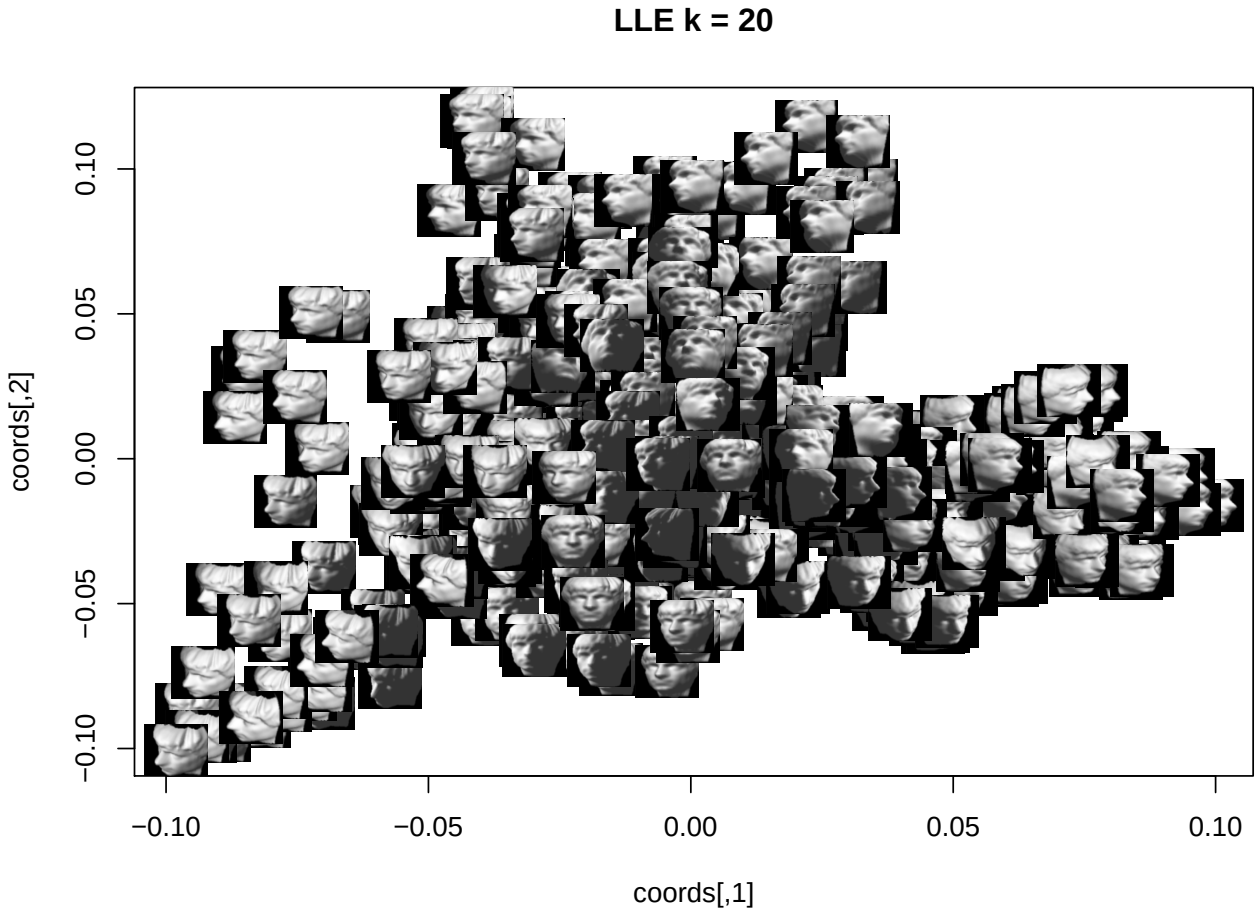
With k = 10, compute local linear embedding coords. Plot coords with thumbnails

```
ws <- get_ws(imgs, 10)
coords <- get_ys(ws)
plot(coords, main = "LLE k = 10")
scale_x <- 0.01 * .6
scale_y <- 0.015 * .6
i <- 1
for (i in 1:698) {
  rasterImage(pic_raster(i), coords[i, 1] - scale_x, coords[i, 2] - scale_y,
              coords[i, 1] + scale_x, coords[i, 2] + scale_y)
}
```



The result is very bad with k = 10. Try k = 20.

```
ws <- get_ws(imgs, 20)
coords <- get_ys(ws)
plot(coords, main = "LLE k = 20")
scale_x <- 0.01 * .6
scale_y <- 0.015 * .6
i <- 1
for (i in 1:698) {
  rasterImage(pic_raster(i), coords[i, 1] - scale_x, coords[i, 2] - scale_y,
              coords[i, 1] + scale_x, coords[i, 2] + scale_y)
}
```



Comments

ISOMAP produces a more uniformly distributed set of coordinates. Also, ISOMAP works well with smaller k (number of neighbors). When LLE was applied with k = 10, the result was a degenerate mapping into a "star shape" rather than a truly 2-dimensional representation. LLE works better with k = 20, but the result is still not as uniform as ISOMAP.

The fact that ISOMAP works with smaller k is desirable since it means that it can detect the structure of manifolds with "rougher" embeddings