STAT-S675

Homework 6

John Koo

Problem 1

[Exercise 5.7.2 from the text]

We want to show $LX = M(X)X$ is equivalent to $\sum_k^n w_{rk}(x_{rs} - x_{ks}) = \sum_k^n w_{rk}\frac{\delta_{rk}}{d_{rk}}(x_{rs} - x_{ks})$.

$L = \frac{1}{2}\sum_{i,j} w_{ij}(e_i - e_j)(e_i - e_j)^T$
$L = T - W$
$M(X) = \sum_{i,j} w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}(e_i - e_j)(e_i - e_j)^T$

Show $LX = \sum_k^n w_{rk}(x_{rs} - x_{ks})$

Since $L = T - W$, $LX = (T - W)X = TX - WX$.

$[TX]_{ik} = \sum_j t_{ij}x_{jk}$

But T is a diagonal matrix. Therefore $[TX]_{ik} = t_{ii}x_{ik}$.

Then note that $t_{ii} = \sum_j w_{ij}$. Therefore, $[TX]_{ik} = \sum_j w_{ij}x_{ik}$.

Then moving onto $[WX]_{ik} = \sum_j w_{ij}x_{ik}$.

Therefore, $[TX - WX]_{ik} = \sum_j w_{ij}x_{ik} - \sum_j w_{ij}x_{ik}$

Or grouping them under one sum, $[TX - WX]_{ik} = \sum_j (w_{ij}x_{ik} - w_{ij}x_{ik})$

But we can just change the variables: $i \to r$, $j \to k$, and $k \to s$. Therefore, $[TX - WX]_{rs} = \sum_k (w_{rk}x_{ik} - w_{rs}x_{ks})$

And after factoring, $[LX]_{rs} = [TX - WX]_{rs} = \sum_k^n w_{rk}(x_{rs} - x_{ks})$

Show $M(X)X = \sum_k^n w_{rk}\frac{\delta_{rk}}{d_{rk}}(x_{rs} - x_{ks})$

Note that $w_{ij}\frac{\delta_{ij}}{d_{ij}}(e_i - e_j)(e_i - e_j)^T$ is a matrix of all zeroes except for the elements at ii and jj which are $w_{ii}\frac{\delta_{ii}}{d_{ii}}$ and $w_{jj}\frac{\delta_{jj}}{d_{jj}}$ respectively, as well as the elements at ij and ji which are $-w_{ij}\frac{\delta_{ij}}{d_{ij}} = -w_{ji}\frac{\delta_{ji}}{d_{ji}}$. So then we can describe the diagonal elements $[M(X)]_{ii}$ as $\sum_j w_{ij}\frac{\delta_{ij}}{d_{ij}}$ and the off-diagonal elements $[M(X)]_{ij} = -w_{ij}\frac{\delta_{ij}}{d_{ij}}$.

Next, we can say that $M(X) = M^d(X) + M^{-d}(X)$ where $M^d(X)$ is the diagonal matrix of $M(X)$ while $M^{-d}(X)$ is the off-diagonal matrix.

Then $[M^d(X)X]_{ik} = m^d_{ii}x_{ik}$ (since $M^d(X)$ is diagonal), but then $m^d_{ii} = \sum_j w_{ij}\frac{\delta_{ij}}{d_{ij}}$, so $[M^d(X)X]_{ik} = \sum_j w_{ij}\frac{\delta_{ij}}{d_{ij}}x_{ik}$.

On the other hand, $[M^{-d}(X)X]_{ik} = m^{-d}_{ij}x_{ik} = \sum_j -w_{ij}\frac{\delta_{ij}}{d_{ij}}x_{ik}$.

Then ...

$$[M(X)X]_{ik} = [(M^d(X) + M^{-d}(X))X]_{ik}$$

$$= [M^d(X)]_{ik} + [M^{-d}(X)]_{ik}$$

$$\begin{aligned}
&= \sum_j w_{ij} \frac{\delta_{ij}}{d_{ij}} x_{ik} + \sum_j -w_{ij} \frac{\delta_{ij}}{d_{ij}} x_{ik} \\
&= \sum_j w_{ij} \frac{\delta_{ij}}{d_{ij}} x_{ik} - w_{ij} \frac{\delta_{ij}}{d_{ij}} x_{ik} \\
&= \sum_j w_{ij} \frac{\delta_{ij}}{d_{ij}} (x_{ik} - x_{ik})
\end{aligned}$$

And then if we just change the arbitrary indices as before, we get:

$$\sum_k w_{rk} \frac{\delta_{rk}}{d_{rk}} (x_{rs} - x_{ks})$$

Problem 2

[Exercise 5.7.3 from the text]

By definition, A is positive definite iff $\forall x \in \mathbb{R}^n \backslash \{0\}$, $x^T A x > 0$.

So given an arbitrary x, we can check if $x^T(L + ee^T)x > 0$.

$$\begin{aligned}
x^T(L + ee^T)x &= x^T L x + x^T ee^T x \\
&= x^T L x + (e^T e)(e^T x) \\
&= x^T L x + (e^T x)^2
\end{aligned}$$

since $e^T x = x^T e \in \mathbb{R}^n$. Then

$$x^T L x + (e^T x)^2 = x^T L x + \Big(\sum_i^n x_i \Big)^2$$

We know that L is positive semidefinite, so $x^T L x \geq 0$. On the other hand, $(\sum_i^n x_i)^2 \geq 0$. Therefore, $x^T(L + ee^T)x = x^T L x + (\sum_i^n x_i)^2 \geq 0 \implies L + ee^T$ is positive semidefinite.

So now what's left is showing that $L + ee^T$ is positive *definite*. Since both L and ee^T are positive semidefinite, we just have to show that if $x^T L x = 0$ then $x^T ee^T x \neq 0$ or vice versa.

Let $x^T ee^T x = 0$. Then $x^T ee^T x = (x^T e)(e^T x) = (e^T x)^2 \implies e^T x = 0 \implies e \perp x$.

So now let $e \perp x$ and examine $x^T L x$. From **Theorem C.1** in the text, we know that L has one zero eigenvalue with its corresponding eigenvector being of the form αe. Therefore, if $x \perp \alpha e$, then $x^T L x \neq 0$. Since $x^T L x \geq 0$, $x^T L x$ must be greater than 0, hence $x^T(L + ee^T)x > 0 \implies L + ee^T$ is positive definite.

Problem 3

Parts 1 and 2

```r
import::from(readr, read_table)
import::from(magrittr, `%>%`, `%<>%`)
source('http://pages.iu.edu/~mtrosset/Courses/675/stress.r')
library(ggplot2)
theme_set(ggthemes::theme_base())
set.seed(6756)

# load the data
# note that it's a lower triangular matrix--fill in the rest
data.url <- 'http://pages.iu.edu/~mtrosset/Courses/675/colors.dat'
Delta <- read_table(data.url, col_names = FALSE) %>%
  as.matrix() %>%
  {. + t(.)}

# generate random 2 column matrix
rows <- nrow(Delta)
cols <- 2
stdev <- 40
X.rand <- matrix(rnorm(rows * cols, sd = stdev), nrow = rows, ncol = cols)

# compute raw stress of the random matrix
mds.stress.raw.eq(X.rand, Delta)
```

```
[1] 160697
```

```r
# center X.rand
X.rand %<>% apply(2, function(x) x - mean(x))
apply(X.rand, 2, sum)
```

```
[1] -2.309264e-14  6.217249e-15
```

```r
# and check its stress criterion
mds.stress.raw.eq(X.rand, Delta)
```

```
[1] 160697
```

Centering the data matrix did not change the stress criterion. Looking at `mds.stress.raw.eq` more closely, this is expected, since it just compares the distance matrix of `X.rand` to `Delta`, and shifting `X.rand` doesn't do anything to its distance matrix.

Part 3

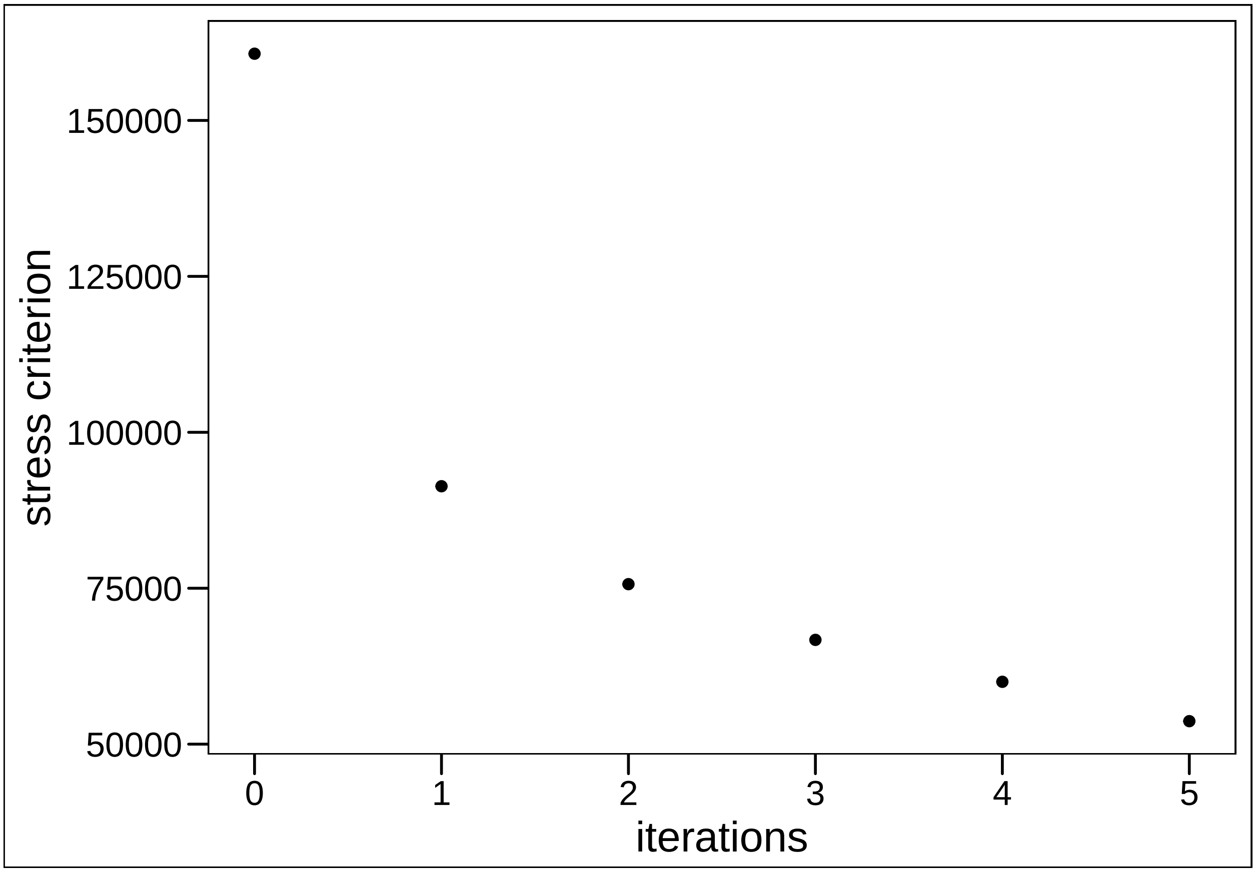
```r
# iterations
K <- 64
k.vector <- seq(0, K)

# initialize stress vector
stress.vector <- rep(NA, K + 1)
stress.vector[1] <- mds.stress.raw.eq(X.rand, Delta)

# iterate
for (k in k.vector[-1]) {
  X.rand <- mds.guttman.eq(X.rand, Delta)
  stress.vector[k + 1] <- mds.stress.raw.eq(X.rand, Delta)
```

```
}

mds.df <- dplyr::data_frame(k = k.vector, random.stress = stress.vector)
ggplot(mds.df[seq(6), ]) +
  geom_point(aes(x = k, y = random.stress)) +
  labs(x = 'iterations', y = 'stress criterion')
```



Part 4

```
X.cmds <- cmdscale(Delta)
mds.stress.raw.eq(X.cmds, Delta)
```

```
[1] 25880.08
```

Part 5

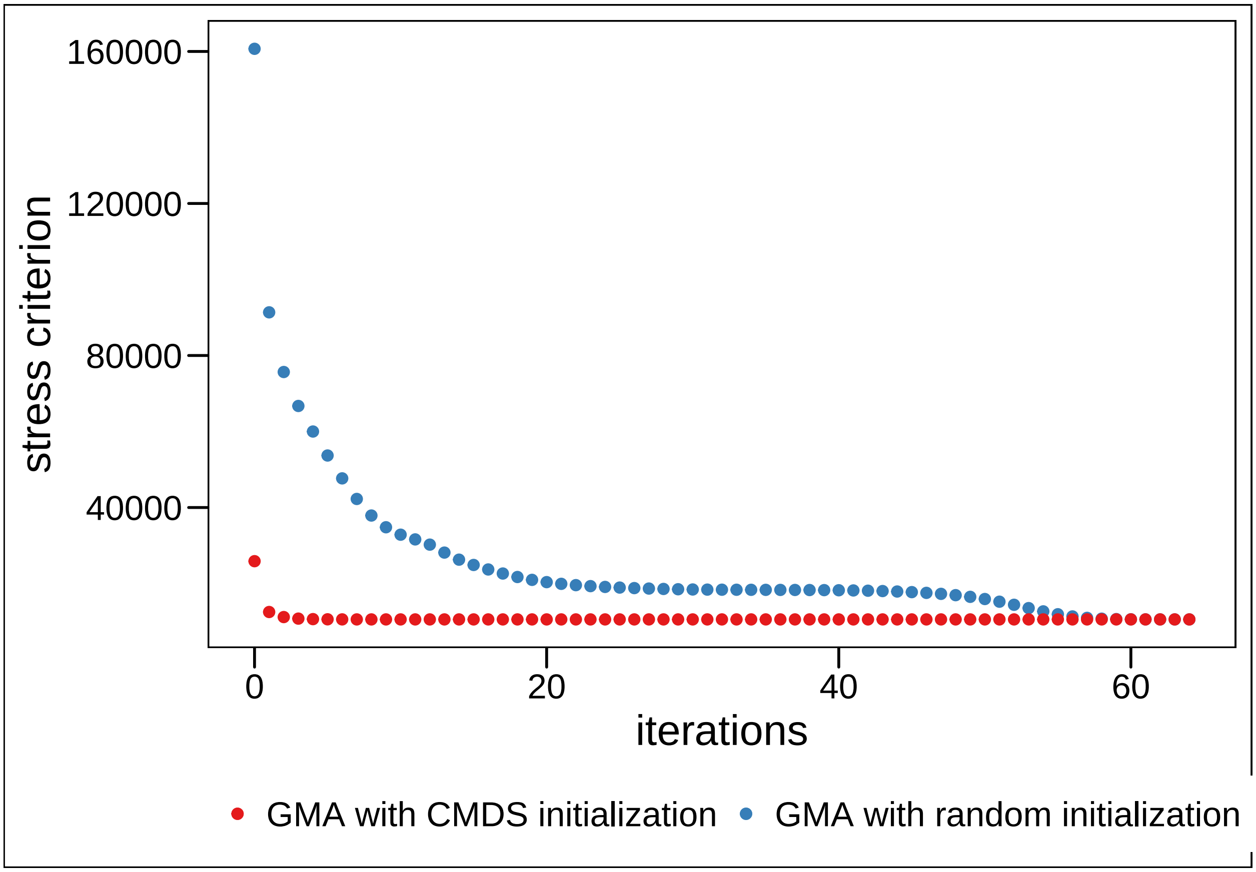
```
# initialize
cmds.stress.vector <- rep(NA, K + 1)
cmds.stress.vector <- mds.stress.raw.eq(X.cmds, Delta)

# iterate
for (k in k.vector[-1]) {
  X.cmds <- mds.guttman.eq(X.cmds, Delta)
```

```
  cmds.stress.vector[k + 1] <- mds.stress.raw.eq(X.cmds, Delta)
}

mds.df$cmds.stress <- cmds.stress.vector

ggplot(mds.df) +
  geom_point(aes(x = k,
                 y = random.stress,
                 colour = 'GMA with random initialization')) +
  geom_point(aes(x = k,
                 y = cmds.stress,
                 colour = 'GMA with CMDS initialization')) +
  scale_colour_brewer(palette = 'Set1') +
  theme(legend.position = 'bottom') +
  labs(x = 'iterations', y = 'stress criterion', colour = NULL)
```



Here we see that while the CMDS initialization basically converges after 3 or 4 iterations, the randomly initialized matrix has a much higher stress criterion and is far from converging. In fact, it takes ~60 iterations for the random matrix to converge to the same stress criterion as the CMDS initialization, and it seems to have gotten stuck at a higher value for ~30 iterations.