Homework 5

Ben Tward

2025-04-09

Question 1

a.

According to the Spectral Theorem, any symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be factored into the product of matrices $\mathbf{VDV}^T$ where $\mathbf{V}$ is orthonormal and $\mathbf{D}$ is a diagonal matrix of the eigenvalues of $\mathbf{A}$. We can also express this as a summation $\sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$.

We can also say that for any matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{A}^T\mathbf{A}$ is symmetric and positive semi-definite (the eigenvalues $\{\lambda_i\}_{i=1}^n > 0$. Define $\sigma_i^2 = \lambda_i$, and these values must be real by definition. From this, we get that $\mathbf{A}^T\mathbf{A} = \mathbf{VDV}^T$. We also know $\mathbf{A}^T\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i = \sigma_i^2\mathbf{v}_i$. We can kind of rearrange this and define a new eigenvector of $\mathbf{AA}^T$ which is $\mathbf{u}_i = \frac{\mathbf{A}\mathbf{v}_i}{\sigma_i}$.

If we make a matrix Σ which consists of the singular values σ on the diagonal, we can express $\mathbf{U} = \mathbf{AV\Sigma}^{-1}$. Rearranging this, we get $\mathbf{A} = \mathbf{U\Sigma V}^T$ which is precisely our SVD. We can also see through this process that all these calculations are concrete and therefore the SVD solution is unique.

b.

We know that a rank-k approximation of $\mathbf{A}$ is $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. We want show that $\mathbf{A}_k$ minimizes the Frobenius norm (or equivalently the squared norm):

$||\mathbf{A} - \mathbf{A}_k||_F^2 = ||\sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T - \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T||_F^2 = ||\sum_{i=k+1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2$

Because $\mathbf{u}_i$ and $\mathbf{v}_i^T$ are orthogonal so the σ_i^2 terms add.

If we compare this to any arbitrary matrix $\mathbf{B}$ with the constraint $rank(\mathbf{B}) = k$, we must show that $||\mathbf{A} - \mathbf{A}_k||_F^2 \leq ||\mathbf{A} - \mathbf{B}||_F^2$

If we do a k-rank approximation for $\mathbf{B}$, we get terms that cannot cancel and we are left with the result that $\mathbf{A}_k$ is the best approximation to minimize the Frobenius norm.

Question 2

a)

```r
data = read.csv('CityDistances.csv')
print(data)
```

```
##      City...City Salt.Lake.City Ann.Arbor  Tokyo Addis.Ababa Cape.Town
## 1 Salt Lake City            0.0    1452.9 5473.4      8520.8    9702.6
## 2      Ann Arbor         1452.9       0.0 6389.5      7368.4    8312.9
## 3          Tokyo         5473.4    6389.5    0.0      6465.3    9158.2
## 4    Addis Ababa         8520.8    7368.4 6465.3         0.0    3252.1
```

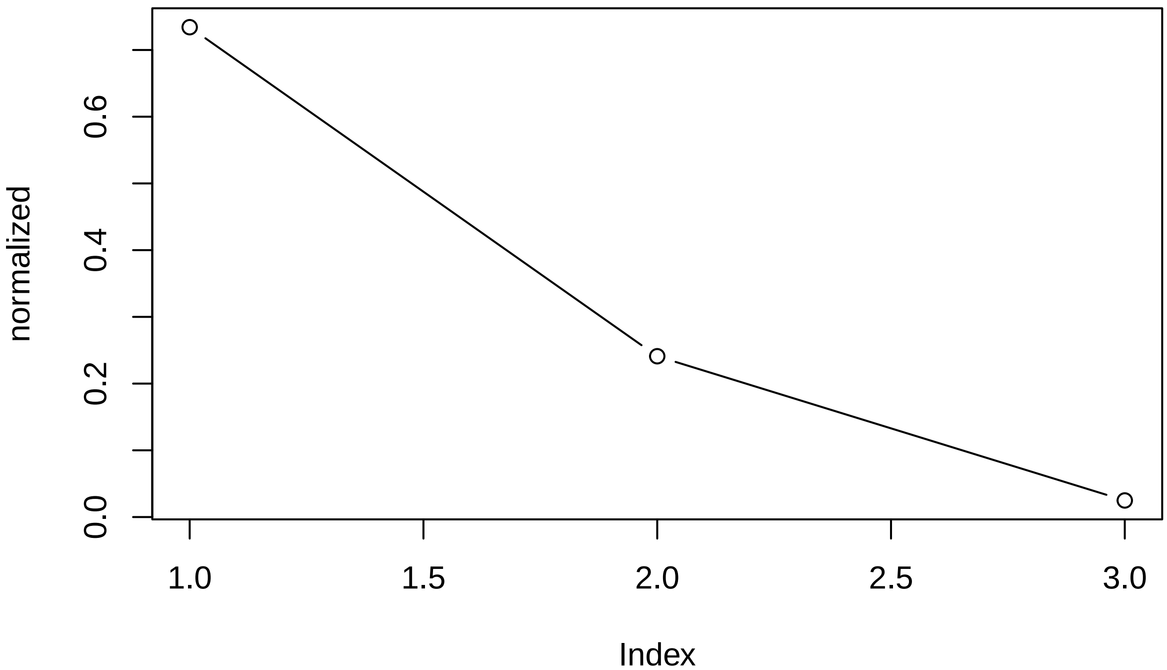
```
## 5      Cape Town         9702.6     8312.9 9158.2      3252.1       0.0
## 6    Los Angeles          580.5     1945.5 5472.2      9099.9    9975.2
## 7  New York City         1968.0      515.7 6737.0      6959.3    7806.8
##   Los.Angeles New.York.City
## 1       580.5        1968.0
## 2      1945.5         515.7
## 3      5472.2        6737.0
## 4      9099.9        6959.3
## 5      9975.2        7806.8
## 6         0.0        2448.8
## 7      2448.8           0.0
```

b)

```
mds = function(D, k) {
  D = as.matrix(D)
  n = dim(D)[1]
  e = as.matrix(rep(1, n), n, 1)
  I = diag(nrow=n)
  H = I - ((1/n) * (e %*% t(e)))
  B = -.5 * (H %*% D %*% H)
  eigenB = eigen(B)
  Uk = eigenB$vectors[,1:k]
  Lambdak = eigenB$values[1:k]
  Xtilde = Uk %*% diag(Lambdak)
  return(list(Xtilde = Xtilde, eigs = Lambdak))
}
```

c)

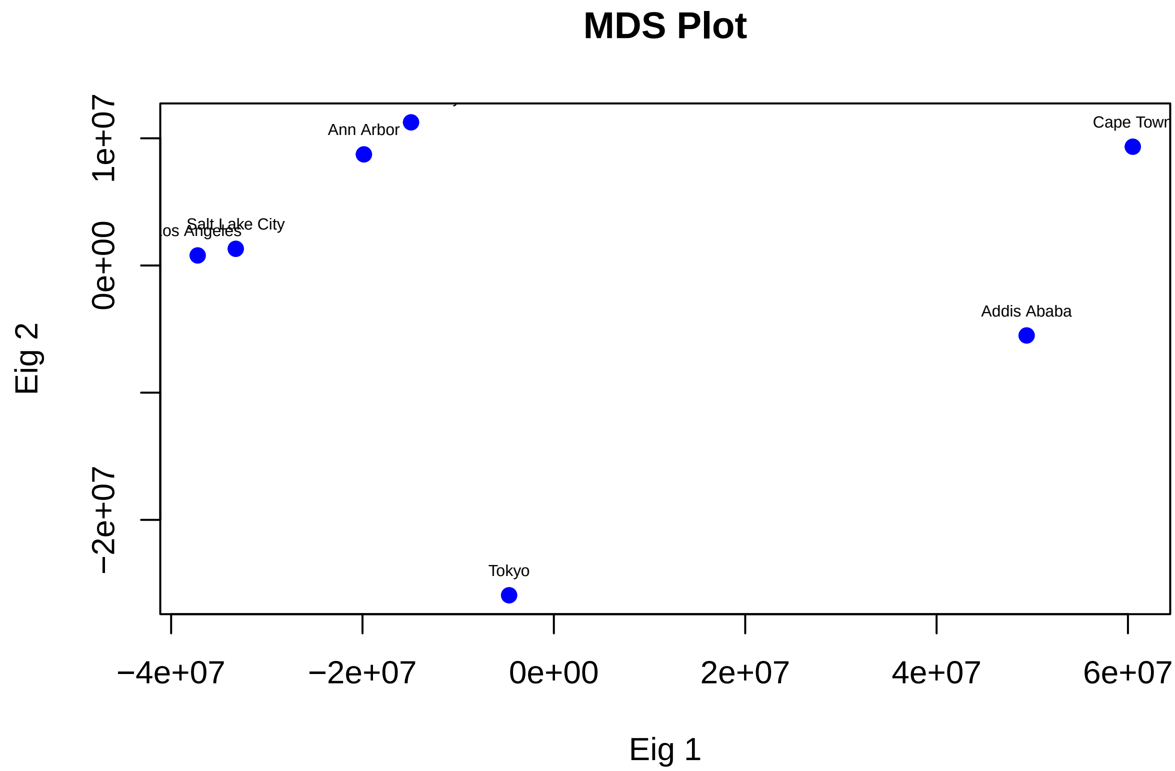
```
D = (data[,-1])^2
mdsD = mds(D, 3)
eigs = mdsD$eigs
normalized = eigs / sum(eigs)
plot(normalized, type='b')
```



I do not see any negative eigenvalues in my data. However, I read online that it is possible for there to be negative eigenvalues, which is usually a sign that MDS is inappropriate on that data. If our distance matrix $\mathbf{D}^{\mathbf{X}}$ is computed using Euclidian distance, then $\mathbf{B}^{\mathbf{X}}$ is guaranteed to be positive semi-definite.

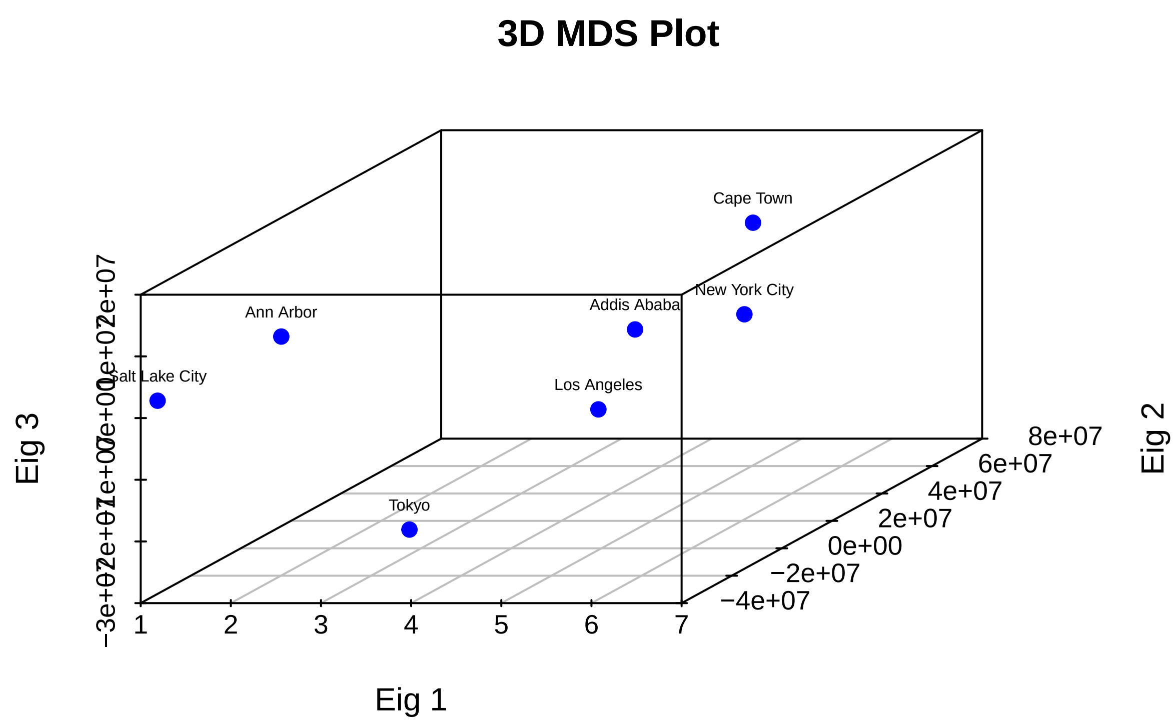
d)

```
names = data[,1]
mdsD = mds(D, 2)
X = mdsD$Xtilde
plot(X[,1], X[,2], xlab = "Eig 1", ylab = "Eig 2",
     main = "MDS Plot", pch = 19, col = "blue")
text(X[,1], X[,2], labels = names, pos = 3, cex = 0.5)
```




```
s3d = scatterplot3d(X, xlab = "Eig 1", ylab = "Eig 2", zlab = "Eig 3",
                    pch = 19, color = "blue", main = "3D MDS Plot")

coords = s3d$xyz.convert(X)
text(coords$x, coords$y, labels = names, pos = 3, cex = 0.5)
```



I notice that when looking at the 2-dimensional representation, we can see a distinct separation of cities in the USA versus Asia versus Africa. So that representation seems good. However, when I look at the 3-dimensional representation, the clusterings are more difficult to perceive. It might be the scaling of the plot or the challenge to put 3d coordinates on a 2d screen.