1 Let $A \in \mathbb{R}^{m \times n}$ be a square matrix with full rank, rank$(A) = n$. Suppose $A$ admits two singular value decompositions

$$A = U_1 \Sigma V_1^T = U_2 \Sigma V_2^T$$

where $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal with entries being the distinct singular values of $A$, $\sigma_1 > \cdots > \sigma_n > 0$, and $U_1, U_2, V_1, V_2 \in \mathbb{R}^{m \times n}$ are orthogonal matrices.

1a Define $D := \Sigma^T \Sigma$ and $V := V_1^T V_2$. Prove that $D$ and $V$ commute.

Proof

Examine $A^TA$ using each of the two SVDs:

$$A^TA = (U_1 \Sigma V_1^T)^T U_1 \Sigma V_1^T = V_1 \Sigma^T \Sigma V_1^T = V_1 D V_1^T$$

$$A^TA = (U_2 \Sigma V_2^T)^T U_2 \Sigma V_2^T = V_2 \Sigma^T \Sigma V_2^T = V_2 D V_2^T$$

So we have: $V_1 D V_1^T = V_2 D V_2^T$

Since $U_1, U_2, V_1, \& V_2$ are orthogonal, $V_1^{-1} = V_1^T$ and $V_2^{-1} = V_2^T$ so:

$$V_1^T V_1 D V_1^T V_2 = V_1^T V_2 D V_2^T V_2$$

$$\Rightarrow D V_1^T V_2 = V_1^T V_2 D$$

$$\Rightarrow DV = VD$$

$\therefore$ $D$ and $V$ commute. ■

**11b** Based on **11a**, prove that $V$ is a diagonal matrix with entries being either 1 or -1.

Proof

Note, by definition of $D$, $D = \begin{pmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \ddots \sigma_n^2 \end{pmatrix}_{n \times n}$

where $\sigma_1 > \cdots > \sigma_n > 0$.

Thus $D^{-1}$ exists since $\det(D) \neq 0$ since $\sigma_i \neq 0$, and

$$D^{-1} = \begin{pmatrix} 1/\sigma_1^2 & & 0 \\ & 1/\sigma_2^2 & \\ 0 & & \ddots 1/\sigma_n^2 \end{pmatrix}_{n \times n}$$

From **11a**: $DV = VD$ so:

$$V = D^{-1} V D$$

Index-wise:

$$V_{ij} = \sum_k (D^{-1}V)_{ik} D_{kj}$$

$$= \sum_{k,l} D^{-1}_{il} V_{lk} D_{kj}$$

Now, since $D$ is diagonal, $D_{ij} = \delta_{ij} \sigma_i^2$ and $D^{-1}_{ij} = \delta_{ij} \frac{1}{\sigma_i^2}$

So:

$$V_{ij} = \sum_{k,l} \delta_{il} \delta_{kj} \frac{\sigma_j^2}{\sigma_i^2} V_{lk}$$

$$\rightarrow V_{ij} = \frac{\sigma_j^2}{\sigma_i^2} V_{ij}$$

Since $\sigma_i \neq \sigma_j$ for $i \neq j$, we know $\frac{\sigma_j^2}{\sigma_i^2} \neq 1$ for $i \neq j$.

Thus, for $i \neq j$, $V_{ij} = 0$.

Therefore $V$ is diagonal.

Notice, by the definition of $V$: $V(V_1^T V_2)^T = V_1^T V_2 V_2^T V_1 = I$

Thus $V^{-1}$ exists with $V^{-1} = V^T = V$ since $V$ is diagonal.

Next Page

**11b cont.**

So, $V^2 = I$. In index notation this is:

$$\sum_k V_{ik} V_{kj} = \delta_{ij} = I_{ij}$$

Since $V$ is diagonal, $V_{ij} = V_{ji} = \delta_{ij} V_i$ $\left( V = \begin{pmatrix} v_1 & & 0 \\ & v_2 & \\ 0 & & \ddots \; v_n \end{pmatrix}_{n \times n} \right)$

So:

$$\sum_k \delta_{ik} \delta_{kj} v_i v_j = \delta_{ij}$$

$$\rightarrow \delta_{ij} v_i v_j = \delta_{ij}$$

$$\Rightarrow v_i^2 = 1$$

$$\Rightarrow v_i = \pm 1$$

Thus $V$ is diagonal with entries either $+1$ or $-1$. ∎

**11c** Prove that $U_2 = U_1 V$ and $V_2 = V_1 V$

*Proof*

By definition of $V$:

$$V_1 V = V_1 V_1^T V_2 = V_2 \quad \text{since } V_1 \text{ is orthogonal}$$

Thus $V_2 = V_1 V$

Now consider:

$$A = U_1 \Sigma V_1^T = U_2 \Sigma V_2^T$$

$$= U_1 \Sigma V_1^T = U_2 \Sigma (V_1 V)^T$$

$$= U_1 \Sigma V_1^T = U_2 \Sigma V^T V_1^T$$

Next Page

**1c cont.** Since $\Sigma$ and $V$ (and therefore $V^T$) are diagonal, they commute, so:

$$A = U_1 \Sigma V_1^T = U_2 V^T \Sigma V_1^T$$

Since $\sigma_1 > \cdots > \sigma_n > 0$, we know $\Sigma^{-1} = \begin{pmatrix} 1/\sigma_1 & & 0 \\ & 1/\sigma_2 & \\ 0 & & \ddots \; 1/\sigma_n \end{pmatrix}_{n \times n}$

Right multiplying the above eq. by $V_1 \Sigma^{-1}$ yields:

$$U_1 \Sigma V_1^T V_1 \Sigma^{-1} = U_2 V^T \Sigma V_1^T V_1 \Sigma^{-1}$$

$$\rightarrow U_1 \Sigma \Sigma^{-1} = U_2 V^T \Sigma \Sigma^{-1}$$

$$\rightarrow U_1 = U_2 V^T$$

As shown before $V^{-1} = V^T$ so right multiplying by $V$ yields:

$$U_2 = U_1 V .$$

■

[2] Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with distinct positive eigenvalues $\lambda_1 > \cdots > \lambda_n > 0$.

[2a] Prove that the singular values of $A$ are exactly $\lambda_1 \cdots \lambda_n$

Proof

Recall the definition of singular value:

$\sigma_i$ is a singular value of $A$ when $\sigma_i = \sqrt{\gamma_i}$

where $\gamma_i$ is an eigenvalue of $A^TA$.

Let $A = U\Lambda U^T$ be the eigenvalue decomposition of $A$.

Since $A$ is symmetric, $A = A^T$, so:

$$A^TA = AA^T = A^2 = U\Lambda U^T U \Lambda U^T = U\Lambda^2 U^T \quad \text{since } U \text{ is orthogonal.}$$

where

$$\Lambda^2 = \begin{pmatrix} \lambda_1^2 & & \\ & \lambda_2^2 & 0 \\ 0 & \ddots & \\ & & \lambda_n^2 \end{pmatrix}_{n \times n}$$

Thus, the $i^{th}$ eigenvalue of $A^TA$, $\gamma_i$, is $\lambda_i^2$.

So the $i^{th}$ singular value of $A$ is:

$$\sigma_i = \sqrt{\gamma_i} = \sqrt{\lambda_i^2} = |\lambda_i| = \lambda_i \quad \text{since } \lambda_i > 0$$

■

[2b] If $A = U\Sigma V^T$ is a SVD of $A$, prove $U = V$.

Proof

Let $A = U_1 \Sigma V_1^T$ be the SVD of $A$, and

let $A = U_2 \Lambda V_2^T$ be the EVD of $A$ where $V_2 = U_2$

From [2a] we know $\Sigma = \Lambda$ so:

$$A = U_1 \Sigma V_1^T = U_2 \Sigma V_2^T \quad \text{where } V_2 = U_2$$

From [1c], we know then,

$$\begin{cases} V_1 V = V_2 \\ U_1 V = U_2 \end{cases} \quad \text{where } V = V_1^T V_2$$

Using the fact that $V_2 = U_2$ here, we then have:

$$V_1 V = U_1 V$$

Recall, from before that $V^{-1} = V^T$ so right multiplying by $V^T$ yields:

$$U_1 = V_1$$

■

[3] Let $x^{(1)}, \dots, x^{(n)} \in \mathbb{R}^p$ be $n$ sample points with $p$ features. Form the sample matrix $X = [x^{(1)}, \dots, x^{(n)}] \in \mathbb{R}^{p \times n}$ and the kernel matrix $K := X^T X$ as usual. We assume $K$ is positive definite with distinct positive eigenvalues $\lambda_1 > \cdots > \lambda_n > 0$. In this problem, we look for low-dimensional representations $y^{(1)}, \dots, y^{(n)} \in \mathbb{R}^d$ $(d < p)$ such that the kernel matrix $\tilde{K} := Y^T Y$ best approximates $K$; here $Y = [y^{(1)}, \dots, y^{(n)}] \in \mathbb{R}^{d \times n}$. To this end, we consider the following minimization problem

$$\min_{Y \in \mathbb{R}^{d \times n},\ \operatorname{rank}(Y) = d} \| Y^T Y - K \|_F^2$$

[3a] Define $Z := Y^T Y$. Prove that $\operatorname{rank}(Z) = \operatorname{rank}(Y)$

Proof 1

Let $Y = U \Sigma V^T$ be the SVD of $Y$

then $Z = Y^T Y = (U \Sigma V^T)^T U \Sigma V^T = V \Sigma^2 V^T$ { since $U$ & $V$ are orthogonal and $\Sigma$ is diagonal

Note: $\operatorname{rank}(Y) = \operatorname{rank}(\Sigma)$

and $\operatorname{rank}(\Sigma^2) = \operatorname{rank}(\Sigma)$

and $\operatorname{rank}(Z) = \operatorname{rank}(\Sigma^2)$

So $\operatorname{rank}(Z) = \operatorname{rank}(Y)$ ∎

Proof 2

Recall for any $A \in \mathbb{R}^{n \times n}$, $\operatorname{nullity}(A) + \operatorname{rank}(A) = n$

Let $x \in \ker(Y)$, that is $Yx = 0$

Notice then, $\forall x \in \ker(Y)$, $Zx = Y^T Y x = Y^T 0 = 0$

Thus $\ker(Y) \subseteq \ker(Z)$

Now let $x \in \ker(Z)$, so $Zx = Y^T Y x = 0$

right multiplying by $x^T$ gives $x^T Y^T Y x = 0 \Rightarrow \|Yx\|^2 = 0 \Rightarrow Yx = 0$

Thus $\ker(Z) \subseteq \ker(Y)$ so $\ker(Z) = \ker(Y)$

$\therefore$ $\operatorname{nullity}(Z) = \operatorname{nullity}(Y)$ and so $\operatorname{rank}(Z) = \operatorname{rank}(Y)$. ∎

[3b] From [3a], it remains to consider

$$\min_{Z \in \mathbb{R}^{n\times n},\ \text{rank}(Z)=d,\ Z=Z^T,\ Z \text{ is positive semi-definite}} \|Z-K\|_F^2$$

Let $K_d$ be a matrix of rank $d$ that best approximates $K$ with respect to the Frobenius norm. Prove that $K_d$ is a minimizer of the above problem.

Proof

We have $K_d = \underset{X \in \mathbb{R}^{n\times n},\ \text{rank}(X)=d}{\text{argmin}} \|X-K\|_F^2$

Recall from PCA, if $K = U\Sigma V^T$ is the SVD of $K$, then

$$K_d = U\Sigma_d V^T \text{ where } \Sigma_d = \begin{pmatrix} \sigma_1 & & & & \\ & \sigma_2 & \cdots & 0 & \\ & \vdots & \sigma_d & & \\ & 0 & & 0 & \\ & & & & \ddots \end{pmatrix}_{n\times n}$$

Where $\sigma_1 \cdots \sigma_d$ are the $d$ largest singular values of $K$.

and $U$ & $V$ are the same orthogonal matrices in the SVD of $K$.

It remains to show that: - $K_d$ is symmetric

- $K_d$ is positive semi-definite

- $\text{rank}(K_d) = d$

First, $\text{rank}(K_d) = \text{rank}(\Sigma_d) = d$.

Second, since $K$ is symmetric with distinct positive eigenvalues, we know from [2] that $U=V$

So $K_d = U\Sigma_d U^T$. Thus $K_d^T = U\Sigma_d U^T = K_d$

Therefore $K_d$ is symmetric

Finally, we then know by [2] that the singular values of $K_d$ are exactly the eigenvalues of $K_d$. Thus all the eigenvalues of $K_d$ are non-negative, which proves that $K_d$ is positive semi-definite.

■

[3c] Prove that $Y$ can be taken as

$$Y = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_d} \end{pmatrix}_{d\times d} U^T$$

where the columns of $U \in \mathbb{R}^{n\times d}$ are the eigenvectors of $K$ associated with the $d$ largest eigenvalues $\lambda_1, \dots, \lambda_d$, respectively.

Proof

We want to show that

$$K_d = Y^T Y \quad \text{with} \quad Y = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_d} \end{pmatrix}_{d\times d} U^T$$

Recall from the previous problem:

$$K_d = V \Sigma_d V^T \quad \text{where} \quad \Sigma_d = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_d \\ & 0 & \end{pmatrix}_{n\times n} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \\ & 0 & \end{pmatrix}_{n\times n}$$

where $\sigma_i = \lambda_i$ are the singular/eigen-values of $K$, and the columns of $V \in \mathbb{R}^{n\times n}$ are the $n$ eigenvectors of $K$ associated with $\lambda_1, \dots, \lambda_n$ respectively

Index-wise, this is:

$$(K_d)_{ij} = \sum_{k=1}^{n} (V\Sigma_d)_{ik} (V^T)_{kj} = \sum_{k,l=1}^{n} V_{il} (\Sigma_d)_{lk} V_{jk}$$

Now, since $\Sigma_d$ is diagonal, $(\Sigma_d)_{lk} = \begin{cases} \delta_{lk}\sigma_k = \delta_{lk}\lambda_k & \text{for } k \le d \\ 0 & \text{for } k > d \end{cases}$

So $$(K_d)_{ij} = \sum_{k=1}^{d} V_{ik} \lambda_k V_{jk} + \sum_{k=d+1}^{n} 0 = \sum_{k=1}^{d} V_{ik} \lambda_k V_{jk} \quad (*)$$

Similarly, we have $Y^T Y = U \Lambda_d U^T$ where $\Lambda_d = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}_{d\times d}$

and the columns of $U$ are the $d$ eigenvectors of $K$ associated with the $d$ largest eigenvalues of $K$

Next Page

**3c)** Index-wise, we have:

$$(V^TV)_{ij} = \sum_{k=1}^{d} (U\Lambda_d)_{ik} U_{jk} = \sum_{k,l=1}^{d} U_{il} (\Lambda_d)_{lk} U_{jk}$$

Since $\Lambda_d$ is diagonal, $(\Lambda_d)_{lk} = \delta_{lk}\lambda_k$ so:

$$(V^TV)_{ij} = \sum_{k=1}^{d} U_{ik}\lambda_k U_{jk}$$

Finally, since the first $d$ columns of $V$ are equal to the first $d$ columns (all the columns) of $U$, we know

$$U_{ik} = V_{ik} \quad \forall i = 1,\dots,n \quad \forall k = 1,\dots,d$$

Thus, from ⊛, we can write:

$$(K_d)_{ij} = \sum_{k=1}^{d} V_{ik}\lambda_k V_{jk} = \sum_{k=1}^{d} U_{ik}\lambda_k U_{jk} = (V^TV)_{ij}$$

Thus $K_d = V^TV$.

■

# CMSE 820 HW 7 Problem 4

Patrick D. Cook

November 2, 2020

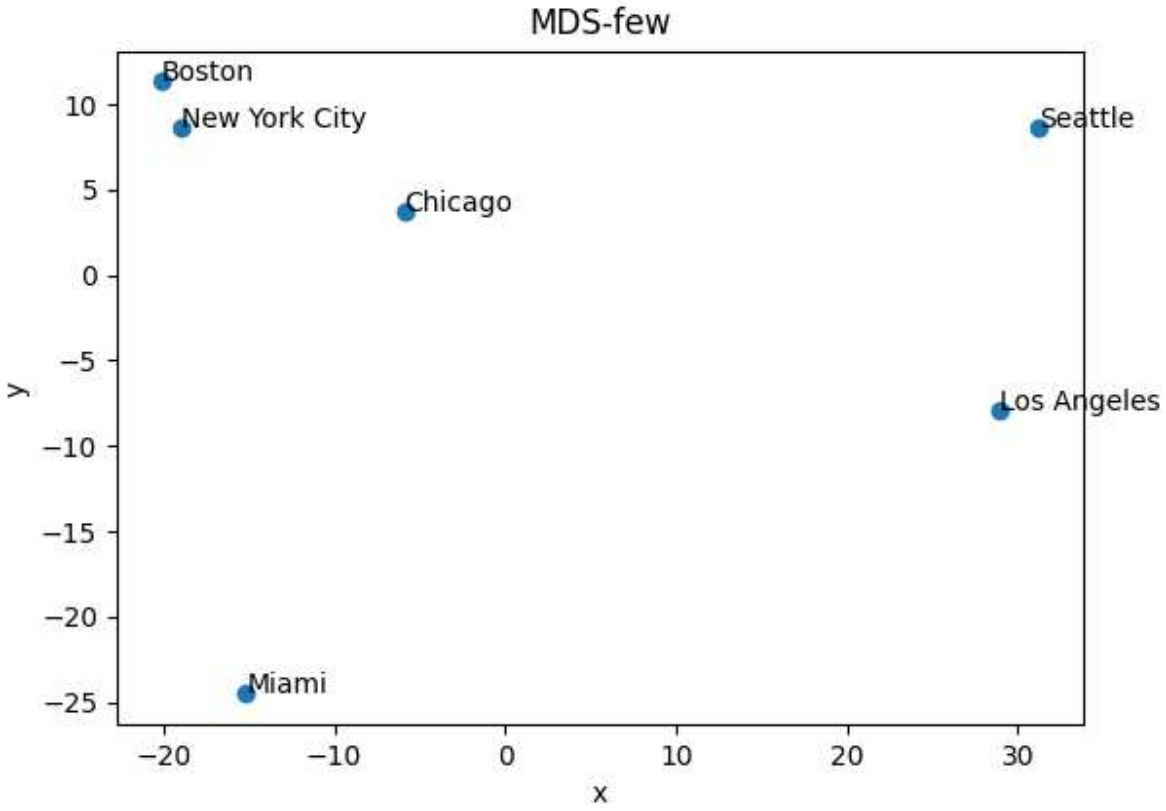


Figure 1: Output of MDS on the pairwise distances of selected US cities.

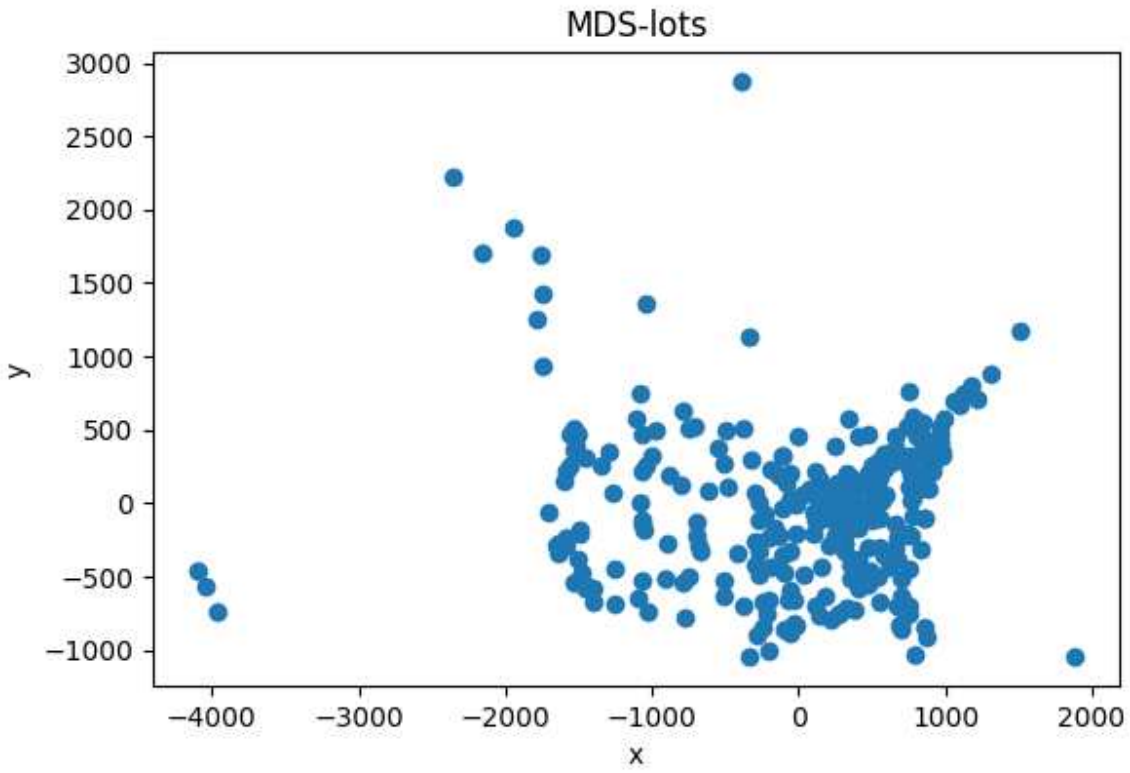


Figure 2: Output of MDS on a list of pairwise distances between 312 US cities[2]. The y-component of the data has been flipped to orient the graph.

[2]https://people.sc.fsu.edu/ jburkardt/datasets/cities/cities.html

# 1 Code

```
"""
Patrick Cook
MSU CMSE 820 - Dr. Yang
11/02/2020
"""

"""
For this problem, submit both source code and the generated output.
Identical source code from different people will receive no point.

    (a) Implement the MDS algorithm as in the lecture notes. You can program in
        any language, but only existing functions for singular value
        decomposition and eigenvalue decomposition are allowed. Using existing
        MDS functions from machine learning packages such as scikit-learn will
        receive no point.

    (b) Measure the pairwise driving distances on Google map between two of the
        following cities: Boston, Chicago, Los Angeles, Miami, New York City,
        and Seattle. Run your code in (a) to make a scattered plot of those
        cities using 2 features.
"""

import numpy as np

def generatePairwiseDistances(X):
    """
        Generate paiwise distances from a set of sample points

        Arguments:
            X [nxp array of floats] - the n sample points, each with p features
                                      each ROW is a sample point

        Returns:
            d [nxn array of floats] - the pairwise distances between the points
    """

    n, _ = np.shape(X)
```

```
    d = np.zeros((n,n))

    for i, x_i in enumerate(X):
        for j, x_j in enumerate(X):
            d[i,j] = np.sqrt(np.dot(x_i-x_j,x_i-x_j))

    return d

def formKernelFromD(D):
    """
        Form the kernel matrix from the square distance matrix D

        Arguments:
            D [nxn array of floats] - the square distance matrix

        Returns:
            K [nxn array of floats] - the kernel matrix
    """

    n, _ = np.shape(D)
    H = (np.identity(n)-(1/n)*np.ones((n,n))) # centering matrix

    return -0.5*np.dot(np.dot(H,D),H)

def MDS(D, d = 2,):
    """
        Multidimensional Scaling

        Arguments:
            D [nxn array of floats] - the square distance matrix of the data,
                                      must be symmetric, with 0 on the
                                      diagonal

            d [int] - the number of artificial features per sample point

        Returns:
            X [nxd array of floats] - the d artificial features of each sample
                                      point
    """
```

```
    # check that D is symmetric
    if not (D==D.T).all():
        raise RuntimeError("D must be symmetric.")

    # check that D is has zeros on the diagonal
    if not (np.diagonal(D)==0).all():
        raise RuntimeError("D must have 0 for each element on the diagonal.")

    n, _ = np.shape(D)

    # form kernel matrix
    K = formKernelFromD(D)

    # find eigenvalues and orthonormal eigenvectors of K
    #   the COLUMNS of vec are the eigenvectors of K
    val, vec = np.linalg.eigh(K)

    # sort eigenvalues and eigenvectors in descending order
    idx = np.argsort(val)
    val = val[idx][::-1]
    vec = vec[:,idx][:,::-1]

    ## floating point error causes eigenvalues that should be 0 to be slightly
    ##   negative sometimes, so we handle that here
    #epsilon = 1e-10
    #val[np.abs(val)<1e-10]=0.

    # calculate principal components
    X = np.dot(np.sqrt(np.diag(val[:d])),vec[:,:d].T)

    return X

"""
    Below I use MDS to generate a graph of the specified cities from their
    pairwise DRIVING distances. I also use MDS to generate a map of 312 cities
    in the US and Canada from their pairwise "ACTUAL"* distances.

    * Not accounting for the curvature of the Earth
"""
```

```
# import necessary package
import matplotlib.pyplot as plt

# pairwise DRIVING distance data for selected cities (in miles), squared to get
#   the square distance matrix

           # Boston, Chicago, LA, Miami, NYC, Seattle
D = np.array([[   0,  982,  2983, 1490,  218, 3024], # Boston
              [ 982,    0,  2015, 1377,  789, 2064], # Chicago
              [2983, 2015,     0, 2733, 2793, 1136], # LA
              [1490, 1377,  2733,    0, 1276, 3300], # Miami
              [ 218,  789,  2793, 1276,    0, 2852], # NYC
              [3024, 2064,  1136, 3300, 2852,    0]])# Seattle

# apply MDS
X = MDS(D)
# plot
plt.figure()
plt.scatter(X[0,:],X[1,:])

# annotate cities
plt.annotate(xy=X[:,0], s="Boston")
plt.annotate(xy=X[:,1], s="Chicago")
plt.annotate(xy=X[:,2], s="Los Angeles")
plt.annotate(xy=X[:,3], s="Miami")
plt.annotate(xy=X[:,4], s="New York City")
plt.annotate(xy=X[:,5], s="Seattle")

# format
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel("x")
plt.ylabel("y")
plt.title("MDS-few")
# save figure
plt.savefig("MDS-few.png")

# load pairwise ACTUAL distance data for 312 US and Canadian cities, and square
#   it to get the square distance matrix
D = np.loadtxt("City-to-City.txt")**2
# apply MDS
```

```
X = MDS(D)
# plot
plt.figure()
plt.scatter(X[0,:],-X[1,:]) # ARBITRARILY INVERT Y AXIS TO ORIENT GRAPH
# format
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel("x")
plt.ylabel("y")
plt.title("MDS-lots")
# save figure
plt.savefig("MDS-lots.png")

# show the plots
plt.show()
```