MATH5473 HW1 By XIA Wencan

In [1]:
```
import numpy as np
import numpy.linalg as alg
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import matplotlib.cm as cm
from matplotlib.ticker import MaxNLocator
```

(a) Set up data matrix $X = (x_1, \ldots, x_n) \in \mathcal{R}^{p \times n}$

In [2]:
```
data = pd.read_csv(r'C:\Users\kosta\Jupyter Notes\MATH5473\HW1\train7.csv')
X = np.array(data)
# Transpose X
X = X.T
p = np.size(X,0)
N = np.size(X,1)
(p,N)
```

Out[2]:
```
(256, 644)
```

(b) Compute sample mean

In [3]:
```
mu = np.mean(X, axis=0)
e_mu = np.tile(mu,(p,1))
X1 = X - e_mu
```

(C) SVD

In [4]:
```
K = 20
u, s, v = alg.svd(X1)
print('The top K eigen value of SVD is', [s[i] for i in range(K)])
```

```
The top K eigen value of SVD is [213.1028054513784, 86.52866446947652, 69.03940680
506211, 63.783287997054686, 45.372011549134434, 40.075414536059675, 37.23019023609
938, 33.99130402927157, 32.217677310968256, 30.60695628571329, 30.40756273896366,
27.619761338324693, 23.546973623538417, 22.946377287485504, 21.589029639924817, 2
1.126075435284424, 20.469100238882444, 20.06005816704625, 19.15728601177822, 18.23
3854847225313]
```

(d) Plot eigenvalue curve

In [5]:
```
# Calculate the covariance matrix
cov_X = (X1 @ X1.T)/N
tr = np.trace(cov_X)
print(cov_X)
# calculate eig value
eigen_values, eigen_vectors = alg.eig(cov_X)
```

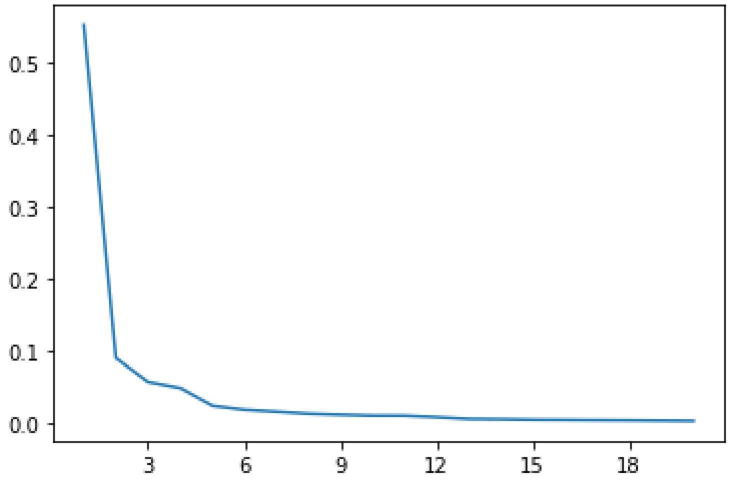
```
# Making eigen tuples
eigen_pairs = [ (np.abs(eigen_values[i]), eigen_vectors[:, i]) for i in range(len(

# Sort the tuples
eigen_pairs.sort(key = lambda eigen_pairs: eigen_pairs[0])
eigen_pairs.reverse()

# Visualize the eigenvalues
t = [i+1 for i in range(K)]
y = [eigen_values[i] for i in range(K)]/tr
plt.plot(t, y)
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.show()
```

```
[[0.1801248  0.15257622 0.09487202 ... 0.1750388  0.1750388  0.1750388 ]
 [0.15257622 0.22538202 0.19249783 ... 0.12692913 0.12692913 0.12692913]
 [0.09487202 0.19249783 0.28283074 ... 0.07473296 0.07473296 0.07473296]
 ...
 [0.1750388  0.12692913 0.07473296 ... 0.18854689 0.18854689 0.18854689]
 [0.1750388  0.12692913 0.07473296 ... 0.18854689 0.18854689 0.18854689]
 [0.1750388  0.12692913 0.07473296 ... 0.18854689 0.18854689 0.18854689]]
```

```
C:\Users\kosta\anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:1298: Comp
lexWarning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
```

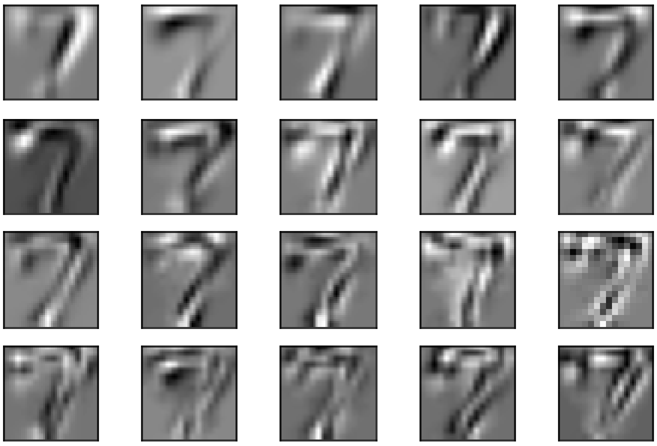


(e) Visualize top-20 left singular vector

In [6]:
```
plt.figure()
cmap  = cm.gray_r
for i in range(1,21):
    pc = u[:, i]
    pc_matrix = np.reshape(pc, (16, 16))

    plt.subplot(4, 5, i)
    plt.imshow(pc_matrix, cmap=cmap)
    plt.xticks([])
    plt.yticks([])

plt.show()
```



(f) Order the images

In [7]:

```
v1 = eigen_vectors[:, 0]
imgpro = np.dot(X.T, v1)
imgpro_tuple = [(i, imgpro[i]) for i in range(N)]
imgpro_tuple.sort(key= lambda imgpro_tuple: imgpro_tuple[1])
print('imgpro are sorted by value imgpro[:,1], index is in imgpro[:,0]')
```

```
imgpro are sorted by value imgpro[:,1], index is in imgpro[:,0]
```

(g) Scatter plot

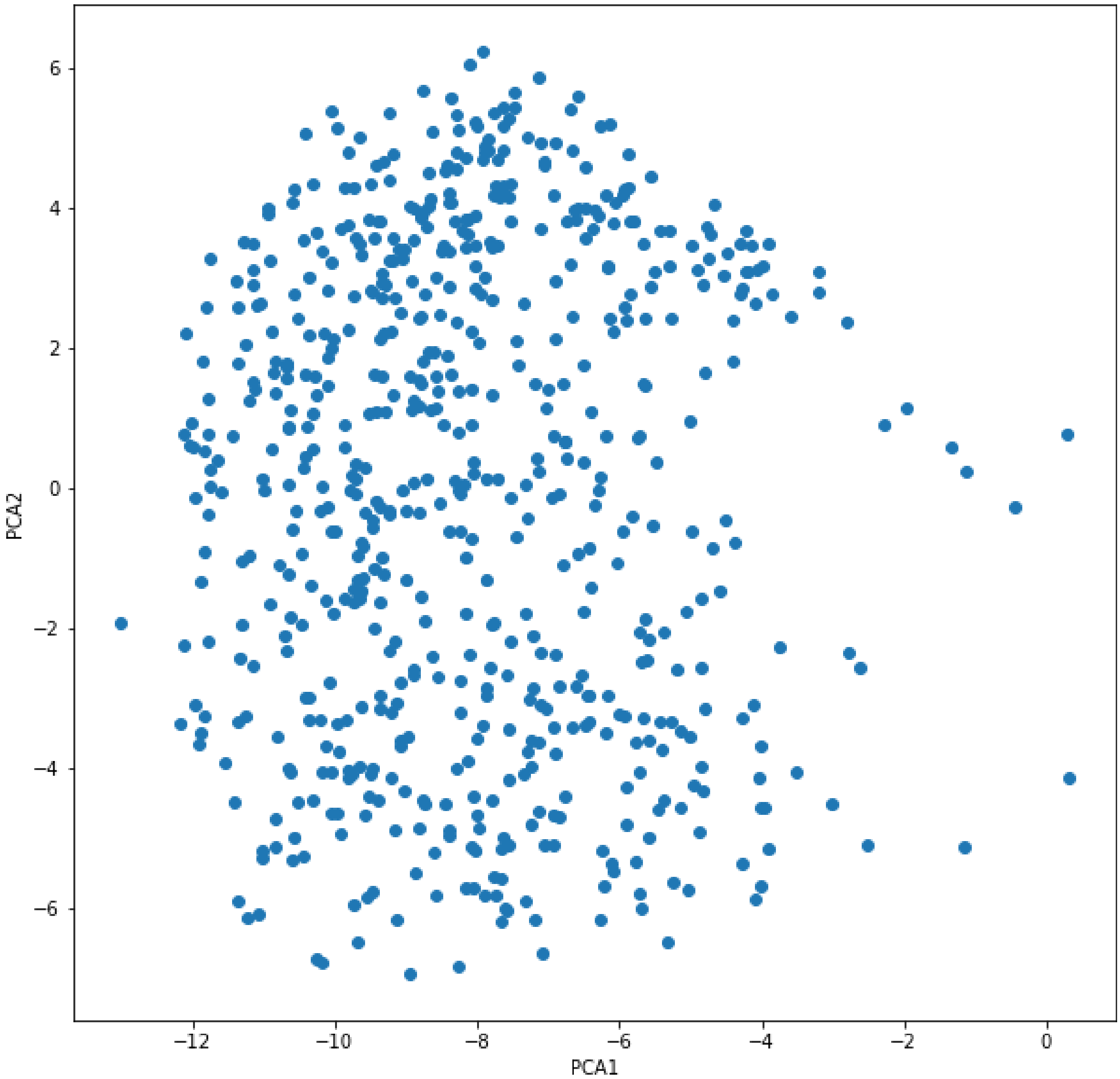
In [8]:

```
pc1 = eigen_pairs[0][1][:, np.newaxis]
pc2 = eigen_pairs[1][1][:, np.newaxis]

W = np.hstack((pc1, pc2))
#print(W.shape)
#print(X.shape)
X_pca = np.dot(X.T, W)

plt.figure(figsize=(10, 10))
x = np.array(X_pca[:, 0])
y = np.array(X_pca[:, 1])
plt.scatter(x, y)
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title = ('Samples projected onto 2-D by PCA')
plt.show()
```

```
C:\Users\kosta\anaconda3\lib\site-packages\matplotlib\collections.py:200: ComplexW
arning: Casting complex values to real discards the imaginary part
  offsets = np.asanyarray(offsets, float)
```



In [ ]:

In [1]:
```python
import numpy as np
import pandas as pd
import numpy.linalg as alg
import scipy as sp
import matplotlib.pyplot as plt
```

(a) Data collection

In [2]:
```python
data = pd.read_csv(r'C:\Users\kosta\Jupyter Notes\MATH5473\HW1\distance2.csv')
Data = np.array(data)
n = len(Data)
cities = np.array(data.columns)
print(data)
```

```
   Beijing  Shanghai  Guangzhou  Hongkong  Chengdu   London  Bangkok
0     0.00   1068.00    1890.00   1974.00  1516.00  8138.09  3297.79
1  1068.00      0.00    1206.63   1227.83  1658.00  9196.34  2886.89
2  1890.00   1206.63       0.00    129.07  1238.00  9497.75  1702.96
3  1974.00   1227.83     129.07      0.00  1369.30  9626.00  1725.68
4  1516.00   1658.00    1238.00   1369.30     0.00  8279.42  1916.29
5  8138.09   9196.34    9497.75   9626.00  8279.42     0.00  9532.18
6  3297.79   2886.89    1702.96   1725.68  1916.29  9532.18     0.00
```

(b) MDS

In [3]:
```python
def mds(D, dim=[]):
    H = -np.ones((n, n))/n
    H = -H.dot(D ** 2).dot(H)/2
    evals, evecs = alg.eigh(H)

    # Sort by eigenvalu in descending order
    idx = np.argsort(evals)[::-1]
    evals = evals[idx]
    evecs = evecs[:, idx]

    #Compute the coordinates using positive eigenvalued components only
    w, = np.where(evals > 0)
    if dim!=[]:
        arr = evals
        w = arr.argsort()[-dim:][::-1]
    if np.any(evals[w]<0):
        print('Error: Not enough positive eigenvalues for the selected dim.')
        return []
    L = np.diag(np.sqrt(evals[w]))
    V = evecs[:, w]
    Y = V.dot(L)
    return Y, evals, evecs
```

(c) Eigenvalue Plot

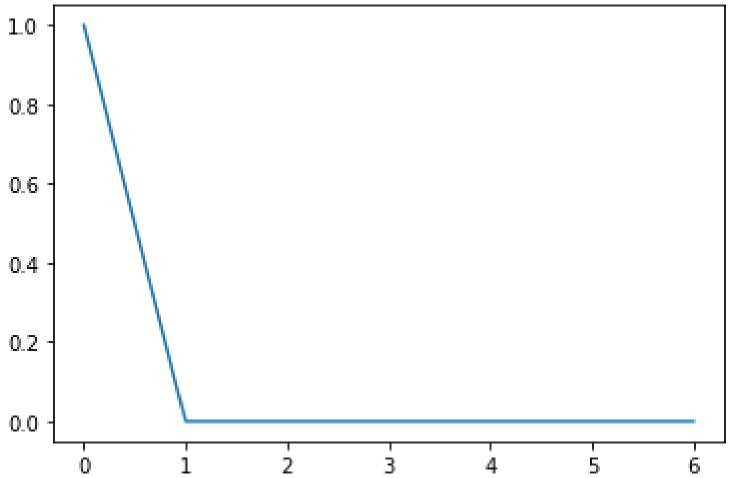
In [8]:
```python
X2, eigen_values, eigen_vectors = mds(Data, dim=2)

normed_eigen_values = eigen_values/np.sum(eigen_values)
print("Normed Eigenvalues are:\n", normed_eigen_values)

plt.figure()
```

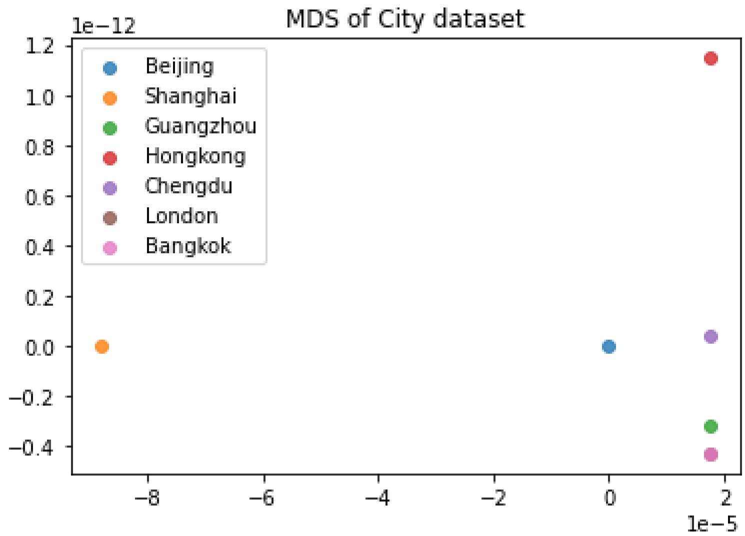
```
plt.plot([n-1-i for i in range(n)], normed_eigen_values)
plt.show()
```

```
Normed Eigenvalues are:
 [-1.20307317e-16 -2.31402840e-32  6.58540852e-65  1.05450821e-48
  4.44077060e-33  2.40614633e-17  1.00000000e+00]
```



(d) Scatter Plot

In [9]:
```
plt.figure()
for i in range(n):
    plt.scatter(X2[i, 0], X2[i, 1], alpha=.8, label=cities[i])
plt.legend()
plt.title('MDS of City dataset')
plt.show()
```



The graph basically reflect the position relationship: Bangkok is west of most cities. London is far away from those cities, so the scatter hasn't show up.

In [ ]:

3. *Positive Semi-definiteness:* Recall that a n-by-n real symmetric matrix K is called positive semi-definite (*p.s.d.* or $K \succeq 0$) iff for every $x \in \mathbb{R}^n$, $x^T K x \geq 0$.

(a) Show that $K \succeq 0$ if and only if its eigenvalues are all nonnegative.

(b) Show that $d_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ is a squared distance function, *i.e.* there exists vectors $u_i, v_j \in \mathbb{R}^n$ $(1 \leq i, j \leq n)$ such that $d_{ij} = \|u_i - u_j\|^2$.

(c) Let $\alpha \in \mathbb{R}^n$ be a signed measure s.t. $\sum_i \alpha_i = 1$ (or $e^T\alpha = 1$) and $H_\alpha = I - e\alpha^T$ be the Householder centering matrix. Show that $B_\alpha = -\frac{1}{2} H_\alpha D H_\alpha^T \succeq 0$ for matrix $D = [d_{ij}]$.

(d) If $A \succeq 0$ and $B \succeq 0$ $(A, B \in \mathbb{R}^{n\times n})$, show that $A + B = [A_{ij} + B_{ij}]_{ij} \succeq 0$ (elementwise sum), and $A \circ B = [A_{ij}B_{ij}]_{ij} \succeq 0$ (Hadamard product or elementwise product).

(a) ($\Rightarrow$) if $K \geq 0$. and K is real symmetric matrix. so

there exists P, s.t. $P^T K P = \text{diag}(\lambda_1, \cdots, \lambda_n)$

$\forall y \in \mathbb{R}^n$. let $x = Py$. and $x^T K x \geq 0$

$\because x^T K x = y^T P^T K P y = \sum_{i=1}^{n} \lambda_i y_i^2 \geq 0$, for $\forall y$

so $\lambda_i \geq 0$, $\forall i$.

($\Leftarrow$) if $\lambda_i \geq 0$ and $K = Q^T \text{diag}(\lambda_1, \cdots, \lambda_n) Q$, $Q = P^{-1}$.

$\forall x \in \mathbb{R}^n$. $x^T K x = (Qx)^T \text{diag}(\lambda_1, \cdots, \lambda_n)(Qx)$

let $y = Qx$ $\quad = \sum_{i=1}^{n} \lambda_i \cdot y_i^2 \geq 0$. so K is SPD.

(b) Since $(e_i - e_j)^T K (e_i - e_j) = k_{ii} + k_{jj} - 2k_{ij}$.

and we have Cholesky decomposition for $K = A^* A$

let $u_i = Ae_i$ and $v_j = Ae_j$.

$d_{ij} = \|u_i - v_j\| = \|A(e_i - e_j)\| = (e_i - e_j)^* A^* A (e_i - e_j) = k_{ii} + k_{jj} - 2k_{ij}$.

(c) let $\Lambda = \text{diag}(k_{11}, \cdots, k_{nn})$. $D = \Lambda e^T + e\Lambda^T - 2K$.

$B_\alpha = -\frac{1}{2} H_\alpha D H_\alpha^T = -\frac{1}{2} H_\alpha (\Lambda e^T + e\Lambda^T - 2K) H_\alpha^T$

$H_\alpha \Lambda e^T H_\alpha^T = (I - e\alpha^T)\Lambda e^T (I - \alpha e^T) = (I - e\alpha^T)\Lambda [e^T - (\sum \alpha_i) \cdot e^T] = 0$

$H_\alpha e \Lambda^T H_\alpha^T = (H_\alpha \Lambda e^T H_\alpha^T)^T = 0$.

Thus. $B_\alpha = H_\alpha K H_\alpha^T$.

$\forall x \in \mathbb{R}^n$. let $y = H_\alpha^T x \in \mathbb{R}^n$ and K is SPD

so $x^T B_\alpha x = y^T K y \geq 0$, implies B_α is SPD.

(d). If $A \geq 0$. $B \geq 0$. $\forall x \in \mathbb{R}^n$

$x^T (A+B) x = x^T A x + x^T B x \geq 0 \quad \Rightarrow A + B \geq 0$

If $B \geq 0$, let $B = P^T P$. $\forall x \in \mathbb{R}^n$.

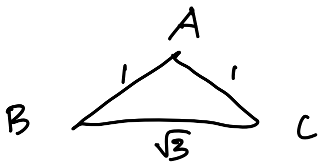
then $x^T (A \circ B) x = x^T A \circ (P^T P) x = \sum_{i,j=1}^{n} x_i \cdot a_{ij} \left(\sum_{m=1}^{n} P_{mi} \cdot P_{mj}\right) x_j$

$$= \sum_{m=1}^{n} (x \circ P_{m\cdot})^T \cdot A \cdot (x \circ P_m) \geq 0 , \quad \text{so} \quad A \circ B \geq 0$$

4. (a) d^2 is not distance function.

suppose $p=2$. $d(A \cdot B) = |\vec{AB}|$ is distance function in R^2.

let $A \cdot B \cdot C$ as follow:



$d^2(A \cdot B) + d^2(A \cdot C) = 2 < 3 = d^2(B, C)$

so triangle ineq. is not satisfied!

(b) $\sqrt{d}$ is a distance function: (denote $\tilde{d}(a,b) = \sqrt{d(a,b)}$)

1) $\tilde{d}(a,b) = \sqrt{d(a,b)} \geq 0$ and $\tilde{d}(a,b) = 0 \Leftrightarrow d(a,b) = 0 \Leftrightarrow a = b$

2) $\tilde{d}(a,b) = \sqrt{d(a,b)} = \sqrt{d(b,a)} = \tilde{d}(b,a)$

3) $\left(\sqrt{d(a,c)} + \sqrt{d(b,c)}\right)^2 = d(a,c) + d(b,c) + 2\sqrt{d(a,c)\,d(b,c)}$

$$\geq d(a,b)$$

thus $\tilde{d}(a,c) + \tilde{d}(b,c) \geq \tilde{d}(a,b)$.

So $\sqrt{d}$ is distance function.