hw1

February 19, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy
```

1 PCA experiments

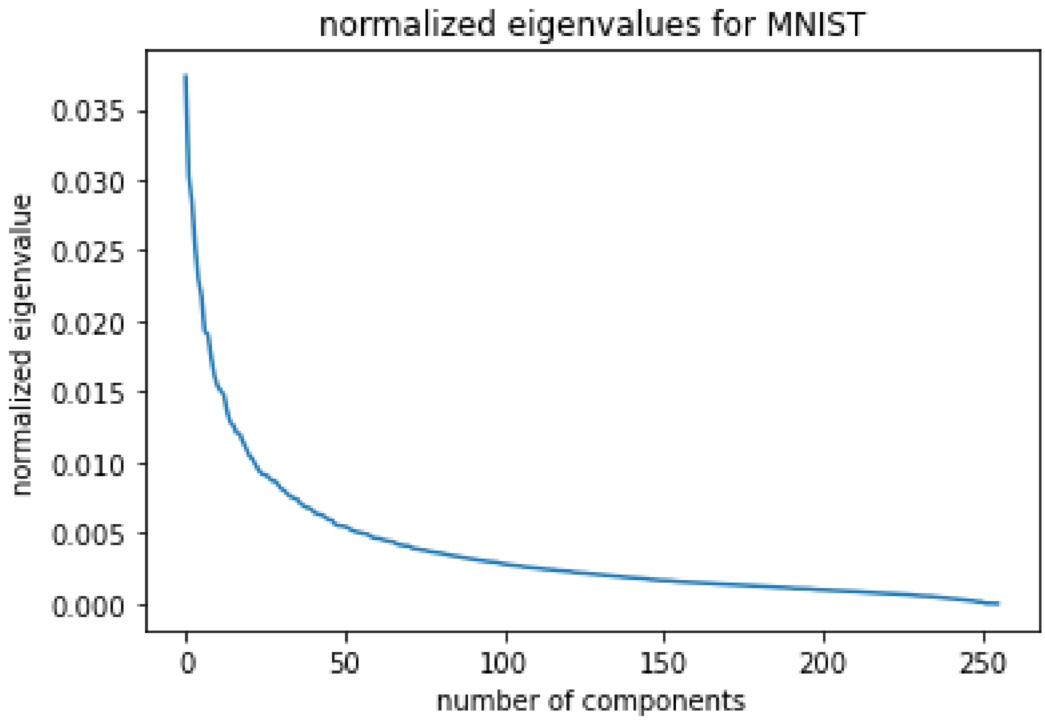
1.1 Loading "2" in MNIST dataset and preprocessing

[2]:
```
eights_img=np.loadtxt('digit2.txt',delimiter=",")
eights_img=eights_img.transpose()

eights=np.loadtxt('digit2.txt',delimiter=",")
eights=eights.transpose()
print(eights.shape)
eights=eights-eights.mean(axis=1).reshape(256,-1)
U, s, Vh = np.linalg.svd(eights)
```

```
(256, 731)
```

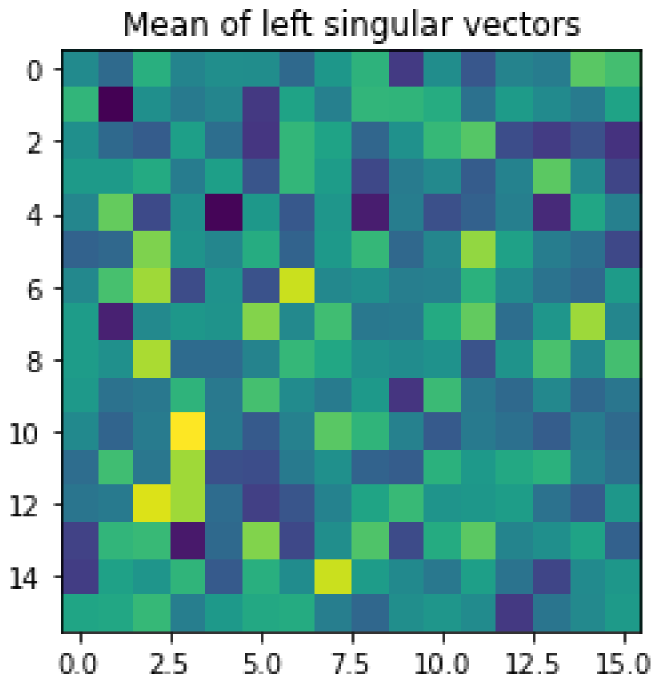
1.2 Plot eigenvalue curve

[3]:
```
plt.plot(s/np.sum(s))
plt.title("normalized eigenvalues for MNIST")
plt.xlabel("number of components")
plt.ylabel("normalized eigenvalue")
plt.show()
```

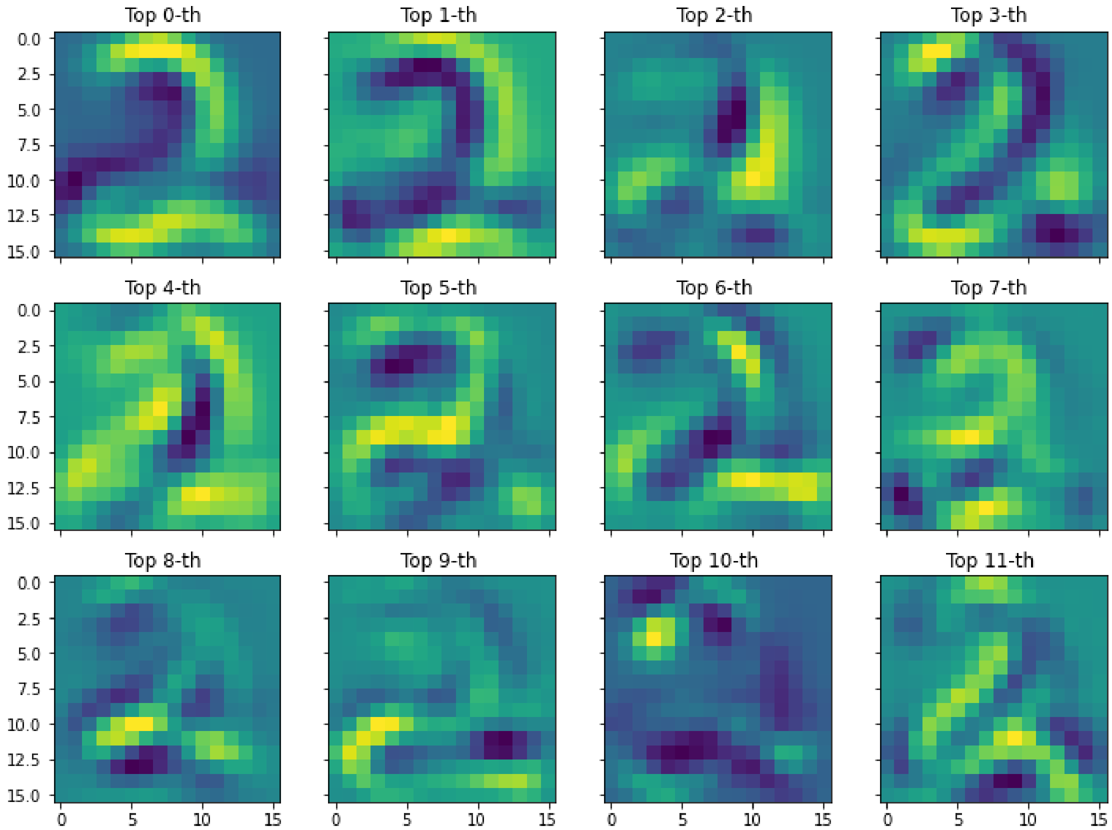


1.3 explore left singular vectors

[4]:
```
plt.imshow(U.mean(axis=1).reshape(16,16))
plt.title("Mean of left singular vectors")
plt.show()
```

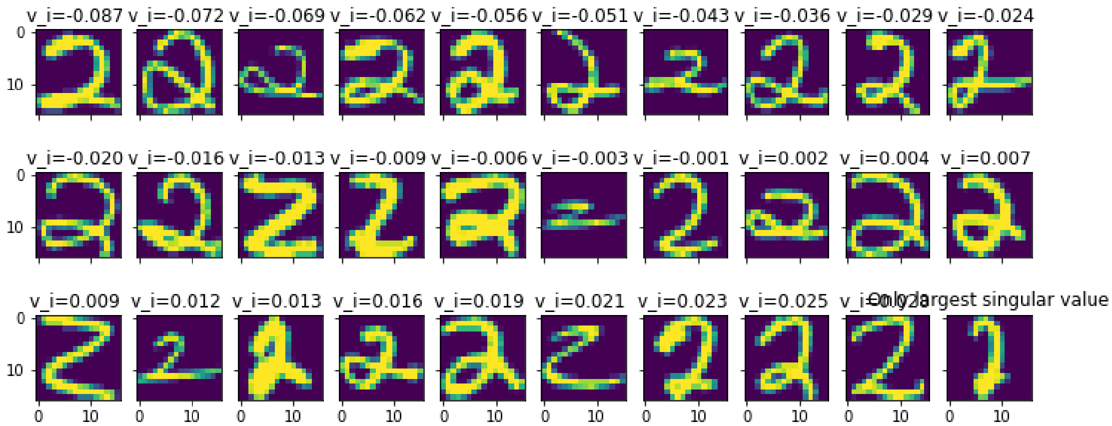


[5]:
```
fig, ax = plt.subplots(3, 4, sharex='col', sharey='row',figsize=(12,9))
index=0
for i in range(3):
    for j in range(4):
        ax[i, j].imshow(U[:,index].reshape(16,16))
        ax[i, j].set_title("Top {}-th ".format(index))
        index+=1
```



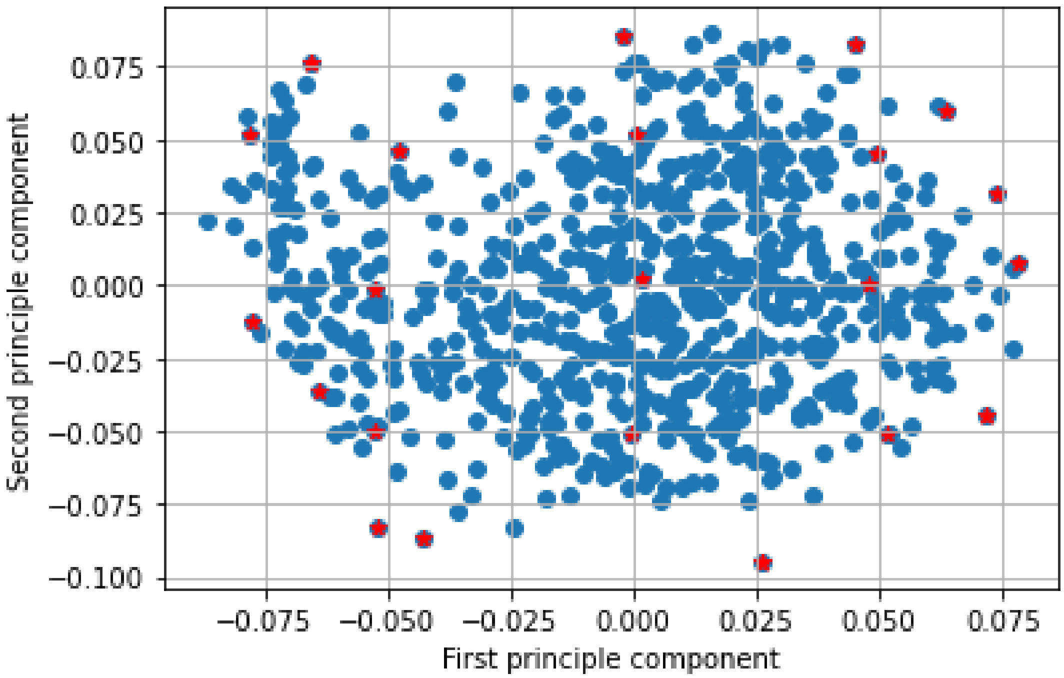
1.4 explore right singular vectors

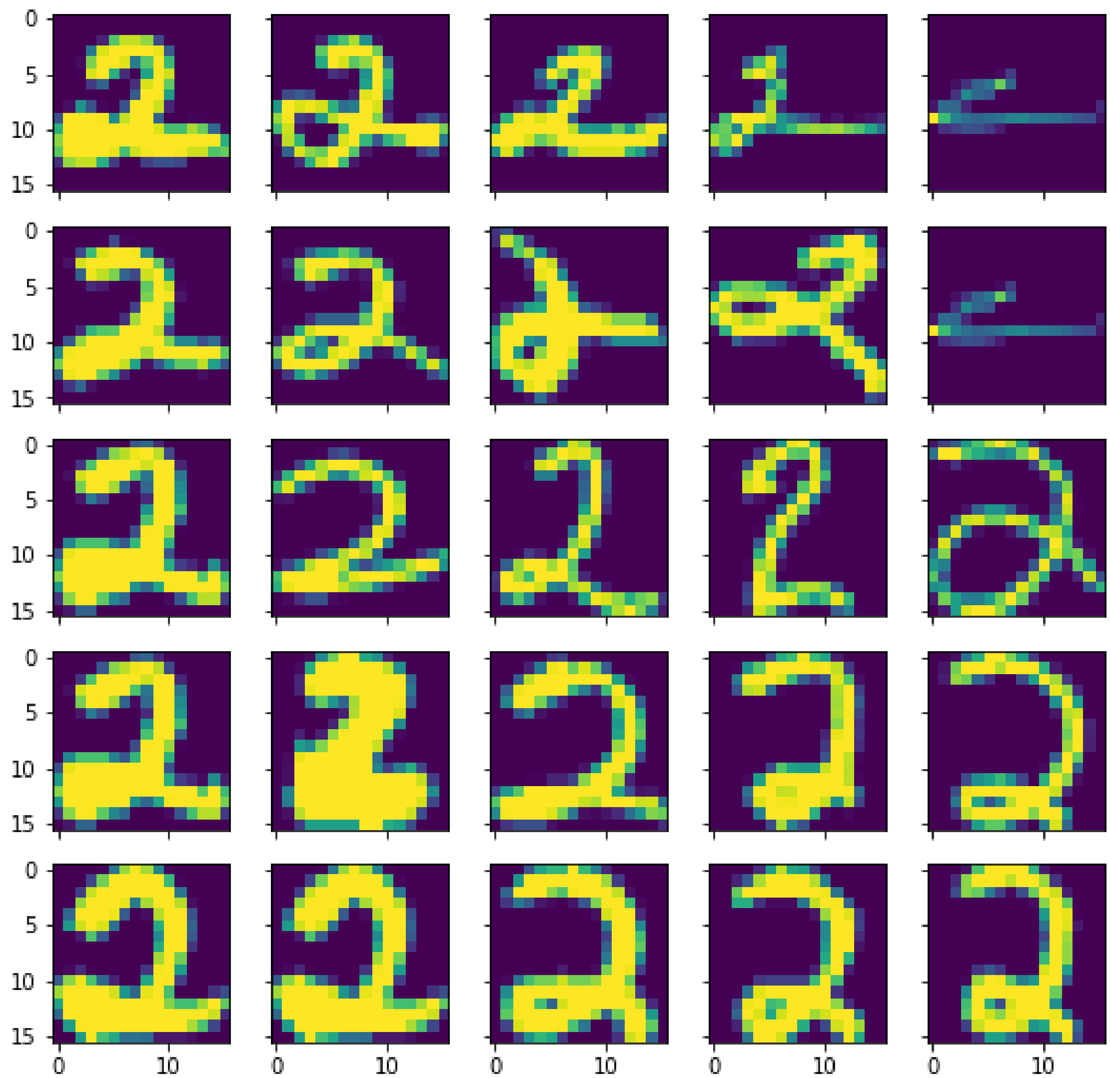
[6]:
```
Vh[:,0].shape
rank=np.argsort(Vh[0,:])

fig, ax = plt.subplots(3, 10, sharex='col', sharey='row',figsize=(12,5))
index=0
for i in range(3):
    for j in range(10):
        ax[i, j].imshow(eights_img[:,rank==index].reshape(16,16))
        ax[i, j].set_title("v_i={:.3f} ".format(Vh[0,:][rank[index]]))
        index+=20
plt.title("Only largest singular value")
plt.show()
```



1.5 Top 2 largest singular value

[7]:
```python
def find_nearest(xy,data):
    distance=[ (data[:,i]-xy).transpose().dot((data[:,i]-xy)) for i in range(data.shape[1])  ]
    return np.argmin(np.array(distance))

indexs=[]
plt.scatter(Vh[0,:],Vh[1,:])
for x in np.linspace(-0.1,0.1,5):
    for y in np.linspace(-0.1,0.1,5):
        index=find_nearest( np.array([x,y]), Vh[0:2,:]  )
        indexs.append(index)
        plt.scatter(Vh[0,index],Vh[1,index],color="r",marker="*")
plt.grid()
plt.xlabel("First principle component")
plt.ylabel("Second principle component")

plt.show()

fig, ax = plt.subplots(5, 5, sharex='col', sharey='row',figsize=(9,9))
index=0
for i in range(5):
    for j in range(5):
        ax[i, j].imshow(eights_img[:,indexs[index]].reshape(16,16))
        # ax[i, j].set_title("{}-th ".format(index))
        index+=1
```


0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
Second principle component
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
First principle component



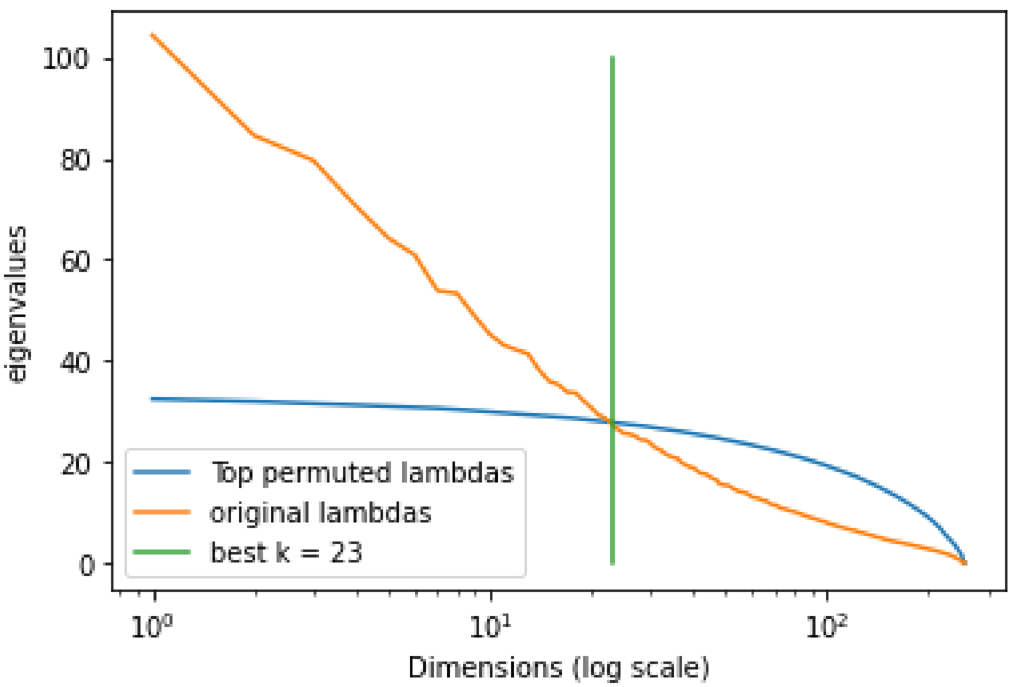
1.6 Parallel analysis

[8]:
```
p_value=0.05
R=100
def permute_data(data):
    #data:p*n
    permuted_data=np.zeros_like(data)
    for j in range(data.shape[0]):
        permuted_data[j,:]=np.random.permutation(data[j,:])
    return permuted_data

def get_lambda(data):
    data=data-data.mean(axis=1).reshape(256,-1)
```

```
    U, s, Vh = np.linalg.svd(data)
    return s
lambdas=[]
for i in range(R):
    permuted_data=permute_data(eights_img)
    lambdas.append(get_lambda(permuted_data))
lambdas=np.array(lambdas)
top_lambdas=np.percentile(lambdas,100-100*p_value,axis=0)

true_lambda=get_lambda(eights_img)
```

[9]:

```
plt.plot( [i+1 for i in range(len(top_lambdas))] ,top_lambdas,label="Top␣
 ↪permuted lambdas")
plt.plot([i+1 for i in range(len(top_lambdas))] ,true_lambda,label="original␣
 ↪lambdas")
best_k=np.sum( top_lambdas< true_lambda)
plt.plot( [best_k,best_k],[0,100] ,label="best k = {}".format(best_k) )
plt.xscale("log")
plt.xlabel("Dimensions (log scale)")
plt.ylabel("eigenvalues")
plt.legend()
plt.show()
```



2 MDS of cities

I choose seven cities: 'hongkong', 'beijing', 'shanghai', 'tokyo', 'taibei', 'chengdu', 'baotou'

I collect their flight distance from https://www.distancecalculator.net/

[10]:
```
D=np.loadtxt('cities.csv')
D=D*D
```

[11]:
```
H=np.eye(D.shape[0])-1/D.shape[0]
```

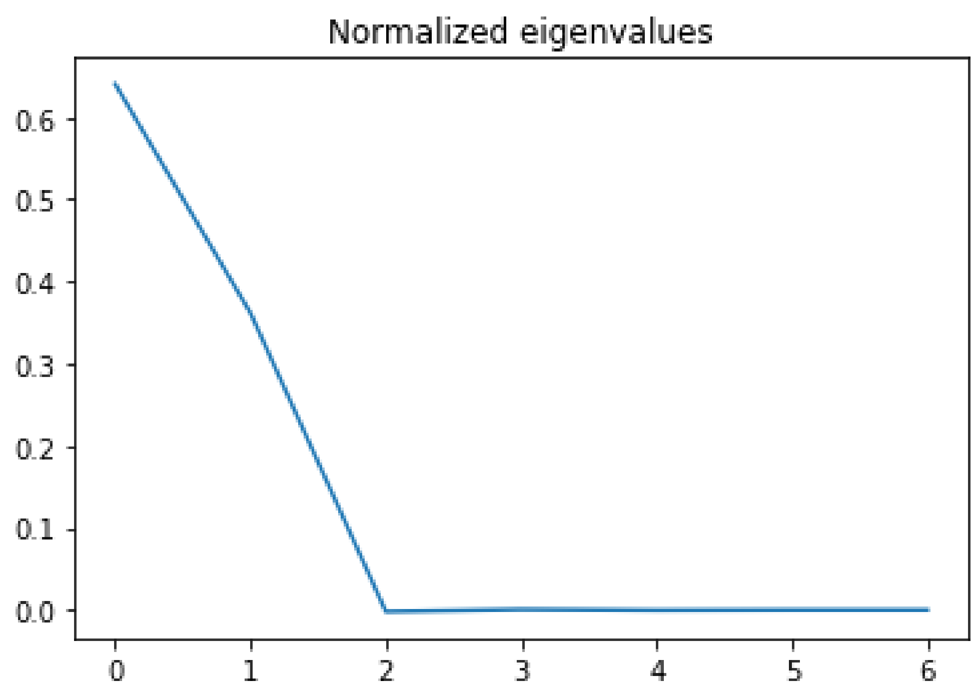
[12]:
```
B=-0.5*H.dot(D.dot(H.transpose()))
```

[13]:
```
Lambda,U=np.linalg.eig(B)
```

2.1 Plot the normalized eigenvalues

I find that the 3-rd and 5-th eigenvalues are negative. I think the reason is that here i use the flight distance which is geodesic distance on a ball. We can can not reconstruct it in Euclidean space.

[14]:
```
plt.plot(Lambda/np.sum(Lambda))
plt.title("Normalized eigenvalues")
plt.show()
print(Lambda/np.sum(Lambda))
```

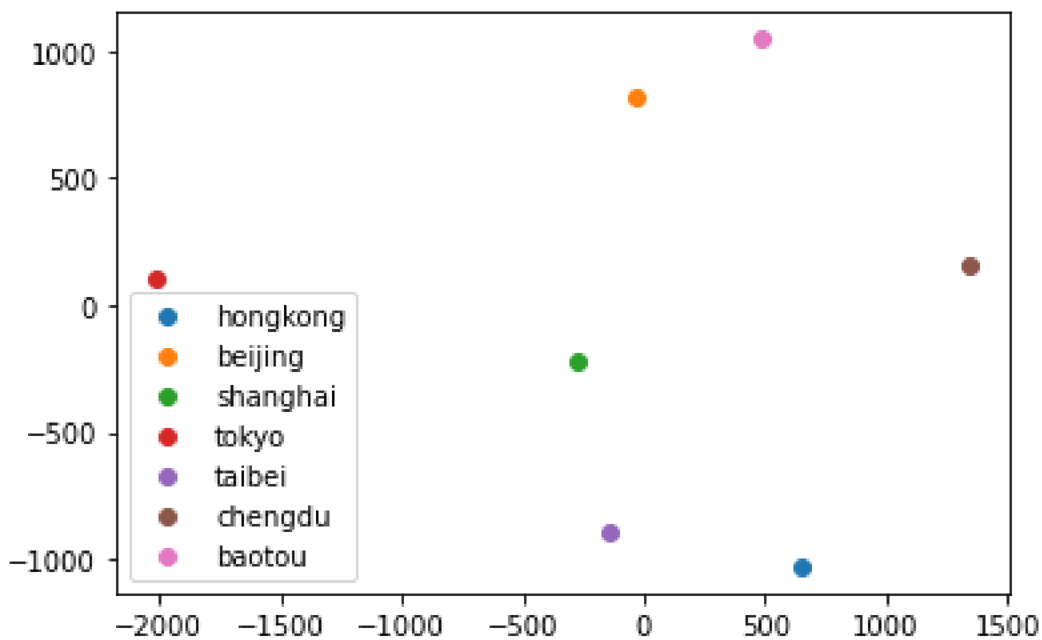


```
[ 6.40398239e-01  3.61396142e-01 -2.09031634e-03  6.34052026e-04
 -3.40448889e-04  3.51379767e-17  2.33159839e-06]
```

2.2 scatter plot of cities

I found that the results obtained are very close to the actual situation. But the horizontal flip occurred. Because horizontal flip and rotation will not change the distance.

[15]:
```
name=[ ␣
↪'hongkong',        'beijing',        'shanghai',        'tokyo',        'taibei',        'c
↪]
X_hat=U.dot(np.diag( np.sqrt(Lambda) ))
for i in range(len(name)):
    plt.scatter(X_hat[i,0],X_hat[i,1],label=name[i])
plt.legend()
plt.show()

print(Lambda)
```




```
[ 6.57749540e+06  3.71188007e+06 -2.14695252e+04  6.51231378e+03
 -3.49673198e+03  3.60900243e-10  2.39477199e+01]
```