INF552 Machine Learning for Data Informatics HW3

1. Contribution:

a. Shih-Yu Lai (50%):

Email: shihyula@usc.edu

USC-ID: 7183984563

Implement FastMap code and write the ReadMe.

b. Shiuan-Chin Huang (50%):

Email: shiuanch@usc.edu

USC-ID: 9815633745

Implement PCA code and write the ReadMe.

2. Steps of the algorithm and data structure

In PCA.py, we follow the dimensional reduction steps we learned in class:

Step 1: Normalize data points(x' = x - mean).

Step 2: Compute covariance matrix.

Step 3: Computer eigenvalue and eigenvectors.

Step 4: Use the largest 2 eigenvalues to calculate our dimensional reduction result.

In FastMap.py, we follow the below steps to implement our algorithm:

Step 1: Read data from fastmap-data.txt and fastmap-wordlist.txt and store in the numpy array.

Step 2: Find all objects in the array (Object 1 to Object 10).

Step 3: Find objects which have furthest distance. (get Object A and Object B and max distance)

Step 4: Find first dimension Xi. (will generate 10 X for 10 objects)

formula: (dai^2 + dab^2 – dbi^2) / 2dab

Step 5: Find new distances.

formula: D(Oi,Oj)^2 – (Xi – Xj)^2

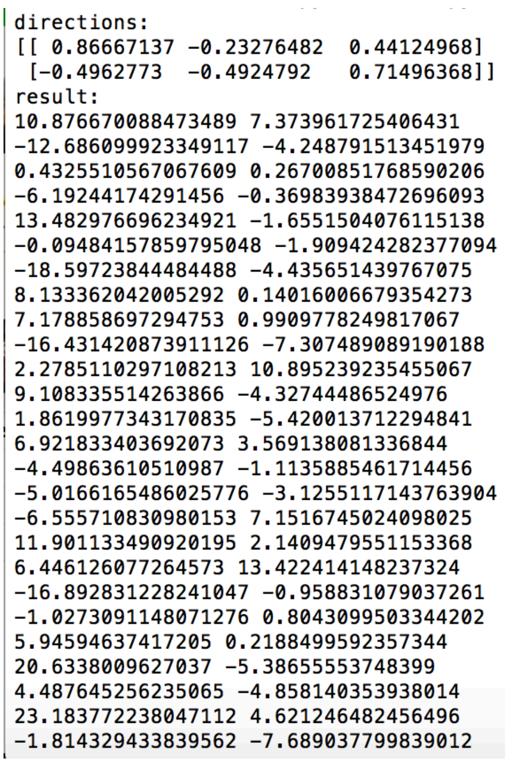
Step 6: Find second dimension using the formula in Step 4.

For the FastMap and PCA, we use numpy array and array to store the data.

3. Our Outcome

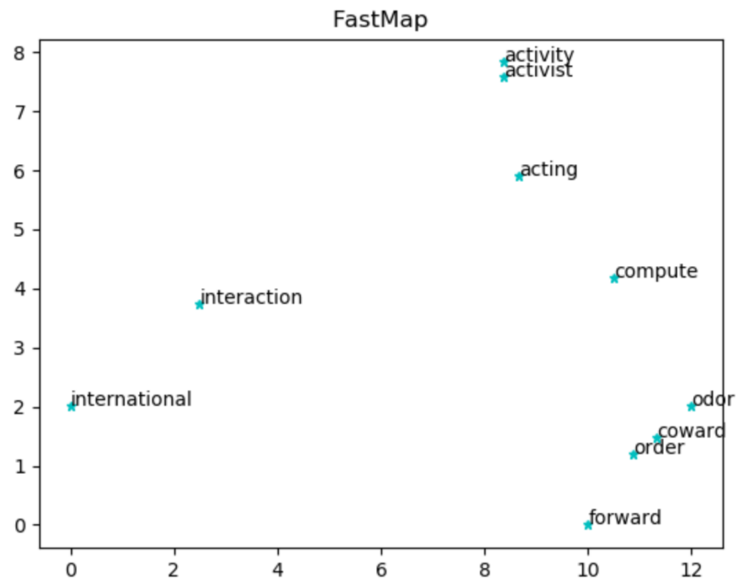
PCA:

directions of the first two components and result of dimensional reduction to pca-data.txt



```
directions:
[[ 0.86667137 -0.23276482  0.44124968]
 [-0.4962773  -0.4924792   0.71496368]]
result:
10.876670088473489 7.373961725406431
-12.686099923349117 -4.248791513451979
0.4325510567067609 0.26700851768590206
-6.19244174291456 -0.36983938472696093
13.482976696234921 -1.6551504076115138
-0.09484157859795048 -1.909424282377094
-18.59723844484488 -4.435651439767075
8.133362042005292 0.14016006679354273
7.178858697294753 0.9909778249817067
-16.431420873911126 -7.307489089190188
2.2785110297108213 10.895239235455067
9.108335514263866 -4.32744486524976
1.8619977343170835 -5.420013712294841
6.921833403692073 3.569138081336844
-4.49863610510987 -1.1135885461714456
-5.0166165486025776 -3.1255117143763904
-6.555710830980153 7.151674502409802 5
11.901133490920195 2.1409479551153368
6.446126077264573 13.422414148237324
-16.892831228241047 -0.958831079037261
-1.0273091148071276 0.8043099503344202
5.94594637417205 0.2188499592357344
20.6338009627037 -5.38655553748399
4.487645256235065 -4.858140353938014
23.183772238047112 4.621246482456496
-1.814329433839562 -7.689037799839012
```

FastMap:



Coordinates:

```
[[ 8.66666667   5.91497729]
 [ 8.375        7.57789976]
 [10.5          4.18888905]
 [11.33333333   1.48938277]
 [10.           0.        ]
 [ 2.5          3.74207421]
 [ 8.375        7.83322252]
 [12.           2.01066674]
 [10.875        1.20280957]
 [ 0.           2.01066674]]
```

4. Challenge

In order to implement PCA, we need to calculate lots of important variables. Although numpy can directly use the function, we still calculate those variables by ourselves. Hence, that is more difficult than using the functions.

The challenge we met in the FastMap is how to choose the furthest distance correctly because they may have several max distances and different choose will have different outcome. Also, we need to find the projections in k dimension, so how to store values to run the recursive function is also need to schedule.

5. Optimization

PCA:

When we implement PCA at Step2: compute covariance matrix, we store our data as a 3*n matrix instead of n*3, and thus our covariance matrix turned out to be a 3*3 matrix instead of n*n, which leads to a much faster computation of eigenvalue and eigenvector, and the final result is the same.

FastMap:

When we start to find the furthest distance between two objects, we will find the last max distance in all the data. Also, after we calculate the new distances, we will store the max distance and two objects. If we still need to calculate the next dimension, then we can save some time.

6. Software Familiarization

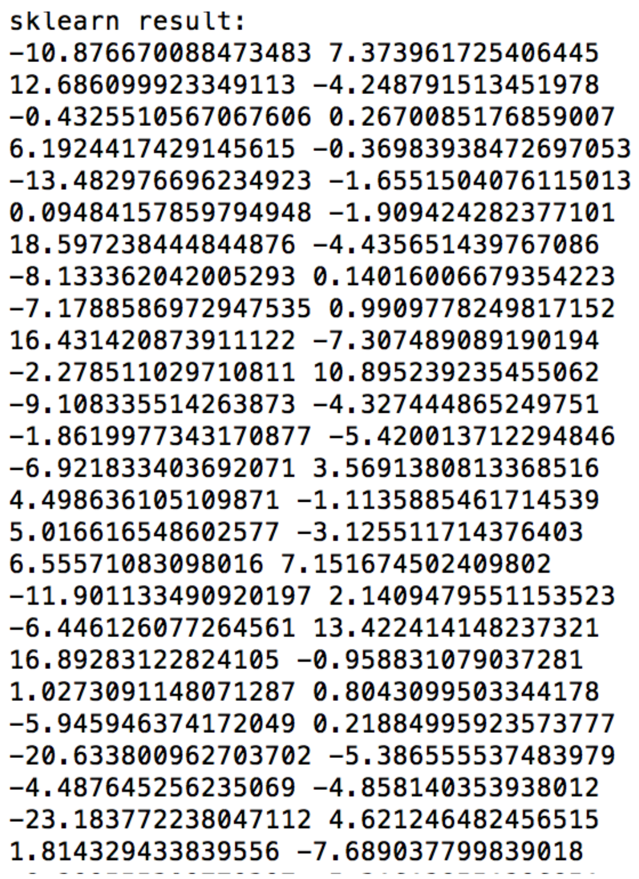
Sklearn.decomposition.PCA:

In addition to the parameter "n_components", dimension we want in our result, there are some optional parameters in this library that we can adjust.

whiten: default=false. If it is set to true, it will remove some redundant data and can sometimes improves the predictive accuracy.

svd_solver: It specifies which singular value decomposition(SVD) method we are going to use (auto, full, arpack, randomized.)

result from sklearn.decomposition.PCA:



```
sklearn result:
-10.876670088473483 7.373961725406445
12.686099923349113 -4.248791513451978
-0.4325510567067606 0.2670085176859007
6.1924417429145615 -0.36983938472697053
-13.482976696234923 -1.6551504076115013
0.09484157859794948 -1.909424282377101
18.597238444844876 -4.435651439767086
-8.133362042005293 0.14016006679354223
-7.1788586972947535 0.9909778249817152
16.431420873911122 -7.307489089190194
-2.27851102971081l 10.895239235455062
-9.108335514263873 -4.327444865249751
-1.8619977343170877 -5.420013712294846
-6.921833403692071 3.5691380813368516
4.498636105109871 -1.1135885461714539
5.016616548602577 -3.125511714376403
6.55571083098016 7.151674502409802
-11.901133490920197 2.1409479551153523
-6.446126077264561 13.422414148237321
16.89283122824105 -0.958831079037281
1.0273091148071287 0.8043099503344178
-5.945946374172049 0.21884995923573777
-20.633800962703702 -5.386555537483979
-4.487645256235069 -4.858140353938012
-23.183772238047112 4.621246482456515
1.814329433839556 -7.689037799839018
```

In addition to the little difference of the sign symbol (since sklearn libaray uses different ways to process the SVD result), this result is the same as ours.

7. Applications

PCA:

1. Image Compression:

   Converting the image in jpg format into numerical matrix, doing some matrix computations and evaluating quality of reconstruction.

2. Facial Recognition:

The principal components are projected onto eigenspace to find the eigenfaces. Recognizing unknown faces by the minimum Euclidean distance.

3. Analysis of bioinformatics data:

Constructing linear combinations of gene expressions as principal components and using different clustering algorithms after reducing their dimensionality.

FastMap:

1. Node localization in Wireless Sensor Networks
2. Human action recognition
3. Pathfinding on Grid Maps