Programming assignment 3: PCA and FastMap
Group Report

Chenqi Liu 2082-6026-02
Che-Pai Kung 5999-9612-95
Mengyu Zhang 3364-2309-80

Part 1: Implementation

PCA

Data Structure and Implementation

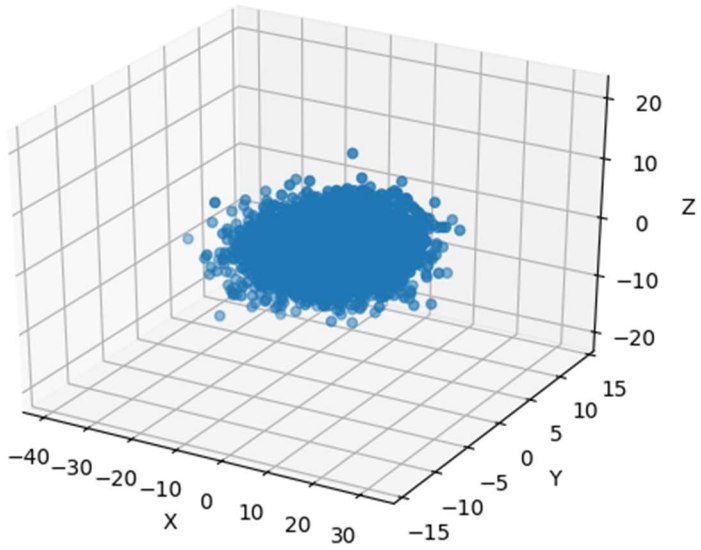
The given data is 3 dimensional and has totally 6000 data sets. The desired result is to reduce them to 2-D. The data structure we use to store the data is a 2D nested array which is easy for matrices calculation. We follow the steps of PCA, normalize every dimension of data, calculate covariance matrix, get the eigenvalues and eigenvectors, and choose the first k's as the principal components. While choosing the first 2 eigenvectors via the largest 2 eigenvalues, we use Hashmap to map eigenvalues and corresponding eigenvectors. The two largest eigenvalues are [609617.1824971355, 119375.39194986533], the corresponding eigenvectors are [[0.86667137, -0.23276482, 0.44124968],[-0.4962773 , -0.4924792 , 0.71496368]]. Finally, we dot product the normalized n-by-3 data matrix with 3-by-2 vectors and get the n-by-2 result**.**

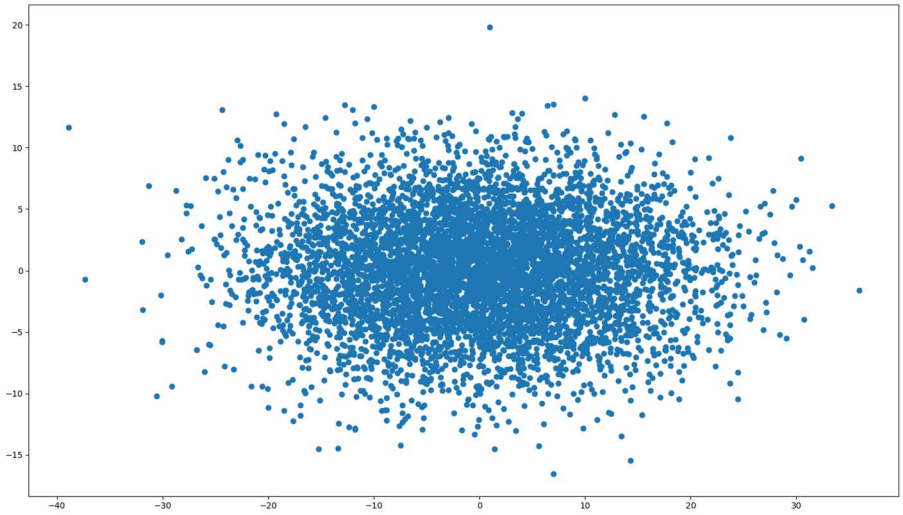
Code-level optimizations

By directly using functions of numpy, we reduced some transformations from data frame to numpy arrays and simplify the code. Besides, we did not reduce each point and combine all the points in a list at last. Instead, we use the whole matrix dot products its transpose and then do the processes, which saves the time of iteration and combination.

Output

The 6000 2d data points are saved in pca-output.txt file. Below are plots visually showing the 3d normalized data points and its 2d reduction:


20
10
0
−10
−20
Z
15
10
5
0
−5
−10
−15
Y
−40
−30
−20
−10
0
10
20
30
X


20
15
10
5
0
−5
−10
−15
−40
−30
−20
−10
0
10
20
30

FastMap

Data structure

Instead of being provided the distance function, we had only the result from the “fastmap-data.txt”. Therefore, we constructed a dictionary that stored the distances between each object to be the distance function in this program. In addition, the array (P), whose dimension is “number of objects” times “the dimensions of created space”, was created to store outputs after each iteration.

Code-level optimizations

We first used the pandas library to read the file. However, we later found that it is unnecessary to use the data frame structure as the column names are not important to us and the way to get the data is not straightforward. For example, if we want to get the value at row1, column2, it is not df[0][1] but df[1][0]. We then turned to loadtxt function from numpy, which gave us a matrix so that it is more convenient and instinct when retrieving data.

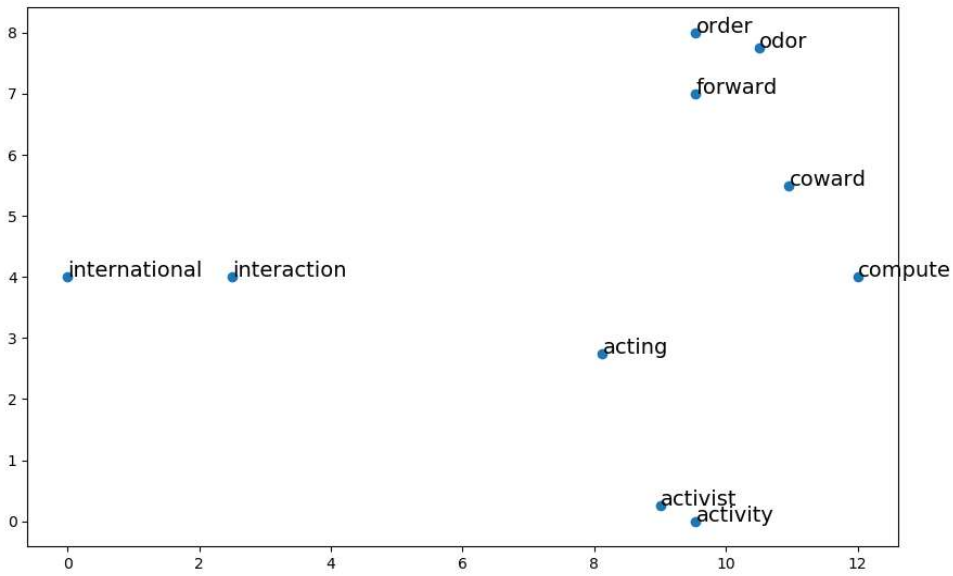
In addition, the functions were built independently at first. It was noticed that each function took lots of parameters and was not organized. Hence, to increase the readability of the program, the program was reorganized to object-oriented style, building all the functions needed in fastmap class.

Challenge

The biggest challenge was we were not familiar with the fastmap concepts, so when the first result came out, we were not sure if that makes sense. All we understood was that the similar words should be closer than other words. Although part of the result seemed not to follow such rule, we could not find error after reviewing the codes at that time. After being stuck for a while, we came up with a way to check if the farthest pair in each dimension shows the longest distance in the artificial space, then we found that there was an error that when calculating the distance.

2D Plane Output

We got the output as below, showing that the similar words are closer to each other and the wordlists provided could be therefore clustered into several groups if we need further analysis on those words.



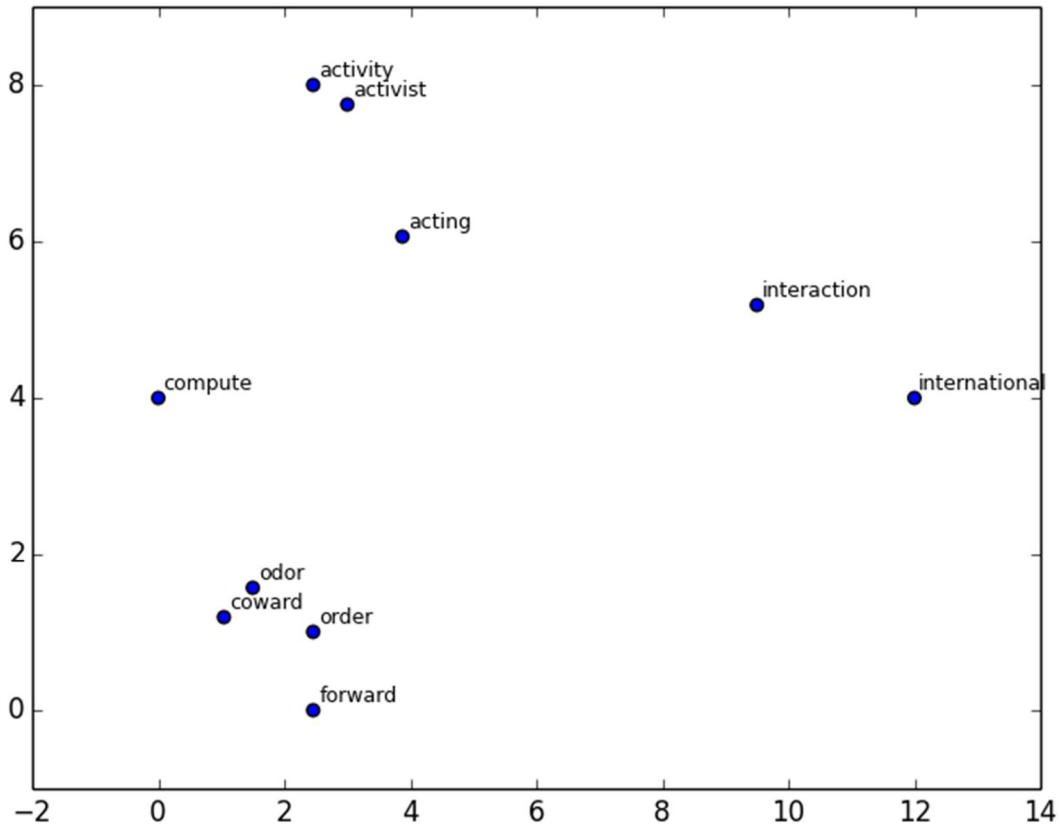
Part 2: Software Familiarization

PCA

The library we used for PCA is sklearn.decomposition.PCA from scikit Learn library. We used the library simply by calling the fit() and transform() functions inside the package. We compared the results and found out that the coordinates are almost the same but tiny difference since the 10th digit after the decimal point, which can be negligible. We also noticed that the sign of the first coordinate between the results are opposite. However, it shouldn't be a big deal. Thinking of the two implementations as looking at the data from different perspectives. The outcome shows that scikit library doesn't do improvements but using the basic PCA algorithm, and the way we are implementing the code is right.

FastMap

Since there is no fastmap package inside the scikit Learning library, and we barely find other well-written libraries that can run on our personal laptop. We decided to use someone else's algorithm in this assignment as a comparison for our implementation. We did some modification of the code so it can fit with our data. And here is the result:



By comparing this result with the result we generated, there is a slight difference between them. After looking into the data and algorithm, we found out that both (activity,forward) and (activity, order) are the pairs with maximum distance after one iteration. As we choose the maximum

distance starting by random point, it is possible that our implementation generates the same result as above. On the other hand, considering only (odor, coward, order, forward), it is intuitive that they should be separated into clusters (order,odor) and (forward, coward). However, the Fastmap algorithm tends to cluster into either (order,odor,forward) versus (coward) or (order,odor,coward) versus (forward). We deduced that the distance features between each object are not reflected fully after projecting on 2D plane and it is possible that we separate them well if projecting on higher dimensional space.

Part 3: Applications

PCA

PCA is a widely used dimensionality reduction algorithm which appears in graphic domains like facial recognition, image compression, and computer vision, as well as high-dimensional data fields like finance, bioinformatics, psychology, etc.

For image compression, PCA helps compress images with minimum loss of information as well as save a huge amount of space. For quantitative finance, institutional portfolio managers use this to allocate funds amongst assets and asset classes. Interest rate structure and quants use PCA to model the yield curve and analyze its shape and implement HJM Model.

FastMap

As FastMap transforms abstract distance concepts into Euclidean distance, it makes it possible to plot data points in 2D space or even higher dimensions while remaining their distance features. Therefore, Fastmap is found being applied on clustering and visualization a lot. Take WINE dataset for example, there are 13 variables of each data point (row) and the wine is derived from three different cultivars. With the help of Fastmap, it is feasible to separate one of the clusters on a 2D plane. Moreover, the clusters are distinguished well on the 3D plot.

References:

https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial
https://www.risklatte.xyz/Articles/QuantitativeFinance/QF_199.php
http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf
https://www.researchgate.net/publication/2609830_FastMap_A_Fast_Algorithm_for_Indexing_Data-Mining_and_Visualization_of_Traditional_and_Multimedia_Datasets

Part 4: Individual Contributions

- Model discussion: Chenqi Liu, Che-Pai Kung, Mengyu Zhang
- Model implementation: Che-Pai Kung, Mengyu Zhang
- Model optimization: Chenqi Liu, Che-Pai Kung, Mengyu Zhang
- Software Familiarization: Chenqi Liu
- Applications: Che-Pai Kung，Mengyu Zhang