INF 552 Programming Assignment 3

PCA and FastMap

4472-6221-33 Yo Shuan Liu

I. Introduction

I complete this homework in Python3. There are three python files in this folder, PCA.py, FastMap.py and ExecuteMeFor_PCA_FastMap.py. Logic and implementation details can be found in PCA.py and FastMap.py.

I Integrate both implementation, PCA and FastMap, in "ExecuteMeFor_PCA_FastMap.py". Please execute this file for reproducing result, also modify parameters or file path here if needed.

I use linalg method in numpy library to calculate eigenvalues and eigenvectors in PCA implementation, and the argmax method in numpy library to make finding farthest pair of objects more efficient while implementing FastMap.

I also visualize the scattering plot in 3D to get a picture of relationship between data and eigenvectors, by running my PCA.py, the interactive 3D plot will pop up in a new window, just remember to execute `%matplotlib auto` in the console to enable the interactive feature.

In this report, I will introduce both my PCA and FastMap implementations in the order of:
(1) Data Structure
(2) Challenges I Face
(3) Result

Then I will list out all the code level optimization I have done in this assignment. Finally, I will briefly introduce two interesting applications regarding PCA and FastMap.

II. Implementation of PCA

Data Structure

Input data are stored as an (6000*3) array, **covariance matrix** is stored as a (3*3) array. After obtaining covariance matrix's eigenvalues and eigenvectors, I pair them up as a list of tuples, that is, storing as a list of **eigen-pairs**. Therefore, I can sort the eigen-pair according to the eigenvalue and retrieve the two eigenvectors with higher eigenvalue.

Challenges I Face: visualizing the abstract concept in 3D

Since the concept of eigenvectors and eigenvalues is a little bit abstract to me, I decided to plot a 3D plot to get an idea of how eigenvectors look like along with the scatter plot. However, I couldn't figure out how to keep the eigenvector in upper layer while plotting 3D plot. Therefore, the arrow which represent the eigenvector was covered by my data points, especially the one with smaller eigenvalue (which makes it shorter and harder to be seen).
Finally, I decided to adjust the opacity of my data points, and also the color of the arrow, which improved the visualization.

Another point I would like to make is that I found out that the method in numpy for calculating covariance matrix use a scale of 1/(N-1), where N is the total number of input. While professor use 1/N as scaling factor. I was confused by it at first, but finally realized that this scaling factor only influence the eigenvalues, and the eigenvectors were identical in both cases. Since the final output data is transformed only according to the eigenvectors, the final output data remained the same, no matter 1/(N-1) or 1/N is used when calculating covariance matrix.

Result

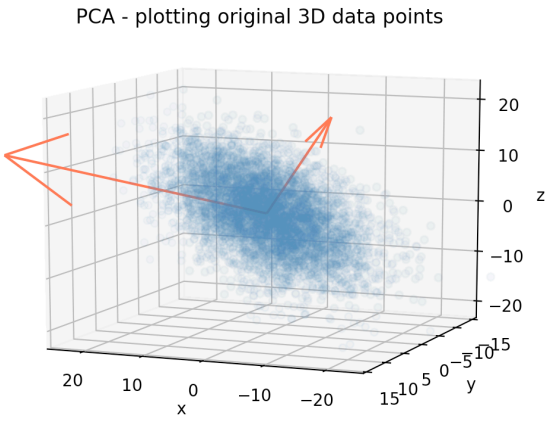
Direction of first two principal components:

(1) **[0.86667137 -0.4962773 -0.0508879]**

(2) **[-0.23276482 -0.4924792 0.83862076]**

Visualization of data points (in blue) and also eigenvectors (in orange):

- Eigenvector indicates the weight of each original variables in the principal component.
- Length of eigenvector in this plot indicates its eigenvalue, which can be interpreted as the importance of its corresponding eigenvector.



Proportion of variance explained: **96.21%**

Since the eigenvalue in PCA tells us how much variance can be explained by its associated eigenvector, it can be concluded that the two eigenvector we chose explained 96.21% of the variance in the original data. This indicator, proportion of variance explained, is calculated by sum of chosen eigenvalue divided by the sum of all eigenvalues.

We can also observe the center of transformed data after our PCA transformation is (0, 0). This results from the data centralization we apply before applying PCA.

Software Familiarization: sklearn.decomposition.PCA

Sklearn's PCA has some special attributes, for example, explained_variance_ and noise_variance_, which are related to eigenvalues.

I didn't calculate the information of explained variance (96.21% indicated above) at first. Not until I learned how to use PCA library, had I realize that PCA in sklearn has an attribute "explained_variance_", which is equal to n_components largest eigenvalues of the covariance matrix of input data. Therefore, I decided to add it as one of the quality indicator in my implementation.

My 2D output data points and the output of sklearn PCA library are identical, so are the eigenvalues and eigenvectors.

III. Implementation of FastMap

Data Structure

There are 10 different objects in the input data. I store the distance between any two objects in a 10* 10 matrix (array), let's call it **distance matrix**. For example, distance between object 1 and object 2 is stored in the (0, 1) and (1, 0) position of this distance matrix, since the distance from object 1 to object 2 is

identical with the distance from object 2 to object 1. And all elements on the diagonal are zero, which means distance from object to itself equals to zero. **Output data points** have two coordinates which allow object to be represented in a 2D plot. Output data points are stored in an 10*2 array.

Challenges I Face: A weird error could be solved by restarting IDE

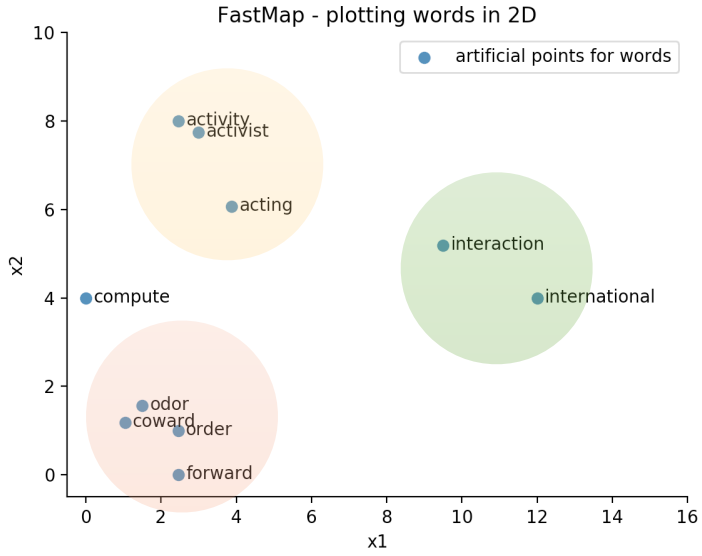
Sometimes I would hit a weird error (shown below) related to plotting, but the problem resolved after I restart my python IDE (Spyder 3.2.3). Not sure if this is an issue with Spyder or matplotlib.

```
  File "/anaconda3/lib/python3.6/site-packages/matplotlib/collections.py", line 877, in set_sizes
    scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
AttributeError: 'NoneType' object has no attribute 'sqrt'
```

Result

Words are now represented as 2D points after applying FastMap, I call the coordinates as x1 and x2. We can observe some groups in this 2D plot.

- odor, order, coward, forward
- activity, activist, acting
- interaction, international
- compute



Software Familiarization:

I couldn't find any library in python for FastMap, however, I do find a **C program** written by Christos Faloutsos, who wrote the paper "FastMap: a Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets SIGMOD 1995, pp. 163-174".

The logic of implementing FastMap is mainly located in oa.c file. I notice that instead of storing the distance between two objects in the distance matrix, **the author stores the square of the distance**. By storing distance in this way, I can cut down some calculations.

Remaining implementation are align with what professor mentioned in class, and therefore is also similar to mine.

Source: http://www.cs.cmu.edu/~christos/software.html (FastMap tar file In C.)

IV. Code Level Optimization

1. I wrote an additional “ExecuteMeFor_PCA_FastMap.py” for TA/professor to easily execute and check my code. The two files that implement algorithm (“PCA.py” and “FastMap.py”) are imported as module in this file. In this way, TA/professor only needs to modify the folder location once in “ExecuteMeFor_PCA_FastMap.py”. Parameters needed in both algorithm are also listed out clearly and could be modified here.

2. Before:
   For FastMap algorithm, I store the distance between objects in a distance matrix. Therefore, I need to square the distance when producing coordinates (calculate projection onto the landmarks segment), and also square the old distance, subtract the information explained and then store the square root of new distance when I update the distance matrix before starting a new iteration.
   After:
   Instead of storing the distance between two objects in the distance matrix, I store the square of the distance. I was inspired by the C program written by Christos Faloutsos (mentioned in the software familiarization part).

V. Application

One interesting application for PCA is in chemical biology and drug design. Principal component analysis (PCA), in a typical quantitative structure-activity relationship (QSAR) study, analyzes an original data matrix in which molecules are described by several intercorrelated quantitative dependent variables (molecular descriptors). PCA is a powerful technique for exploring complex datasets in QSAR studies for identification of outliers.

Audio retrieval technique can also base on FastMap algorithm. Nowadays, more and more web and application allow users to voice search. This application combine the segmentation-based retrieval technique and FastMap to increase the overall accuracy. It is proved that FastMap can not only improve the searching speed but also provide a more accurate dimensionality reduction result when analyzing audio data.

Reference:
https://uosc.primo.exlibrisgroup.com/discovery/fulldisplay?docid=wj10.1111%2Fcbdd.13064&context=PC&vid=01USC_INST:01USC&lang=en&search_scope=MyInst_and_CI&adaptor=Primo%20Central&tab=Everything&mode=Basic
https://uosc.primo.exlibrisgroup.com/discovery/fulldisplay?docid=wanfang_jourhbgxyxb2015z1012&context=PC&vid=01USC_INST:01USC&lang=en&search_scope=MyInst_and_CI&adaptor=Primo%20Central&tab=Everything&query=any,contains,FastMap&offset=0