Programming Assignment 3: PCA and Fast Map

Kavya Sethuram(ksetura@usc.edu); Rasika Guru (rguru@usc.edu); Roshani Mangalore (rmangalo@usc.edu)

1. Implementation of PCA and Fast Map Algorithm in Python v2.7

A. DataStructures used in the Implementation are:

- ***Lists:*** Python has a great built-in list type named "list". List literals are written within square brackets [ ]
- ***Dataframe*****:** Data Frame is a 2-dimensional labeled data structure and is the most commonly used panda object. We have generated Dataframe from the datafile using pandas.

B. Explanation:

PCA:

1. Start with data set X, compute the mean(μ) of each of the x, y and dimension vectors.

2. Update the new data matrix(D) by calculating the difference between the original data set and the mean computed in step1.

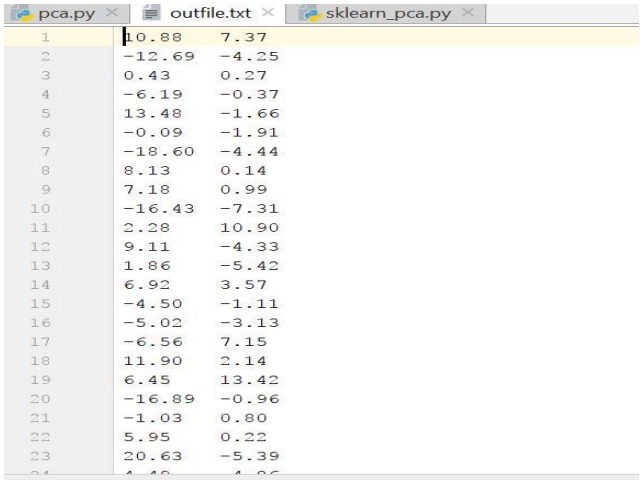
3. Calculate the covariance matrix which is a dot product of the Data matrix D with its transpose divided by the total number of data points(6000 in this case).

4. Compute the eigen values and eigen vectors from the covariance matrix using np.linalg.eig method.

5. Select the two highest eigen values and pick the respective eigen vectors. These eigen vectors are stored in a Eigen Matrix.

6. Matrix reduced to the two-dimensional plane is calculated by the dot product of the eigen matrix and the data matrix D.

Output of PCA displays the reduced matrix on two-dimensional plane. The x, y co-ordinates of all the data points are written into a file (outfile.txt) and is displayed on the output screen as well.



FAST MAP:

1. Given a NxN matrix, choose an arbitrary random number between 0 to N

2. Select the initial vertex to be the element at index from step 1, say oa.

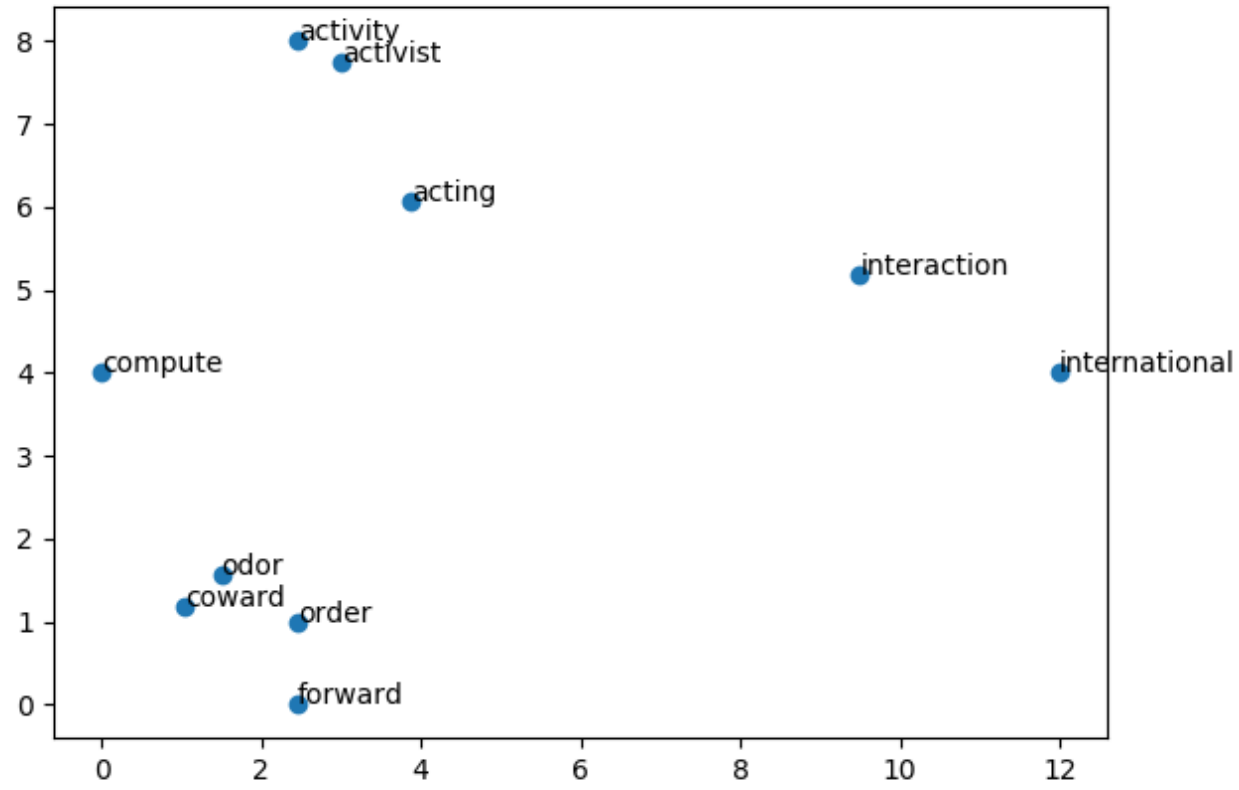
3. Choose the vertex which is at the maximum distance from oa say ob.Now oa and ob are the two pivot points.

4. Compute the distance between oa and ob, ie Dab.
5. Project all the objects in the data array on the line O(oa,ob) and calculate the new Xi using the cosine formula.

6. Calculate the projections on a hyper plane perpendicular to oa,ob.

7. Repeat the steps 2 to 6 k times, where k is the dimensionality to which the objects are projected.

Output of FastMap is as shown below (Mapping of word list in a 2-dimensional plane)



C. Code Level Optimizations:

- Using NumPy comprehensions over explicit for loops in certain places, making code more compact and execution faster.
- Matrix operations are easy with NumPy arrays
- Modularizing parts of code into methods, making code more readable.
- Computation of the farthest points in a space is being done with a complexity of O(n) against the usual O(n^2)

D. Challenges:

- Choosing a data structure to facilitate the computation of the first k-eigen vectors was challenging.
- Visualizing multi-dimensional space for fast map was a little hard.

2. Software Familiarization:

We have used the following packages to implement PCA and FastMap

NumPy

- NumPy is a Numeric Python module. It provides fast mathematical functions.
- Numpy provides robust data structures for efficient computation of multi-dimensional arrays & matrices.
- We used numpy to read data files into numpy arrays and data manipulation.

Pandas

- Provides DataFrame Object for data manipulation
- Provides reading & writing data b/w different files.
- DataFrames can hold different types data of multidimensional arrays.

pylab*:* pylab is the library used to plot graphs. The word lists are represented in 2D space.

Existing Libraries:

1. Scikit-Learn (for PCA)

    - It's a machine learning library. It includes various machine learning algorithms.
    - It uses the following inbuilt library to implement PCA algorithm from sklearn.decomposition import PCA
    - In scikit-learn, PCA is implemented as a transformer object that learns n components in its fit method, and can be used on new data to project it on these components

Existing Library code:

```
import pandas as pd
from sklearn.decomposition import PCA

df = pd.read_csv('pca-data.txt', sep =
'\t',header=None)
data_array = df.as_matrix()

pca= PCA(n_components=2)
result=pca.fit_transform(data_array)
print result
```

2. https://github.com/mahmoudimus/pyfastmap (For Fast Map).

    - We referred to the python implementation of FastMap from the above link ,for indexing, data-mining and visualization of traditional and multimedia datasets.
    - Computation of Distance Matrix is effectively handled in this implementation compared to out code
    - We verified our output against this code and it matches with our implementation

3. Applications of PCA

1. Principal component analysis (PCA) is a well-established technique for monitoring and disturbance detection of multivariate process, as it enables variability assessment through dimensionality reduction. PCA was applied to a hydroprocessing pilot plant to monitor the overall process variability. Contribution plots around points of increased variability were used to analyze process variability and its association with process variables. The methodology monitored successfully the set of 38 variables and diagnosed significant disturbances and their causes

   [Reference]: http://pubs.acs.org/doi/pdf/10.1021/ie0714605

2. Principal Component Analysis is widely used in applied multivariate data analysis, and this article shows how to motivate student interest in this topic using cricket sports data. Here, principal component analysis is successfully used to rank the cricket batsmen and bowlers who played in the 2012 Indian Premier League (IPL) competition. In particular, the first principal component is seen to explain a substantial portion of the variation in a linear combination of some commonly used measures of cricket prowess. This application provides an excellent, elementary introduction to the topic of principal component analysis.

   [Reference]: http://ww2.amstat.org/publications/jse/v21n3/scariano.pdf.

4. Applications of FastMap:

1. Many large and complex computational applications can be modeled as irregular graphs and are typically characterized by a large number of vertices and edges. This paper proposes a new and fast mapping heuristic, called FastMap, to map this class of applications onto heterogeneous meta computing platforms such as computational grids. We exploit a hierarchical resource management infrastructure on the grid to distribute the overhead of mapping among a tree of schedulers and develop a scheme that proves to be almost linear in its scalability. Furthermore, we optimize on the result of the mapping with the help of a genetical algorithm at each scheduler node. Our experiments include a50,000-node application graph from NASA and several other synthetically-generated graphs with as many as 100,000 vertices. Comparisons with another heterogeneous partitioner, MiniMax, show an improvement factor of over 100 on the mapping time, yet with superior quality mapping.

   [Reference]: http://ieeexplore.ieee.org/document/1309098/

5. Individual Contribution

PCA	Kavya Sethuram
Fast Map	Rasika Guru and Roshani Mangalore