INF552: Programming Assignment 3

Part 1: Implementation

The code is in ***Jiashi_Chen_Homework_3_Code.ipynb*** file.

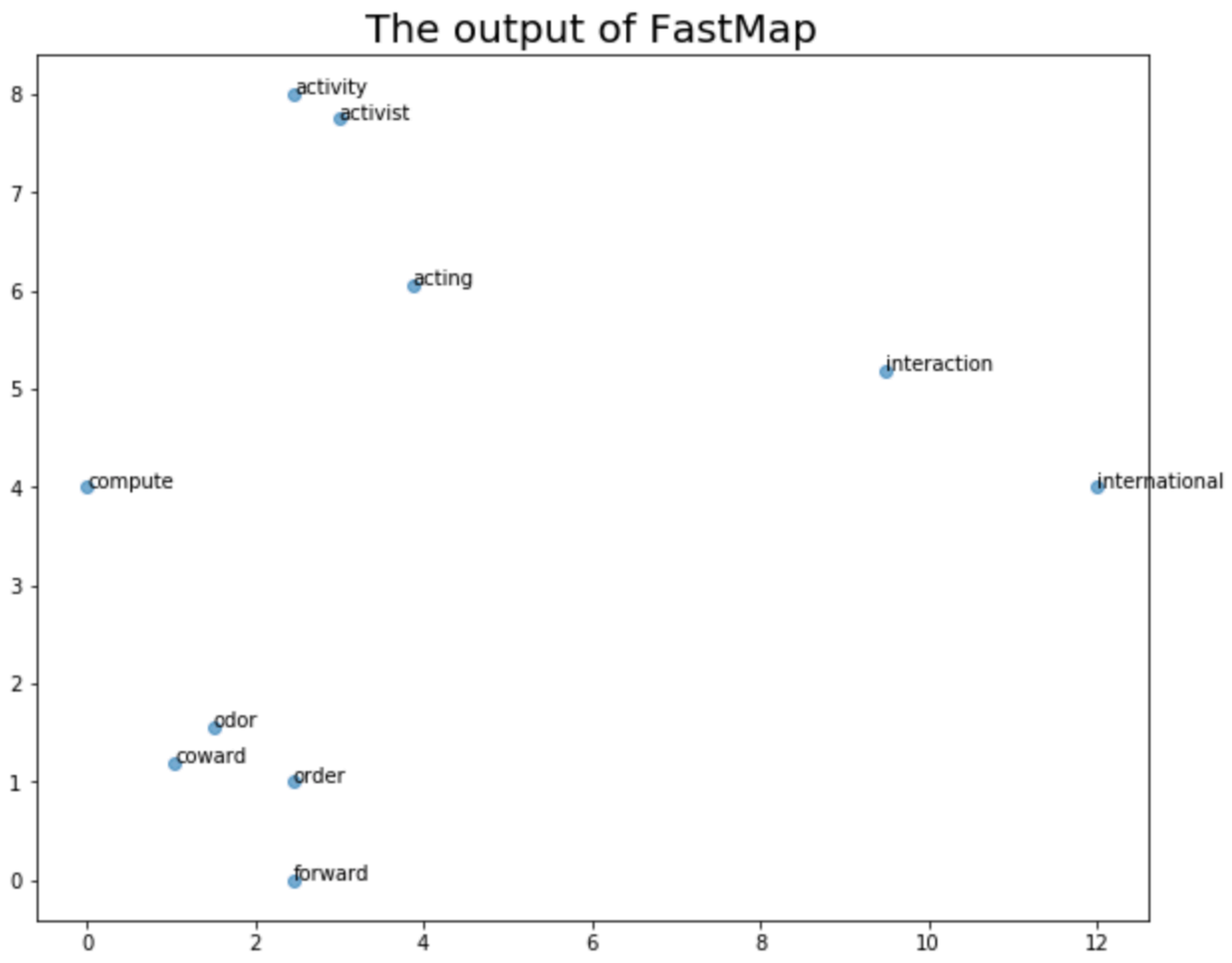
PCA

The directions of the first two principal components:

```
[[ 0.86667137,-0.23276482, 0.441249689],  # the first principal components
 [-0.4962773, -0.492479,  0.71496368]]    # the second principal components
```

FastMap

The words on a 2D plane after using fastmap algorithm:



The **data structure** I use is array and matrix from numpy module and dataframe from pandas module. Dataframe could help me store the orginal data. Array can be used to store the eigenvalues in PCA and new coordinates in fastmap. The matrix could store the eigenvectors and data which is converted from dataframe.

The code-level **optimizations** I perform is that I used the two mature modules numpy and pandas to complete matrix operations and obtain eigenvectors and eigenvalues, which improves the efficiency a lot. In fastmap, I defined each step into each function, which is beneficial to my debugging and improvement in the future. I package the steps that need to be done repeatedly, such as normalization, into a function to make the code cleaner

The **challenges** I face is that I spend a lot of time in figuring out how to get the new distances without computing and storing all of the old distances except the initial distance in the code. Also I pay a lot of efforts to find out the difference between the operation of array in numpy and the operation of matrix in pandas when it comes to the same operator.

Part 2: Software Familiarization

PCA

Sklearn package offers a good implementation of the PCA algorithm

How to use

```
from sklearn.decomposition import PCA ## Obtain necessary package

pca=PCA(n_components=2)   #Choose the number of components to keep
newData=pca.fit_transform(data)  ## Obtain new data
```

I used this package to fit the data in **Jiashi_Chen_Homework_3_Code.ipynb** file too and obtain the same output as my implementation.

Comparsion and Improvement

In sklearn module, it uses Singular Value Decomposition of the data to project it to a lower dimensional space instead of computing the eigenvalues and eigenvectors of the covariance. It uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract. This method is effective when the sample size is large. I should also add this method into my PCA function.

FastMap

I couldn't find any popular module to implement fastmap so I try to understand the implementation which obtains the most stars in the Github.(https://github.com/mahmoudimus/pyfastmap). I find out that his basic idea is similar with mine. What I need to learn is that he put all of the necessary functions in a class named StringMap. I could also try to do that. It will be much more efficient to maintain the code and reuse the algorithm in this way.

Part 3: Applications

PCA in detecting human face[1]

This paper presents a new idea for detecting an unknown human face in input imagery and recognizing his/her facial expression. The objective of this research is to develop highly intelligent machines or robots that are mind implemented. A Facial Expression Recognition system needs to solve the following problems: detection and location of faces in a cluttered scene, facial feature extraction, and facial expression classification. The universally accepted five principal emotions to be realized are: Angry, Happy, Sad, Disgust and Surprise along with neutral. Principal Component Analysis (PCA) is implemented with Singular value decomposition (SVD) for Feature Extraction to determine principal emotions. The experiments show that the proposed facial expression recognition framework yields relatively little degradation in recognition rate due to facial images wearing glasses or loss of feature points during tracking.

fastmap in wireless sensor networks[2]

Node localization in wireless sensor networks (WSNs) has attracted much attention due to the increase of usage and applications of WSNs. Many algorithms and techniques for locating sensor nodes have been proposed in the literature. A recent algorithm, which is based on a given set of pairwise distance estimates among nodes and the target, generates a map of node locations. This algorithm, known as FastMap, uses projections onto orthogonal hyperplanes to find the coordinates successively. In this paper, we analytically study the performance of the FastMap algorithm for 2D positioning. Moreover a modified version of FastMap with better performance is proposed and analyzed. The analysis shows that the optimum anchor placement should be at the edge of the network.

Reference

[1] Kaur M, Vashisht R, Neeru N. Recognition of Facial Expressions with Principal Component Analysis and Singular Value Decomposition[J]. 2011, 9(12):24-28.

[2]Waleed A. Saif, Mounir Ghogho, Desmond C. McLernon. A modified fastmap algorithm for node localization in Wireless Sensor Networks[C]// Signal Processing Advances in Wireless Communications, 2008. SPAWC 2008. IEEE 9th Workshop on. IEEE, 2008.