Assignment 3

Gaurav Baney gbaney@usc.edu
02/22/2020

1. Implementation

1.1 PCA

My implementation of PCA is in the file 'pca.py'
The input file has been hardcoded to run upon executing the file.

The results of my implementation were as follows:

```
[ 0.86667137, -0.4962773 ]
[-0.23276482, -0.4924792 ]
[ 0.44124968,  0.71496368]
```

In my implementation I use numpy arrays to improve the speed of the algorithm and also for certain math features that can be directly applied to the numpy arrays
Using NumPy I can directly subtract out the means from the parsed data, calculate the covariance and also the eigenvalues and eigenvectors

1.1.1 Data Structures

I used one numpy array to store all the data
I used another numpy matrix to store the covariance matrix
I used a list to store the calculated eigenvectors

1.1.2 Code Level Optimzations

I subtracted the means from the data directly upon parsing them to make it more constant through the flow of the program

1.1.3 Experience

Overall PCA was the simpler part of the assignment so I didn't face any challenges during my implementation. It took only a couple minutes to read about the math involved in the algorithms.

1.2 FastMap

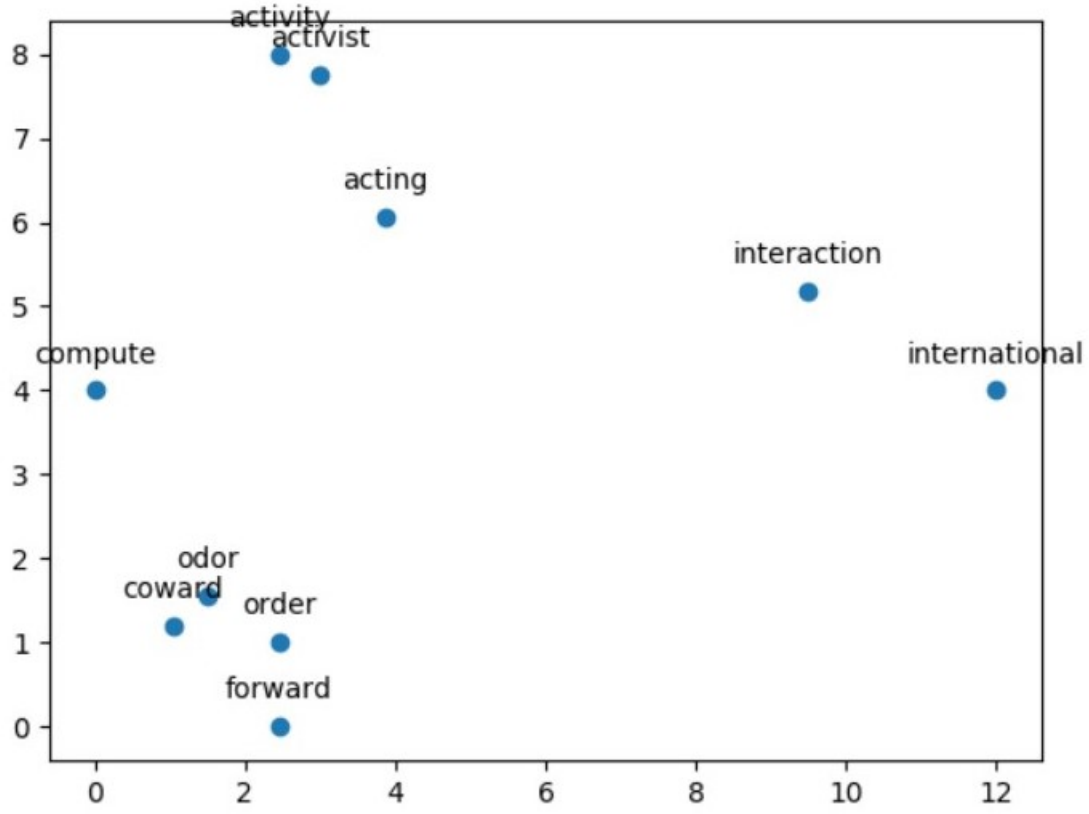
My Implementation of fastmap algorithm is in the file 'fastmap.py'
The input file has been hardcoded to run upon executing the file.

The results of my implementation were as follows:

```
'acting': [3.875, 6.0625],
'activity': [2.4583333333333335, 8.0],
'compute': [0.0, 4.0],
'coward': [1.0416666666666667, 1.1875],
'forward': [2.4583333333333335, 0.0],
'interaction': [9.5, 5.1875],
'international': [12.0, 4.0],
'odor': [1.5, 1.5624999999999996],
'order': [2.4583333333333335, 1.0]

 1: [3.875, 6.0625],
 2: [3.0, 7.749999999999999],
 3: [0.0, 4.0],
 4: [1.0416666666666667, 1.1875],
 5: [2.4583333333333335, 0.0],
 6: [9.5, 5.1875],
 7: [2.4583333333333335, 8.0],
 8: [1.5, 1.5624999999999996],
 9: [2.4583333333333335, 1.0],
 10: [12.0, 4.0]

1: acting
2: activist
3: compute
4: coward
5: forward
6: interaction
7: activity
8: odor
9: order
10: international
```



1.2.1 Data Structures

I used a global list to keep track of all the object IDs from the provided file
I used a dictionary of dictionaries to keep track of the original symmetric distance per object in ascending order with no duplicates. i.e. if I have an entry 1->3 = 7 I won't have an entry 3->1 = 7, and this was to avoid extra looping in the distance calculation methods
I use a map to finally track the mapped objects to their generated coordinates and plot on the scatterplot

1.2.2 Code Level Optimizations

I keep constant track of the current pair of furthest points and the distance between them to avoid recalculation in the distance generation and coordinate assignment algorithms
I have a initial coordinate assignment and furthest pair of points method to make it easier to increase/decrease the dimensions of the problem, but I also realized I hardcoded a bit of it in
I also used the max number of iterations given in class to ensure the furthest point is found.

1.2.3 Experience

Overall this part of the assignment wasn't too bad. It only took a little while to figure out how to store the data that would best make sense to me.
I don't think my implementation is optimal, in that i have a lot of repeated code that I couldn't figure out how to standardize.
Since the distance between 3->10 and and 8->10 is the same, my code has two sets of outputs that it decides between based on which inital furthest pair it picks.

1.3 Software Familiarization

1.3.1 sklearn.decomposition.PCA

I used the sklearn.decomposition.PCA package for PCA.
The results were as follows:

```
1         [ 0.86667137, -0.4962773 ]
2         [-0.23276482, -0.4924792 ]
3         [ 0.44124968,  0.71496368]
```

PCA was pretty straightforward, our answers matched.

2. Applications

2.1 PCA

PCA is primarily used for dimensionality reduction so it is very useful in making large datasets more smaller and easier to build a fit upon. It can be used to take high dimensional data like images or facial recognition points and extract their principal components so that other classification algorithms can identify patterns in them.

2.2 FastMap

Since FastMap makes use of what is essentially the edit distance between two objects, it can be used to map anything set of objects in euclidean or higher dimensional space