HW 03 – PCA AND FASTMAP

Ishan Joshi
USC ID- 3451334123
06/23/2020

Index

1.	Principal Component Analysis: • Introduction • Algorithm • Program Functions • Data Types • Output
2.	Fastmap: • Introduction • Algorithm • Program Functions • Data Types • Output
3.	Code Challenges

Principal Component Analysis

Dimensionality Reduction helps reduce the amount of redundancy in a set of features. Machine learning applications often encounter datasets containing a large number of features. However, this makes data visualization an arduous task. The features are often corelated which causes an increase in redundancy as well. To counter these problems, algorithms like Principal Component Analysis (PCA) reduce the number of features, or the dimensionality of the dataset such that only a select number of principal features are considered.

Algorithm:

1. Normalize the input data by subtracting the mean from each datapoint. The input data for the assignment is 3-dimensional and the user specified reduction is to 2-dimensional data.
2. Calculate the covariance matrix of the normalized data.
3. Calculate the eigen values and corresponding eigen vectors of the covariance matrix.
4. Arrange the eigen values in decreasing order of magnitude. The corresponding eigen vectors will also be arranged in the same order.
5. Select the first two eigen vectors, since the assignment specifies that the data must be reduced to two dimensions.
6. Calculate the resultant data using the following computations:

$$Z(i) = U'x(i)$$

Where Z(i) – new datapoint in 2D

U' – transpose of matrix containing first two eigen vectors

X(i) – Original datapoint

Program Functions:

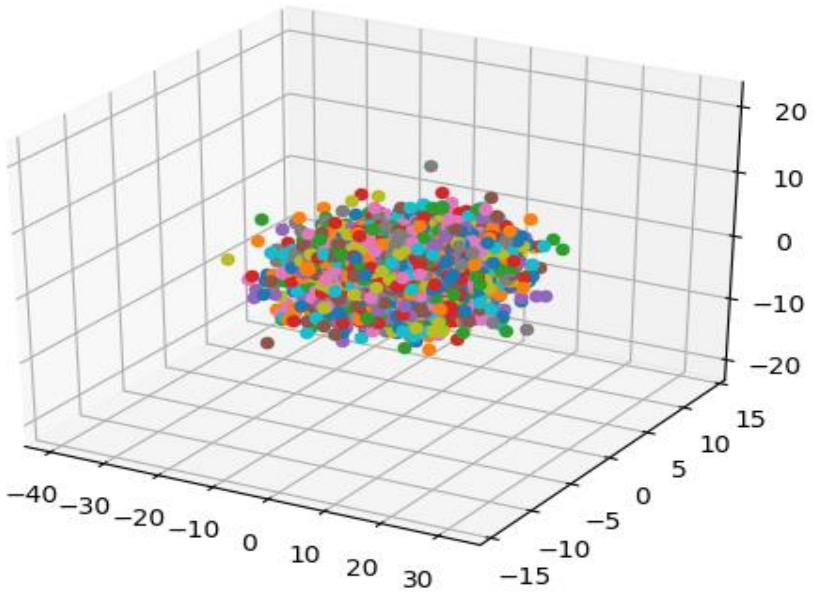
1. **average_and_covariance():** This function takes the original dataset as input as computes the mean and the covariance matrix of the normalized data. It subtracts the mean from each datapoint and uses the resultant matrix to compute the covariance matrix. It returns the covariance matrix
2. **pca():** This function takes the value of k or the specified reduced dimension and computes the reduced data. It returns the resultant eigen vectors and the array containing the reduced data points.

Data Types:

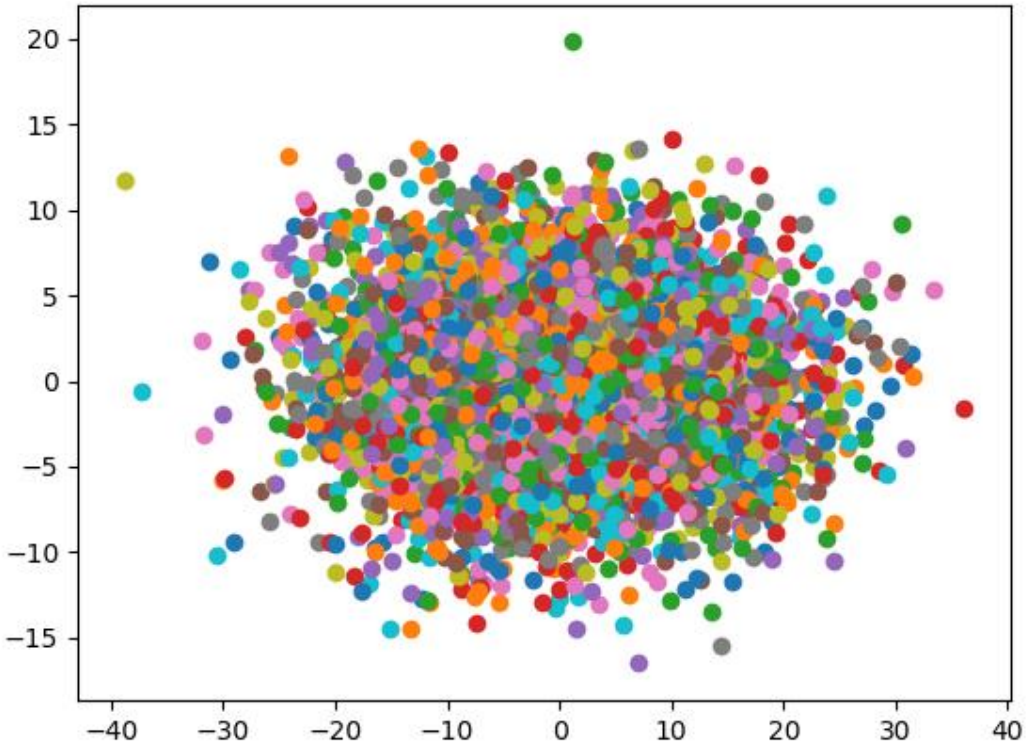
1. **Arrays -** Arrays are used to store data of one data type in a contiguous location. Arrays are used at multiple points in the code to store data at intermediate steps within loops.
2. **Dataframe -** A dataframe is a data structure within the pandas dictionary in python which makes handling and accessing large quantities of data easy for machine learning algorithms. The dataframe in the code is used to store the dataset. It is separated into three columns to store the input dataset.

Output:

The original input data can be plotted as follows:



The reduced data can be seen as follows:



The eigen vectors are as follows:

[[0.86667137 -0.4962773]

[-0.23276482 -0.4924792]

[0.44124968 0.71496368]]

Column 1 corresponds to the first eigen vector and Column 2 the second. We only consider the first two eigen vectors as the user has specified a 2D data as output.

Fastmap

Fastmap is an algorithm that is widely used in data mining applications. It is used to convert a set of N-dimensional objects into k-dimensional points, which can then be used in clustering operations and other uses. Fastmap requires pairwise distances between the objects as an additional input apart from the abstract objects themselves. The kind of distance can be specified by the user, as well as the objective dimension (k). This algorithm is traditionally used on objects such as images, DNA sequences etc where you have data objects that cannot be represented as coordinates or points. A speciality of the algorithm is that it is executed in linear time complexity.

Algorithm:

1. Find the largest pairwise distance, i.e. find the two objects that are farthest apart. This can be done as follows:
    - Choose one point as the initial pivot and find the farthest object from it.
    - Change the pivot to this resultant object and find the farthest object from it. The iteration where this process converges will be the pair of objects that are farthest apart.
    - This process is important as we want to execute the algorithm in linear time itself. Counting all the distances for each pivot and storing them would result in O(n^2) complexity.
2. Let these points be represented as Oa and Ob. For every point Oi, the calculations of coordinates can be graphically depicted using a triangle whose sides are the distances dab, dai, dbi between point Oa and Ob, Oa and Oi, and Ob and Oi respectively. He first coordinate Xi can be computed as:

$$Xi = d_{ab}^2 + d_{ai}^2 - d_{bi}^2/2 * d_{ab}$$

Where the distances are based on the distance function that will be specified in the input for the first iteration.

3. For subsequent iterations, we use the same functions recursively while changing the distance function for each iteration. The new distance function Dnew is defined as:

$$Dnew = D_{old}^2 - (Xj - Xi)^2$$

Where Xi and Xj are the coordinates for points Oi and Oj calculated in the previous iterations.

4. Thus the above functions run recursively for 2 iterations in the case of this assignment.

Program Functions:

1. **Find_Farthest_Pair() –** This function takes the recursion control variable as a parameter and calculates the farthest distance pair for each iteration. It returns the farthest pair of objects for each iteration using the distance function.
2. **Get_distance() –** This function takes the recursion control variable and two objects as input and calculates the distance between them. Based on the value of the recursion control variable, the function either simply returns the input pairwise distance or calculates a new distance function based on the calculation in step 3 of the algorithm.
3. **Fastmap() –** This function takes the target dimensions as the input and calculates the coordinates for each object. It increments the recursion control variable when the iterations are greater than one (for this homework). If Oi = Oa, X(i) = 0. If Oi = Ob, X(i) = dab. In every other case the x coordinate is calculated using the calculations in step 2 of the algorithm. The functions are recursively called by passing (k-1) as the parameter in the next run.

Data Types:

1. **Recursion Control Variable:** The variable "iterations" works as a recursion control variable for the program. Since the values for farthest pairwise distance and fastmap itself depends on the iteration of the recursion, the variable "iterations" essentially determines the result of the code.
2. **Arrays, Dataframe:** These data types are used to store data and input files at multiple points in the program.

Output:

Iteration 1

A:3

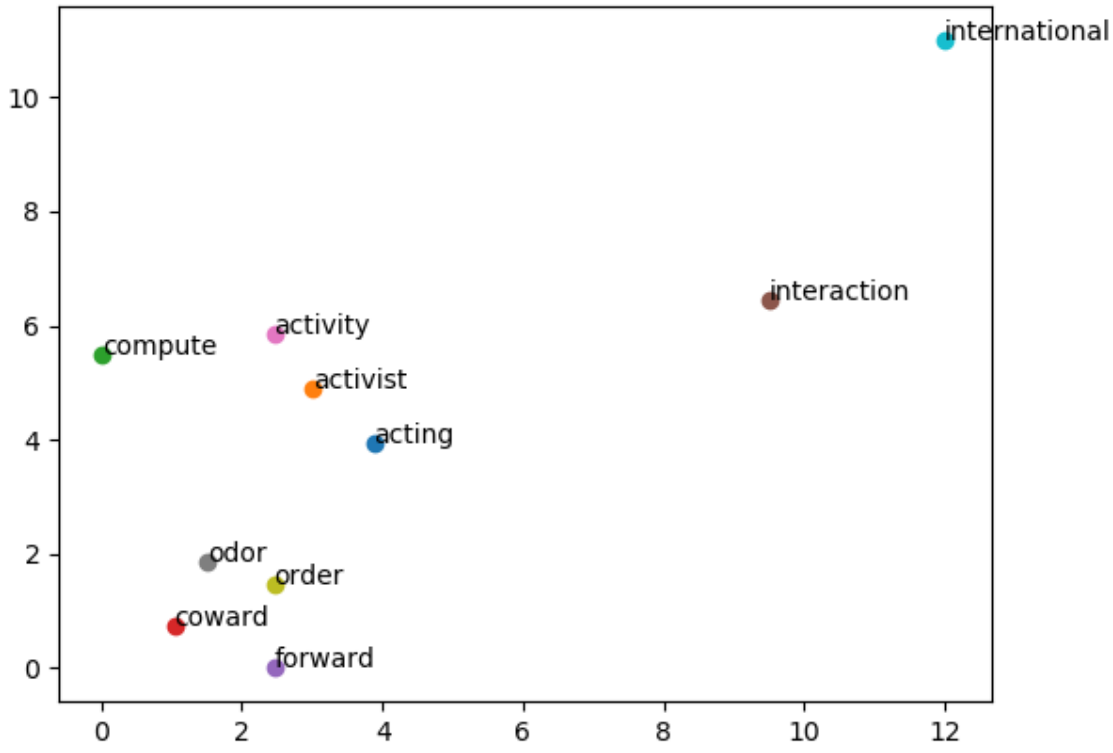
B:10

Iteration 2

A:5

B:10

Recursion limit reached



Code Challenges:

1. The PCA program takes long to code the data in 3D and 2D. This can probably be due to having a relatively large dataset, but the time taken to output the code is slightly longer than expected.
2. The fastmap algorithm presents a new output based on the distances chosen. There is a possibility that two pairs of objects might be farthest apart, in which case implementing either of the two distances could give different outputs, which is expected from the algorithm. At times, if the randomisation of Oa is taken to be 10, it presents a problem since the value 10 doesn't exist in the first column of the dataset.
3. To circumvent these problems, we can try to pick the minimum index in case of a farthest distance tie and can pick Oa randomly from 1-9 instead of 1-10. For this homework picking the minimum or maximum index can be done to ensure a uniform output is obtained.