Project 1 on PCA and Apriori Algorithm

Dong Xuanyu, Yifu Yin, and Tiehang Duan

Department of Computer Science and Engineering
The State University of New York at Buffalo
Buffalo, NY 14260, United States
xuanyudo@buffalo.edu, yifuyin@buffalo.edu, tiehangd@buffalo.edu

Abstract. This project includes two interesting implementations, one on PCA dimensionality reduction algorithm and the other on Apriori algorithm. PCA serves the purpose of dimensionality reduction, and helps data visualization when the dimensionality is reduced to 2, and is mainly used in the field of computer vision to reconstruct images, while Apriori algorithms is widely used in database systems to infer useful rules out of transactions. In this project, we illustrate on the implementation details and also demonstrate our experiment results. **Our team made two different original implementations for each of these algorithms, as can be seen on our team's Github page: https://github.com/xuanyudo/Pca˙Apriori-Algorithm.**

1 Model Description

1.1 Principle Component Analysis

Principle component analysis is built on linear algebra. It can represent data in a more compact form by (1): get the covariance matrix of the dataset, (2): decompose the covariate matrix into a set of eigen vectors and eigen values, (3) pick out the eigen value, eigen vector pairs with the top k largest eigen value, if we want to visualize the data, then k will be set to 2 or 3. (4) The new set of coordinates for each of the data in the dataset will be the multiplication of original coordinates with each of the k eigen vectors, with each data point having k coordinates.

1.2 Apriori Algorithm

The Apriori algorithm mines the association rules out of a frequent set given a set of transactions. The algorithm contains two steps: (1) generate a set of frequent items which exceeds the required support threshold, and (2) generate rules based on the frequent item sets which satisfies the confidence threshold. Direct computation of the frequent item sets and rules is infeasible given the exponential nature of the possible combinations, and further improvements including pruning technique to reduce the number of candidates, vertical based mining algorithms to reduce the number of transactions, and efficient data structures (hash tables) to reduce the number of comparisons is introduced in the field.

2 Model Implementation

2.1 Implementation of PCA

In this implementation, we encapsulated the whole functionality of PCA algorithm into one "PCA" class. After reading in the file and parse the dataset into a data matrix, the "compute covariant" function computes the covariance of the data matrix, whose dimensionality is of size $feature\ No. \times feature\ No.$. Then we called the "np.linalg.eig" function to calculate the eigen value and eigen vector of this covariance matrix, after which the two eigen value, eigen vector pairs with the largest eigen value is kept and serves as the new axises for the plot. Then, we get the coordinates of the each of the data point under this new set of coordinates, which is done by getting the inner product of the data point with each of the 2 eigen vectors. For the plot, we used a dictionary to store all the available colors which matches with the category that each data point belongs to.

2.2 Application of TSNE Method

Based on the project guidance, we adopted readily available packages from sklearn.manifold for creation of the tsne plot. In the package, we call the TSNE function and set the parameter n_components to be 2, so as it can be plotted as a two dimension scatter graph, and the data matrix is feed into the fit_transform method.

2.3 Application of SVD Method

Based on the project guidance, we adopted readily available packages from np.linalg.svd for creation of the SVD plot. We attached visualization of the result below.

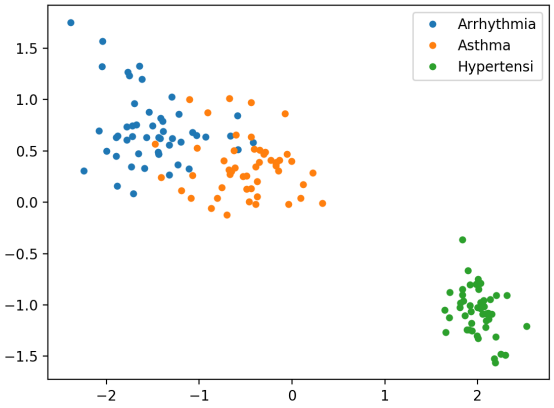
2.4 Implementation of Apriori Algorithm

3 Experiment Result

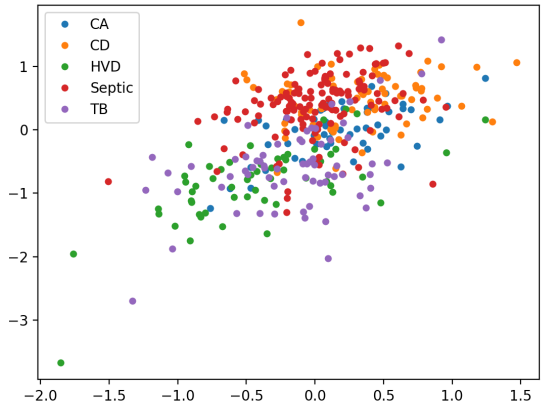
3.1 Result on PCA Analysis

After running PCA algorithm on the three given data files (pca_a.txt, pca_b.txt, pca_c.txt), we plot the result in Figure 1.

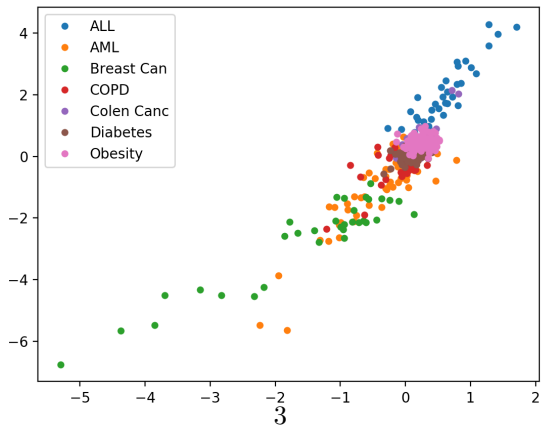
We can see the pca_a data is the easiest to separate among all three dataset, as shown by the different colors denoting the different classes, the different categories of data is not overlapping with each other and is easily distinguishable. For pca_b and pca_c, overlapping among the different clusters exists. Taking a closer look into the result of pca_c, we can see the clusters of data for Breast Cancer, AML and COPD are not separable in the PCA plot.



(a) PCA visualization of pca_a.txt data file

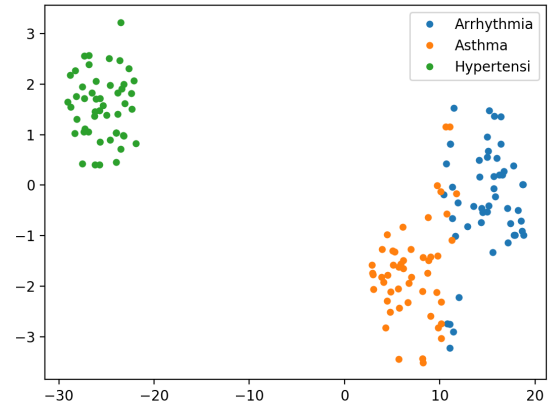


(b) PCA visualization of pca_b.txt data file

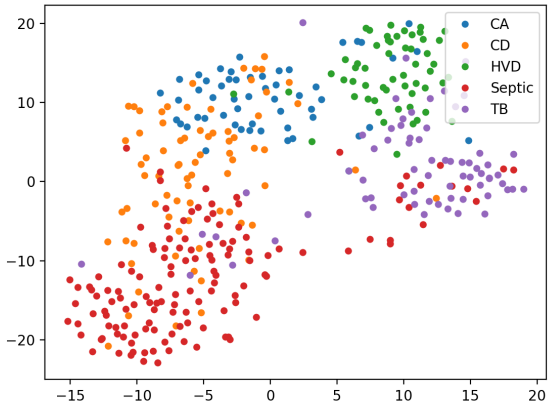


(c) PCA visualization of pca_c.txt data file

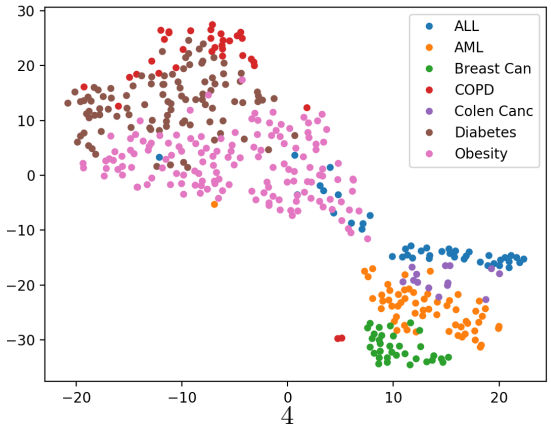
Fig. 1: (a) PCA visualization of pca_a.txt data file, (b) PCA visualization of pca_b.txt data file, (c) PCA visualization of pca_c.txt data file.



(a) TSNE visualization of pca_a.txt data file

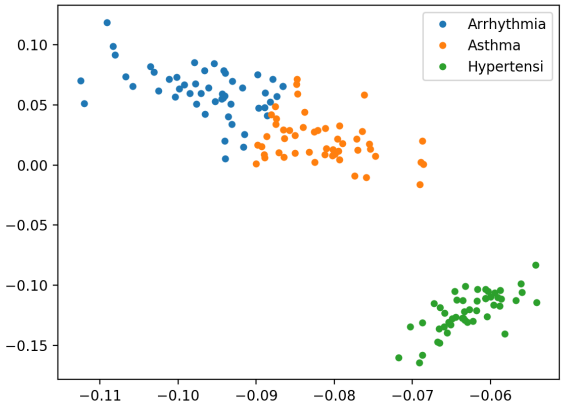


(b) TSNE visualization of pca_b.txt data file

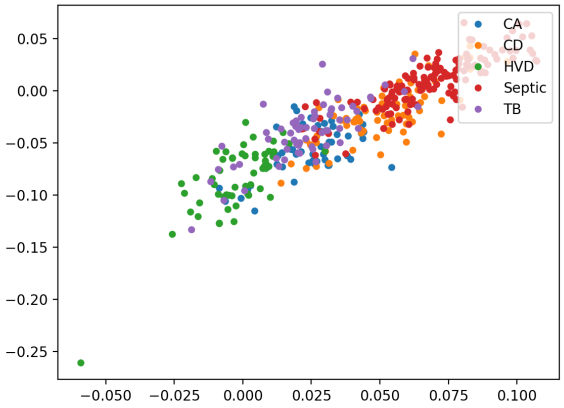


(c) TSNE visualization of pca_c.txt data file

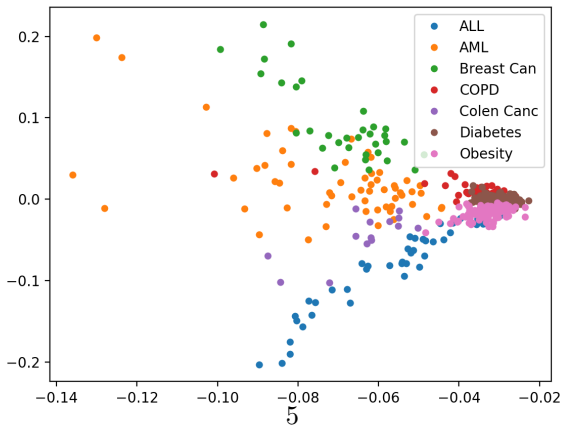
Fig. 2: (a) TSNE visualization of pca_a.txt data file, (b) TSNE visualization of pca_b.txt data file, (c) TSNE visualization of pca_c.txt data file.



(a) SVD visualization of pca_a.txt data file



(b) SVD visualization of pca_b.txt data file



(c) SVD visualization of pca_c.txt data file

Fig. 3: (a) SVD visualization of pca_a.txt data file, (b) SVD visualization of pca_b.txt data file, (c) SVD visualization of pca_c.txt data file.

We further adopted the TSNE and SVD method to visualize the data, the result of which are shown in Figure 2 and Figure 3. Comparing the result of PCA with TSNE, we can see the TSNE method is achieving similar performance as PCA in pca_a dataset, while having better separation effect for the pca_b and pca_c dataset. This is in accordance with the observations of researchers in the field, as currently TSNE has become one the most widely adopted visualization methods in the field of data mining and bioinformatics.

The effectiveness of SVD is similar to PCA, which successfully separates pca_a dataset and not effective on pca_b dataset. The performance on pca_c data lies in between the other two.

3.2 Result on Apriori Algorithm

Based on our implementation of Apriori algorithm as described in the previous section, the number of frequent item set with respect to different itemset length and different support threshold is summarized in Table 1. We can see the number of frequent item set decreases as support threshold increases. When support threshold is small such as 0.2 or 0.3, as length of itemset increases, the number first increases and then decreases, which is the tradeoff between the number of combinations and the appearance of the itemset in the transactions. Please see our github page at https://github.com/xuanyudo/Pca˙Apriori-Algorithm for the set generated.

Table 1: The number of frequent item set with respect to different itemset length and different support threshold, we can see the number of frequent item set decreases as support threshold increases.

Support Set Length	0.2	0.3	0.4	0.5	0.6	0.7
1	202	196	167	109	34	7
2	15,014	5,340	753	63	2	0
3	125,664	5,287	149	2	0	0
4	119,945	1,518	7	0	0	0
5	56,408	438	1	0	0	0
6	28,894	88	0	0	0	0
7	15,710	11	0	0	0	0
8	8,049	1	0	0	0	0
9	3,557	0	0	0	0	0
10	1,210	0	0	0	0	0
11	289	0	0	0	0	0
12	43	0	0	0	0	0
13	3	0	0	0	0	0
14	0	0	0	0	0	0