PCA Report - Dimensionality Reduction

Weiyang Chen[1] and Zhiwen Huang[2]

[1]UBID : 50141904
[2]UBID : 50143439

October 6, 2020

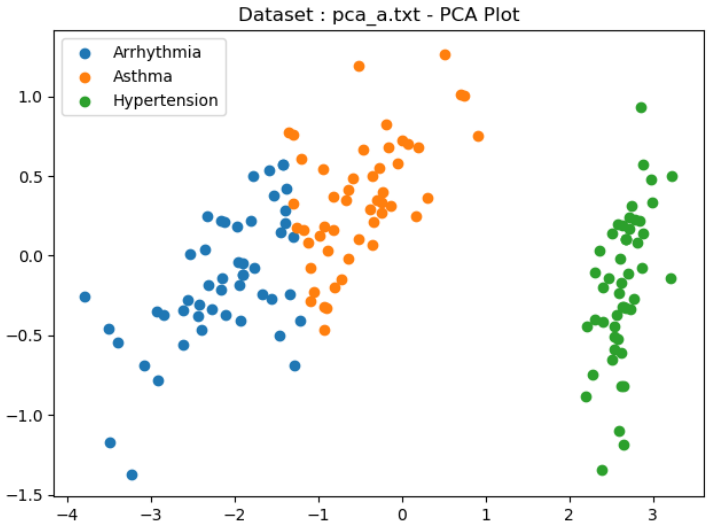
Abstract

The aim of this part is to implement PCA algorithm to do dimensional reduction without using existing library, also implement the SVD, and T-SNE Algorithm by using existing library and apply three giving data file "pca_a.txt", "pca_b.txt" and "pca_c.txt" to three implemented algorithm to draw the scatter plot graph and compare the result.

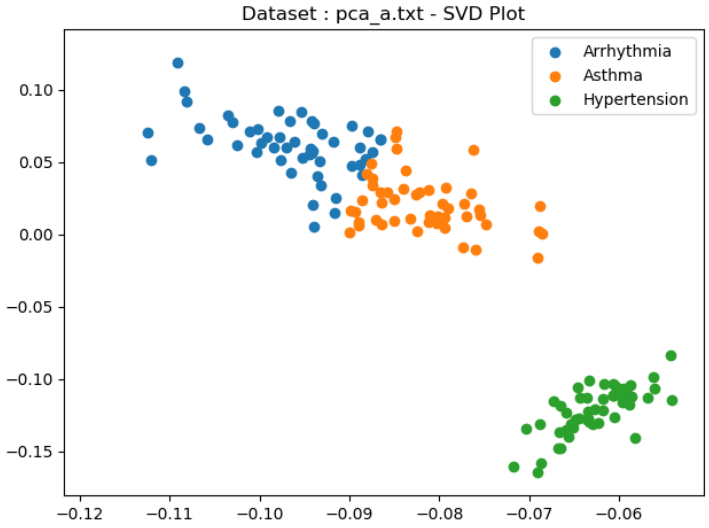
1 Scatter Plot

1.1 Data Set : PCA_A.txt

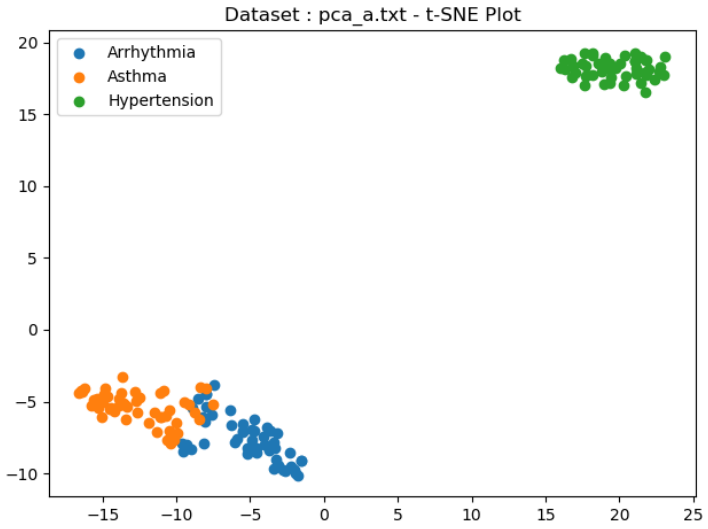
1.1.1 Algorithm : PCA



1.1.2 Algorithm : SVD

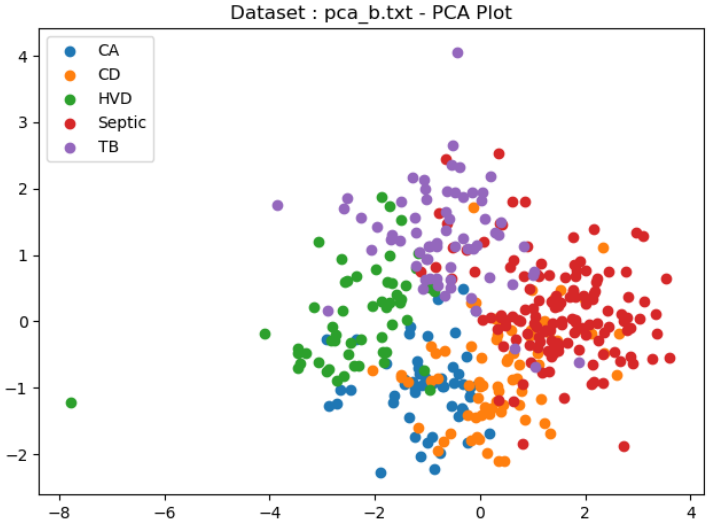


1.1.3 Algorithm : TSNE

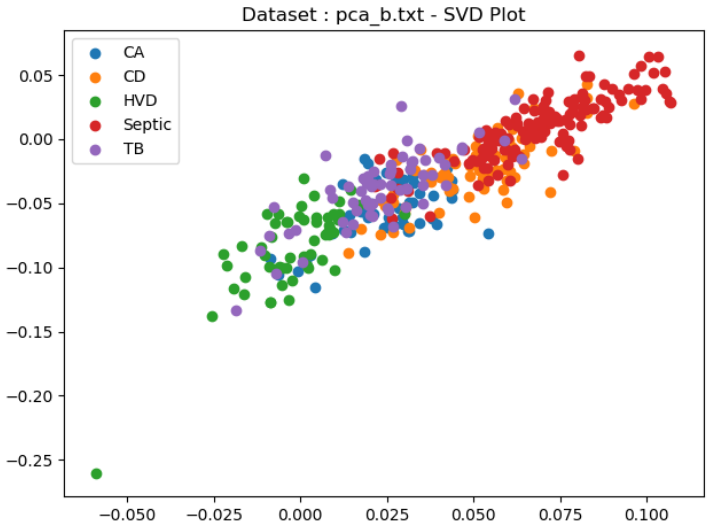


1.2 Data Set : PCA_B.txt

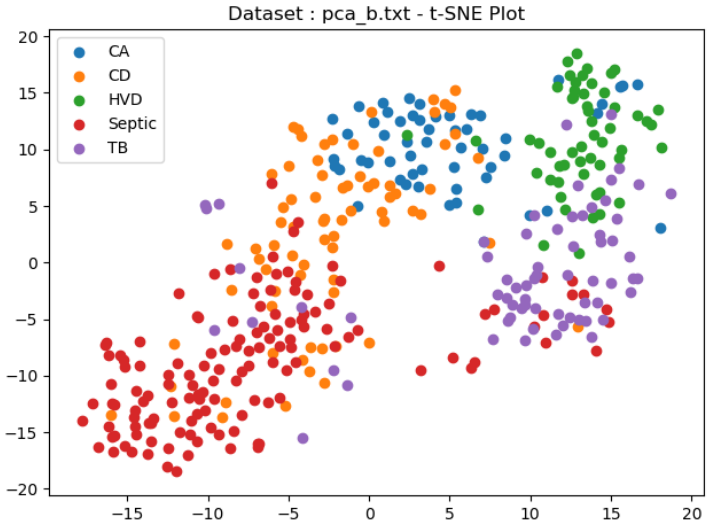
1.2.1 Algorithm : PCA



1.2.2 Algorithm : SVD

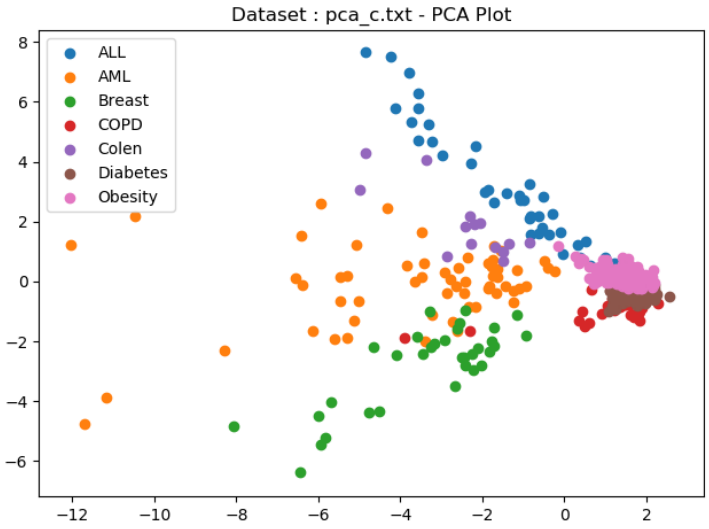


1.2.3 Algorithm : TSNE

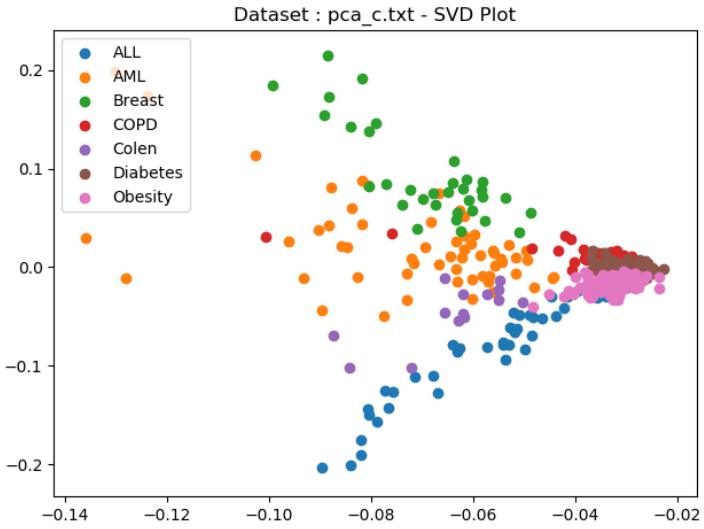


1.3 Data Set : PCA_C.txt

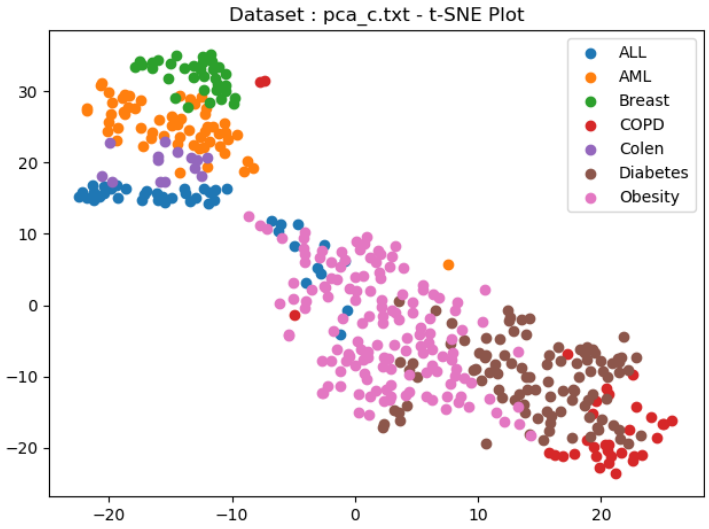
1.3.1 Algorithm : PCA



1.3.2 Algorithm : SVD



1.3.3 Algorithm : TSNE



2 PCA Implementation

2.1 Data Pre-processing

Read the txt file, and split the data into two matrix M and N. The first matrix M is $n * m$ matrix which each row represent one disease and each column represent one attribute. The second matrix N is $n * 1$ matrix which each row represent one disease name corresponding to the number of rows in the first matrix.

2.2 PCA Step (Input : M , N , K)

M : $n * m$ matrix, contain n disease and m attributes.

N : $n * 1$ matrix, contain n disease name corresponding to the number of rows in the M.

K : Integer data type, it means Dimensional reduce to K.

Algorithm :

1. Calculate the mean of each attribute (Find each mean $Mean_j$ of $M[:, j]$ for $j = 0, 1, 2...m - 1$).
2. Each element minus the mean of the corresponding column number to get new matrix M' ($M' = M[i, j] - Mean_j$ for $i = 0, 1, ..n - 1$ and $j = 0, 1, 2...m - 1$).

3. Calculate the Covariance Matrix Cov_M of M'^T by using *numpy.cov* function.($Cov\_M = np.cov(M'.T)$)

4. Calculate the eigen value and eigen vector of Covariance Matrix Cov_M by using *numpy.linag.eig* function($eig\_value, eig\_vector = np.LA.eig(Cov\_M)$).

5. Find top K eigen values, and their corresponding eigen vector. (In this project, we set K as 2 to find top 2)

6. Using top K eigen vector to calculate new K attributes' value. Each eigen vector has length m, so we can build them together as a eigen vector matrix with size (K*m) which means each row represent one eigen vector. Using M'^T dot product with eigen vector matrix to get Dimensionality Reduction matrix. (Reduction Matrix = k * m * m * n = k * n, Final Matrix = Reduction Matrix$^T = n * k$).

Draw Scatter Plot :
After we get the Final Matrix with size $n * k$ from above algorithm. Then using matrix to draw the graph, each row represent each disease and k = 2 means we have 2 attribute for each disease, we can look two attributes as x and y coordinate in the graph. And in the N matrix , we can find the disease which corresponding to its number of rows.

2.3 SVD and t-SNE

Similar to the PCA, apply 2-D data into existing library to get result of dimensional reduction. For the SVD, we used *numpy.linalg.svd* function to compute the result matrix. For the t-SNE, we used $sklearn.manifold.TSNE$ function to compute the result matrix.

3 Algorithm Comparison

3.1 PCA vs SVD

The problems solved by the two of them are very similar. They both perform eigenvalue decomposition on a real symmetric matrix. But, PCA is solving the eigenvalue decomposition of the covariance matrix, and SVD is solving eigenvalue decomposition of the $A * A^T$ matrix. Through the concept of PCA, we can know that PCA can only obtain the principal components in a single direction.

$$A * A^T = (U\Sigma V^T)U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T \quad (1)$$

But as equation one shows, SVD can get the principal components in both left and right direction, because it has left and right singular vector which are V and V^T in the equation.
In the plot graph, we can also see the result of PCA and SVD are similar.

3.2 PCA vs t-SNE

t-SNE uses machine learning to project multi-dimensional data into a two-dimensional plane, So when we use the t-SNE library, we need to input many parameters, for example learning rate, iterations, perplexity and so on. and the result will not fix which means We can't predict the result of its return. So it is much different from the PCA that the result will be always fix that using fix formula and calculate with the fix known variables. But as Scatter Plot shows, t-SNE has better performance than PCA.