SMAI Project

Mid Eval Report

Team Kaaju

Aaradhya Gupta (2019114010)
Akhilesh Aravapalli (2019114016)
Chayan Kochar (2019114008)
Jayant Panwar (2019114013)

Github Repo Link: https://github.com/ChayanK2000/SMAI_Project

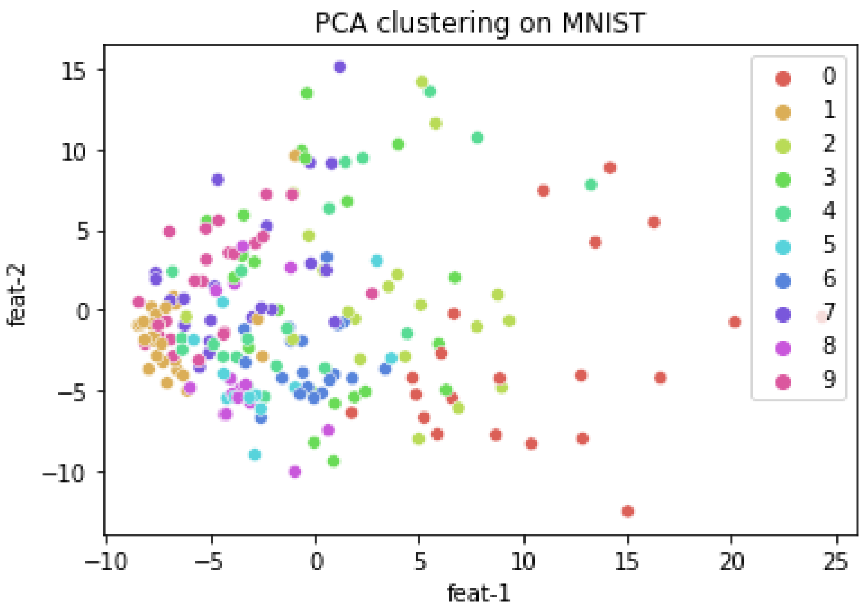
---

Dataset & Preprocessing

Basic overview:

- Collected the MNIST dataset
- Done basic preprocessing of the MNIST Dataset.
- Applied PCA to the dataset for comparison.

We are using the **MNIST digits dataset** - importing from keras. Its train dataset has 60000 samples with 28 x 28 features. On loading it and converting it from 28 x 28 to 784 x 1, we applied PCA to reduce the dimensions from 784 to around 30.
This was done to reduce the complexity for the TSNE algorithm according to what was mentioned in the paper, and also to have a comparison/basic visualization of the data by putting the no of dimension = 2 in PCA. The following is a graph of the PCA output with dimension = 2 of the first 200 samples.



Implementing the tSNE algorithm

Basic Overview:

- Gone through some literature surrounding t-SNE and its versions.
- Implemented t-SNE algorithm.
- The t-SNE algorithm that has been implemented only works for smaller datasets due to high time complexity.

As mentioned above, the MNIST dataset had 60000 samples of 784 dimensions. Though we did apply PCA but the current code implementation was not able to handle such a huge dataset of 60000 samples. This is why we are also thinking of getting our hands dirty with the **UCI digits dataset**(by importing load_digits from sklearn.datasets) in which each of the 1797 samples has only 8 x 8 = 64 dimensions.

For implementing the tSNE algorithm, we followed this paper provided to us:
https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Currently we have just given random dummy data to the tSNE algorithm to give some reasonable output. We tried to compute all the variables as mentioned and required. We wrote it in a modular fashion keeping in mind the various calculations we needed to do for various parts in the algorithm. Thus we have made different function which calculates:

- the neighbours for each point/sample
- Q_{ij}
- P_{ij}
- Kullback-Leibler Cost function

The functions which implement the Gradient Descent using the Kullback-Leibler Cost function and the function to get neighbours for every point will need a review to improve the complexity and correctness of our code.

Future Work:

- Optimize the t-SNE algorithm to run it on bigger datasets (UCI/MNIST).
- Implement other visualisation algorithms for comparative study.
- Study and report the visualisations from all the algorithms on the same dataset.