1. INTRODUCTION

Due to the recent surge in data, handling of such huge amount of data has placed large number of challenges. The data that is recorded has reached deeper level of granularity increasing the dimensions to be handled. Dimensionality reduction is process of using lesser number of dimensions without losing much information. It facilitates reduction in storage space and helps in data compression. It also helps reduce the computation time that raises steeply for a dataset with higher dimensionalities. Another important use of Dimensionality reduction is in visualization of data. Data Analysis provides important insights into the data to take important decisions about the businesses. Visualization of data with higher dimensions quite is not clear and does not help in decision making.

For visualizing the structure of very large datasets, we show how t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed. We illustrate the performance of t-SNE on two large datasets with variety of features and compare it with conventional dimensionality reduction techniques, like PCA. The results produced by t-SNE are significantly better than those produced by the other techniques on both of the datasets.

2. T-SNE

Stochastic neighbor Embedding or other Dimensionality reduction techniques aims to reduce the error of classification when the dimensions are scaled down. In order to achieve this, SNE uses conditional probability to match distance distributions between the high and low dimensionality points. In high dimensionality SNE uses a conditional probability that is based on the distances raised to exponential. When the points which are close to each other, the exponentiation term approaches 1 and the conditional probability increase of points being near to each other. In low dimensionality, the variance is not used. The less the difference between these two conditional probabilities, the better the scaling to the low dimension from high dimension. For doing so it uses KL divergences as the cost function. Cost function is symmetrized, the variance term in high dimensionality does not depend on variance and the conditional probability of two points are same. t-SNE uses optimization parameter making it better than SNE.

The other techniques that are prevalent are LDA and CPA. LDA suffers largely from the fact that is based on the supervised learning technique making it largely dependent on training data.

Principal Component Analysis (PCA): PCA tries to establish relationship between the features and eliminate the features which are redundant. It produces a set of linearly uncorrelated components which are referred as principle components. This makes PCA unsuitable for data where the data points are not normalized and fails in case of complicated relationship between the features. It will not be able to interpret complex polynomial relationship between features. On the other hand, t-SNE is based on probability distributions with random walk on neighborhood graphs to find the structure within the data. A major problem with, linear dimensionality reduction algorithms is that they concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is important that similar datapoints must be represented close together, which is not what linear dimensionality reduction algorithms do.

3. ALGORITHM AND IMPLEMENTATION

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: dataset $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for t=1 to T do
        compute low-dimensional affinities $q_{ij}$ (using Equation 12)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 13)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
    end
end
```

Figure 1. Algorithm for t-SNE

Step 1

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The similarity of datapoint x_i to datapoint x_j is the conditional probability, $p_{j|i}$, x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_j. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ_i). Mathematically, the conditional probability $p_{j|i}$ is given by –

$$p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\,\sigma_i{}^2)}{\Sigma_{k \neq i}\ \exp(-\left\|x_i - x_k\right\|^2 / 2\,\sigma_i{}^2)}$$

where σ_i is the variance of the Gaussian that is centered on datapoint x_i

Step 2

For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$.

$$q_{j|i} = \frac{\exp(-\left\|y_i - y_j\right\|^2)}{\Sigma_{k \neq i}\ \exp(-\left\|y_i - y_k\right\|^2)}$$

Step 3

To measure the minimization of sum of difference of conditional probability SNE minimizes the sum of Kullback-Leibler divergences overall data points using a gradient descent method. SNE cost function focuses on retaining the local structure of the data in the map for reasonable values of the variance of the Gaussian in the high-dimensional space σ_i.

Step 4

It is not likely that there is a single value of σ_i that is optimal for all data points in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i , over all of the other data points. This distribution has an entropy which increases asσ_i increases. t-SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by σ_i the user. The perplexity is defined as –

$$Perp\ (P_i) = 2^{H(P_i)}$$

where $H(P_i)$ is the Shannon Entropy of P_i measured in bits.

The minimization of the cost function is performed using gradient decent. And physically, the gradient may be interpreted as the resultant force created by a set of springs between the map point y_i and all other map points y_j . All springs exert a force along the direction $(y_i - y_i)$. The spring between y_i and y_j repels or attracts the map points depending on whether the distance between the two in the map is too small or too large to represent the similarities between the two high-dimensional data points.

IMPLEMENTATION USING PYTHON

The t-SNE algorithm is implemented in Python with helper libraries as listed below.

Libraries and packages	***Description***
numpy	*Computational purposes and datasets*
matplotlib	*Visualization of results through scatter plots*

Table 1. Python libraries

The methods and function definitions we used to implement the algorithm for feature reduction is listed below with the corresponding descriptions.

Function	
Htsne_beta()	Compute the perplexity and the P-row for a specific value of the precision of a Gaussian distribution.
convert_x2p()	Performs a binary search to get P-values in such a way that each conditional Gaussian has the same perplexity.

pca_preprocess()	Runs PCA on the NxD array X in order to reduce its dimensionality to no_dims dimensions.
tsne_function()	Runs t-SNE on the dataset in the NxD array X to reduce its dimensionality to no_dims dimensions. The syntaxis of the function is *Y = tsne.tsne(X, no_dims, perplexity)* where X is an NxD NumPy array.

Table 2. Functions Description

4. DATA DESCRIPTION

- **Human Activity Recognition database**

The dataset is built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data. The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain. The dataset contains split sets for training and testing. We leverage this and the technique of encoding the categorical dataset and using them to train the system.

For each record, we have -

1 Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
2 Triaxial Angular velocity from the gyroscope.
3 A 561-feature vector with time and frequency domain variables.
4 Activity labels.
5 An identifier of the subject who carried out the experiment

Data set Characteristics	Multivariate, Time Series
Associated Tasks	Classification, Clustering
Number of instances	10299
Number of attributes	561

- **MNSIT Dataset**

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

The MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than SD-1. The reason for this can be found on the fact that SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Therefore, it was necessary to build a new database by mixing NIST's datasets.

5. EVALUATIONS AND RESULTS

t-SNE algorithm was evaluated against the conventional dimensionality reduction algorithms. The datasets we considered to implement the algorithm included varied features. The performance was measured by calculating the accuracy and visualizing the results through scatter plots. The script was run on a system with 16 GB RAM and an Intel core i7 processor.

The table 5a shows the accuracy values for the datasets after 10, 50, 500 and 1000 iterations of the algorithm. The algorithm goes onto reducing the error value through iterations.

Dataset	Mean value of sigma	Error value after Iteration = 10	Error value after Iteration = 50	Error value after Iteration = 500	Error value after Iteration = 1000	Execution time (in secs)
MNIST	2.386597	23.621053	16.590112	1.084097	1.032242	180
Human Activity recognition	1.160265	21.07652	14.324561	0.823432	NA	NA

Table 5a. Result

The table 5b shows the accuracy values obtained with Random Forest classifier after reducing the features using t-SNE on the Human Activity Recognition dataset. We reduced the features to 2 from 561 so that it will be easy to represent in a 2D graph. The reduced attributes were used as inputs to the Random forest classifier. From the table 5b we can see that all the % activities can we clearly seen after reducing the features to 2 attributes.

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	Laying
Walking	306	0	0	0	0	0
Walking Upstairs	3	240	0	1	0	0

Walking Downstairs	0	1	236	0	0	0
Sitting	0	0	0	298	51	0
Standing	0	0	0	49	301	0
Laying	0	0	0	0	0	352

Table 5b. Confusion matrix for dataset Human activity analysis

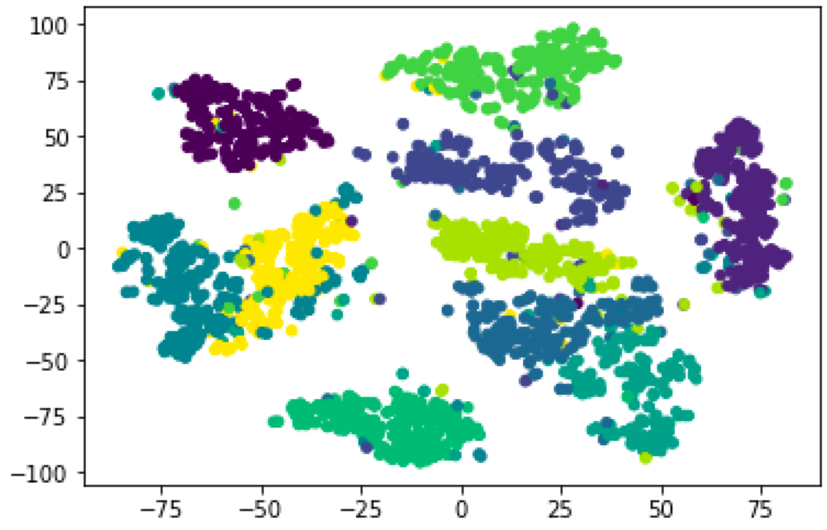

Figure 5c. Visualization of MNSIT feature reduction using t-SNE

Figure 5c and 5d show the scatter plots we obtained after specified iterations of the algorithm on both the datasets for dimensionality reduction. The accuracy obtained from Random Forest Classifier from sk-learn is 92.28% and by using the Random-Forest Classifier implemented by us was 81%.

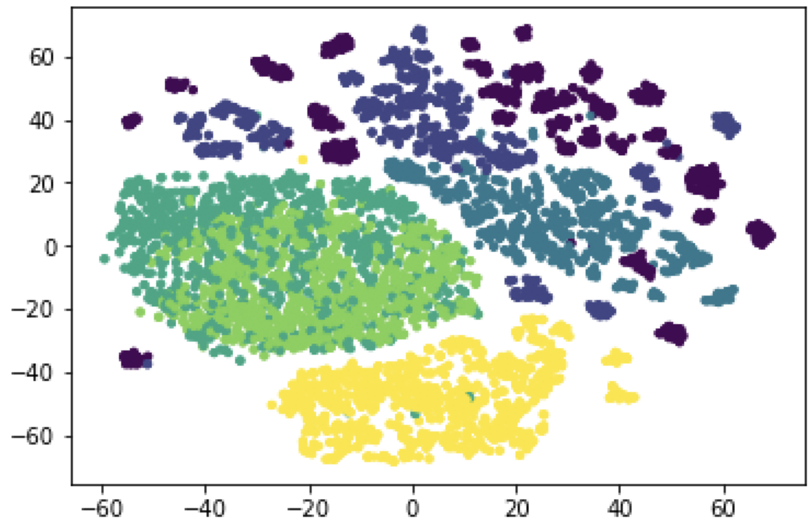

Figure 5d. Visualization of feature reduction using t-SNE for human activity recognition

Time and Space complexity

The algorithm computes pairwise conditional probabilities and tries to minimize the sum of the difference of the probabilities in higher and lower dimensions. This involves a lot of calculations and computations. We observed that the algorithm is quite heavy on the system resources and has a quadratic time and space complexity in the number of data points.

6. CONCLUSION

The project presents a technique for the visualization of similarity data that is capable of retaining the local structure of the data while also revealing some important global structure. Our experiments on a variety of data sets show that t-SNE outperforms existing state-of-the-art techniques for visualizing a variety of real-world data sets.

7. REFERENCES

[1] Visualizing Data using t-SNE
http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
[2] Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types
https://www.nature.com/articles/s41467-017-01689-9
[3] How to Use t-SNE Effectively
https://distill.pub/2016/misread-tsne/
[4] T-SNE visualization of large-scale neural recordings.
https://www.authorea.com/users/94202/articles/110467-t-sne-visualization-of-large-scale-neural-recordings/_show_article
[5] Application of t-SNE to human genetic data
https://www.worldscientific.com/doi/abs/10.1142/S0219720017500172
[6] TSNE vs PCA
https://www.kaggle.com/vimary/tsne-vs-pca
[7] Comparison of Principal Component Analysis and t-Stochastic Neighbor Embedding with Distance Metric Modifications for Single-cell RNA-sequencing Data Analysis
https://www.biorxiv.org/content/early/2017/01/29/102780
[8] Playing with dimensions: from Clustering, PCA, t-SNE… to Carl Sagan!
https://www.r-bloggers.com/playing-with-dimensions-from-clustering-pca-t-sne-to-carl-sagan/
[9] SAR target recognition using parametric supervised t-stochastic neighbor embedding
https://www.tandfonline.com/doi/abs/10.1080/2150704X.2017.1332795
[10] Data-driven identification of prognostic tumor subpopulations using spatially mapped t-SNE of mass spectrometry imaging data http://www.pnas.org/content/113/43/12244.full