Emre Yavaş
emreyavas@sabanciuniv.edu

EE566 PATTERN RECOGNITION

TERM PROJECT

t-Distributed Stochastic Neighbor Embedding

1.Introduction

Dimensionality reduction is an important topic in many different domains. The main goal of dimensionality reduction is reducing the number of features of data while preserving the information. Dimensionality reductions techniques are widely used in lots of machine learning and deep learning algorithms. It is generally used to gain computation power to save time and avoid curse of dimensionality problem. Dimensionality reductions mainly divided into two categories: linear (PCA [2], LDA, etc.) and nonlinear (Isomap, Umap, etc.) approaches. For this project I implemented t-SNE algorithm from scratch and apply it on MNIST and sklearn digits datasets.

2.Method

T-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique which is mainly used to visualize high dimensional data. It typically maps high dimensional data to 2 or 3 dimensions[1]. Visually it is expected that the elements of same class should be mapped closed to each other. Stochastic neighbor embedding (earlier version of t-SNE) algorithm converts Euclidean distances between datapoints into conditional probabilities to represent similarity. Similarity between two datapoints, x_i and x_j, is the conditional probability $p_{j|i}$. It means that 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_i.

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

Equation1. $p_{j|I}$

Emre Yavaş
emreyavas@sabanciuniv.edu

Using a constant σ_i as a hyper parameter might not reflect local similarity between datapoints. So, different σ_i values are computed. Computation of σ_i is presented later in this section. To create low dimensional space, same structure is used but variance is set to $\frac{1}{\sqrt{2}}$ to get simple equation.

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)}$$

Equation2.$q_{i|j}$

If the mapped points y_i and y_j correctly model the similarity between the high dimensional versions x_i and x_j, the conditional probabilities $p_{i|j}$ $q_{i|j}$ will be equal. Kullback Leibler divergence, one of the most popular probability measures, is used. SNE minimizes the sum of KL divergences over all datapoints using a gradient descent method.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}},$$

Equation3. KL divergences of P_i Q_i.

In order to find σ_i perplexity is used. Perplexity can be interpreted as a smooth measure of effective number of neighbors. It is typically between 5-50.

$$Perp(P_i) = 2^{H(P_i)}, \qquad H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

Equation4. Perplexity

Since closed form of σ_i is difficult to compute binary search method is to find out.

But the cost of SNE method for mapping nearby points to widely separate points is small. Later in Symmetric SNE this problem is solved using joint probability.

$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2/2\sigma^2\right)},$$

Equation5. P_{ij} (Joint probability)

This time algorithm becomes more vulnerable to outliers. In t-SNE p_{ij} is set $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2*n}$. This ensures that $\Sigma_j p_{ij} > 1/2n$ for all datapoints x_i. Another problem is in high dimensional space there are more space for mutually equidistant points. In low dimensional space there may not be enough room for whole data which is called Crowding Problem. To eliminate crowding problem in new dimensional space t-distribution with one degree of freedom is used due to its heavy tailed map. So q_{ij} becomes:

Emre Yavaş
emreyavas@sabanciuniv.edu

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}.$$

Equation6. Q_{ij} (Joint Probability)

With the new p_{ij} and q_{ij} gradient of KL Divergences has changed to:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}.$$

Equation7. Gradient of KL divergences

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

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

Algorithm1. t-SNE

2.1 IMPLEMENTATION DETAILS

Before applying t-SNE, data is flattened and normalized and then PCA (written from scratch) is applied to the data as suggested in the paper. To ensure numerical stability if a probability is less than e^{-700} (which is around 10^{-304} (almost zero)) it is set to e^{-700}. Several hyperparameters are needed for the code:
Perplexity and the tolerance for it, number of components of PCA, initial values of y, learning rate and momentum in gradient descent. In experiments and results section their effects are discussed.

3.Experiments and Results

The algorithm is applied to famous MNIST dataset and sklearn digit datasets. MNIST is a handwritten digit dataset with 70000 images and each image has (24*24) 784 pixels. Sklearn digit dataset is a smaller digit dataset. It has 1797 images and each of them consists of only (8*8) 64 pixels.

Emre Yavaş
emreyavas@sabanciuniv.edu

Digits Dataset Results: 800 images taken from the dataset and always the same images are used. For all experiments:

Maximum number of iterations for gradient update is 120. Sigma is computed via binary search and perplexity has 10^{-2} tolerance. (Computed perplexity = target perplexity ± 0.01).

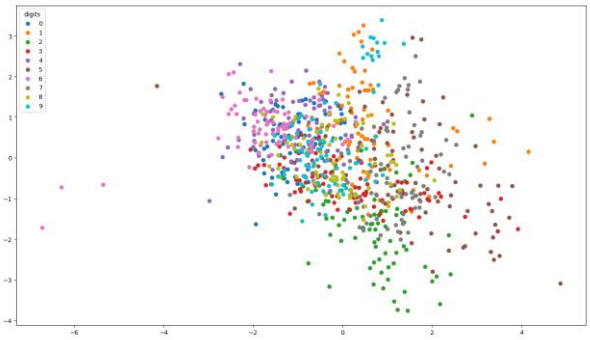

Fig1. Initial values of experiment 1-2-3-4-6 (PCA with 2 dimensions)

For the first and second experiment: Learning rate change is checked. (PCA is used to reduce the data to 30 dimensions. Learning rate is set to 200 and 10, Momentum=0.9. As initial points of y, PCA with 2 dimensions is used. Perplexity=40)

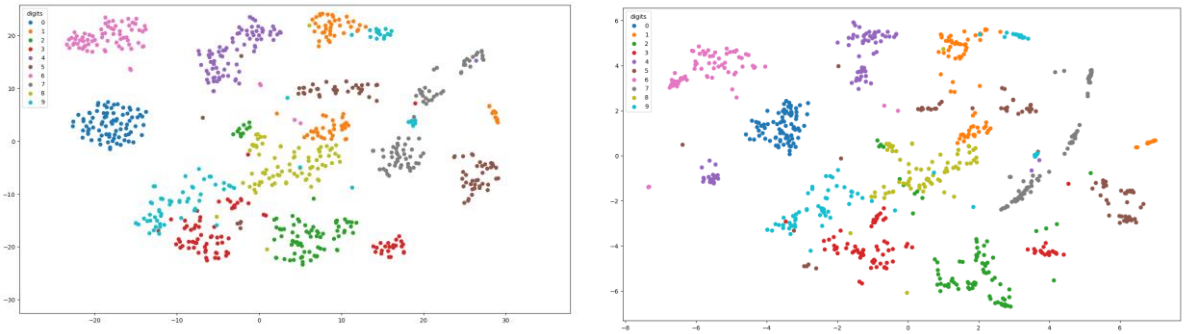

Fig2. Final values of experiment 1 and 2(LRate 200 vs 10)

For the third and fourth experiment: Perplexity effect is checked. (PCA is used to reduce the data to 30 dimensions. Learning rate is set to 100, Momentum=0.9. As initial points of y, PCA with 2 dimensions is used. Perplexity=10 and100)

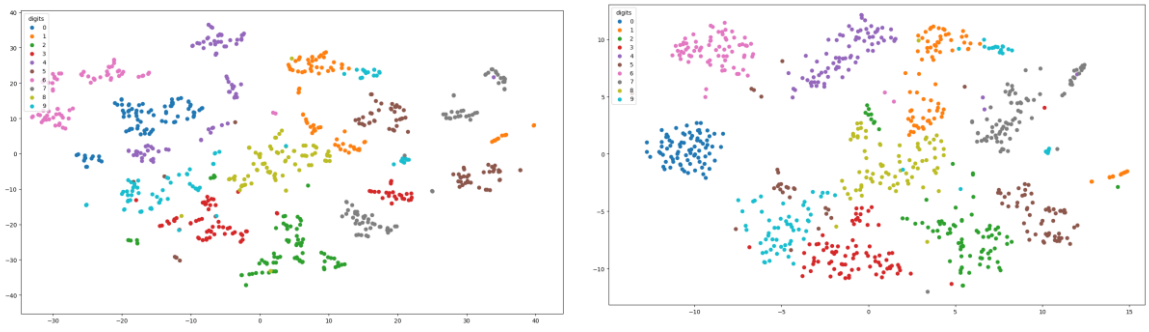

Fig3. Final values of experiment 3 and 4(Perplexity10 vs 100)

Emre Yavaş
emreyavas@sabanciuniv.edu

For the fifth and sixth experiment: Initial value effect is checked. (PCA is used to reduce the data to 30 dimensions. Learning rate is set to 100, Momentum=0.9. As initial points of y, multivariate random distribution with 0 mean, unit covariance and pca with 2 dimensions are used. Perplexity=80)

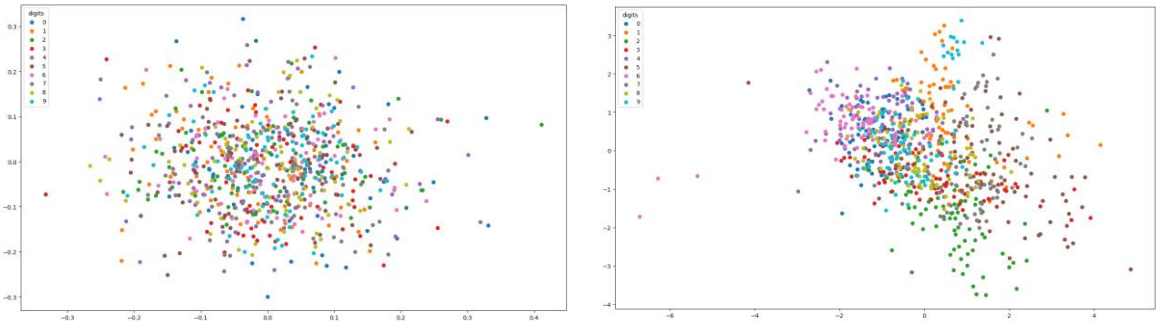

Fig4. Initial values of experiment 5 and 6 (Random vs PCA_2 initializations (Fig1))

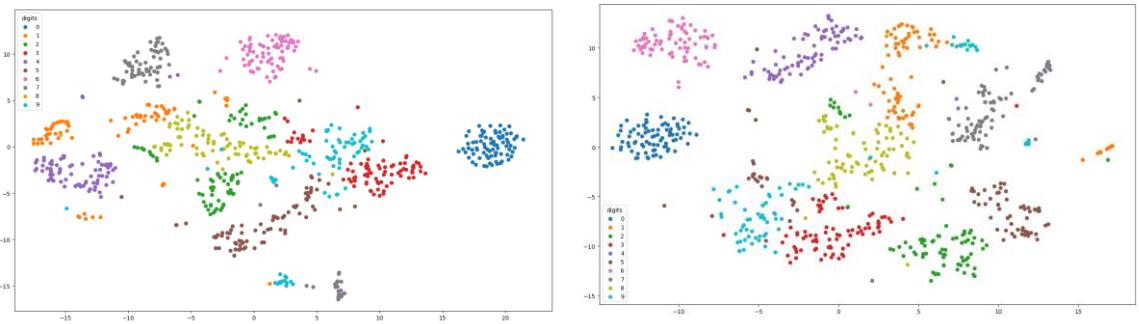

Fig5. Final values of experiments 5 and 6 (Random vs PCA_2 initializations)

For the seventh experiment: Initial PCA effect is checked. (PCA is “not” used. Learning rate is set to 100, Momentum=0.9. As initial points of y, multivariate random distribution with 0 mean, unit covariance. Perplexity=80)

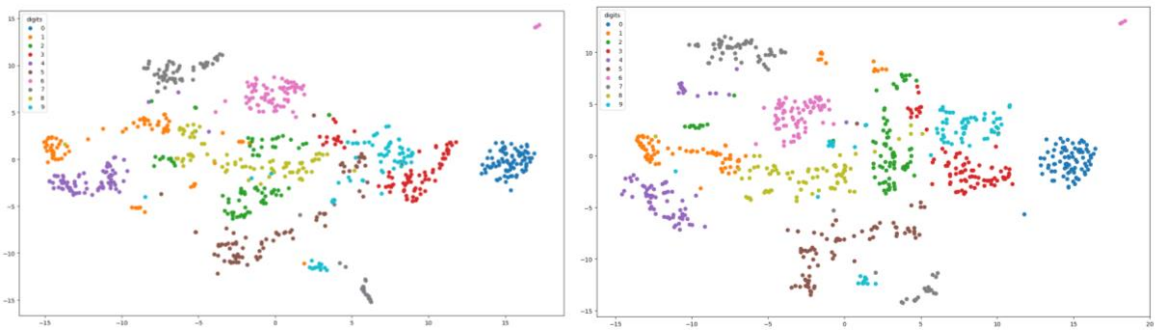

Fig6. 40’th epoch values of experiment 5 and 7

(PCA is applied to data first vs whole data is used)

Emre Yavaş
emreyavas@sabanciuniv.edu

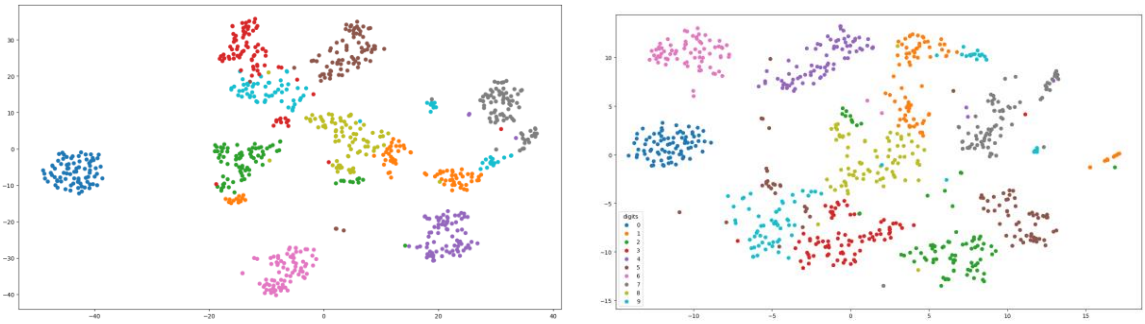

Fig7. Built-in t-SNE vs experiment 6

MNIST dataset: Since it has much more pixels the computation time is quite long. So, I can only use 120 epoch for 1200 MNIST data image. (digits learning rate is 10 perplexity 40.)

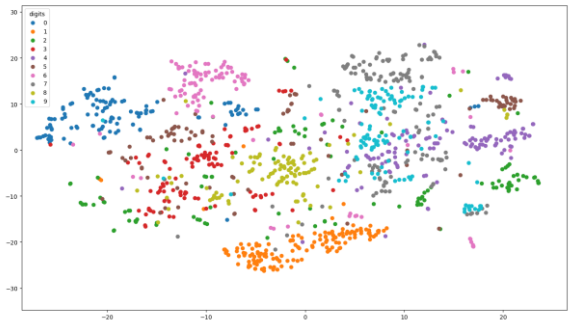

Fig8. MNIST Result for 1200 images

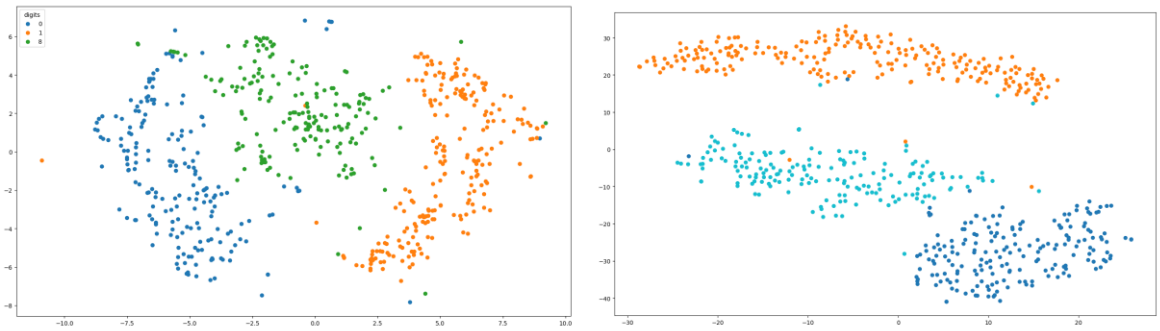

Fig9. MNIST result for 600 data (only 0-1-8 digits) (my t-SNE vs built-in)

4.Discussion

From first two experiments one can observe that 200 is a better learning rate. However, if algorithm iterates more with lower learning rate, they are expected to become so close. From figure3 100 perplexity is more suitable for this dataset. Perplexity is proportional to number of neighbors. It seems that for 800 data with 10 different classes perplexity=100 is a better option. On the other hand, from figure 5 initialization of y does not affect the result that much. From figure 6 we can say that using PCA at the beginning does not affect the result much, but it significantly speeds up the process (5 min vs 15 min). That is why 40 epochs are used. Figure 7 compares my t-SNE with built-in t-SNE. Built-in version is working a little bit better. But the main difference is it works much faster than mine. It is mainly because built-in

functions use better array manipulation techniques. Experiments with MNIST dataset takes much more time. But from fig8 we can say that it starts to separate the classes. To save time I use less data with only 3 digits at figure 9. It separates the data with some anomalies.

5.References

[1] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

[2] H. Abdi and L. J. Williams, “Principal component analysis,” Wiley interdisciplinary reviews: computational statistics, vol. 2, no. 4, pp. 433–459, 2010.

[3] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. Neural Networks, 1:295–307, 1988.