Parallel t-Distributed Stochastic Neighbor Embedding

Yuhao Zhou
Boston University
yuhaoz@bu.edu

Fengxu Tu
Boston University
fengxutu@bu.edu

1. Introduction

Dimensionality reduction is a important problem in machine learning, statistics, and some other domains. Through the visualization of low-dimensional data after dimensionality reduction, it can help us to understand the high-dimensional data itself better. There are many ways for dimensionality reduction such as PCA, NMF, t-SNE. This project focuses on t-SNE algorithm for visualization with some parallelizton method to accelerate the algorithm.

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton.[1] It is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.

2. t-Distributed Stochastic Neighbor Embedding

2.1. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding which is also called SNE maps data points to probability distribution through affinitie transformation. The mainly idea of SNE have two steps. The first step is that SNE constructs a probability distribution between high-dimensional objects, so that similar objects have a higher probability of being selected, while dissimilar objects have a lower probability of being selected. The second step is that SNE constructs the probability distribution of these points in the low-dimensional space so that the two probability distributions are as similar as possible.

Given a set of N high-dimensional objects $\mathbf{x}_1, \ldots, \mathbf{x}_N$, SNE first computes conditional probabilities $p_{j|i}$ that are proportional to the similarity of objects $\mathbf{x}_i$ and $\mathbf{x}_j$ as follows:

$$p_{j|i} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}}} \tag{1}$$

where σ_i is the variance of the Gaussian distribution that is centered on data point $\mathbf{x}_i$, and the conditional probabilities with $i = j$ are set to zero: $p_{i|i} = 0$.

For low-dimensional points $\mathbf{y}_i$, SNE set the $\sigma_i = \frac{1}{\sqrt{2}}$, and we get the equation 2:

$$q_{j|i} = \frac{e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq i} e^{-\|\mathbf{y}_i - \mathbf{y}_k\|^2}} \tag{2}$$

similarly, we set $q_{i|i} = 0$.

In order to keep all local features, SNE uses Kullback-Leibler divergence as cost function which shown as equation 3:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{3}$$

where P_i here represents the conditional probability distribution of all other data points at a given point $\mathbf{x}_i$. It should be noted that KL-divergence is asymmetric, and the penalty weights corresponding to different distances in the low-dimensional mapping are different, so SNE will tend to retain local features in the data.

SNE uses binary search to select σ_i which produces a P_i with a fixed perplexity that is specified by paper [1]. The perplexity is defined as follow:

$$Perp(P_i) = 2^{H(P_i)} \tag{4}$$

where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$. Therefore, using gradient descent to minimize cost function equation 3, we have equation 5:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(\mathbf{y}_i - \mathbf{y}_j) \tag{5}$$

The gradient update is shown as equation 6:

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)}) \tag{6}$$

where t is the number of iterations, $Y^{(t)}$ is the solution at iteration t, η is learning rate, and $\alpha(t)$ is the momentum at iteration t.

SNE is a good method for visualization, but it still has some problems which may affect the performance such as "crowding problem". So Hinton and others made some subsequent improvements bssed on SNE and got t-SNE.

2.2. t-SNE

The biggest difference between t-SNE and SNE is that t-SNE uses a joint probability distribution to replace the conditional probability distribution, and uses t-distribution to solve the "crowding problem".

2.2.1 Symmetric SNE

In t-SNE uses joint distribution instead of conditional probability distribution, where $p_{ij} = p_{ji} = 0$.The pairwise similarities in the high-dimensional space p_{ij} are given by:

$$p_{ij} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}}{\sum_{k \neq l} e^{-\frac{\|\mathbf{x}_k - \mathbf{x}_l\|^2}{2\sigma^2}}} \tag{7}$$

where p_{ii} is set 0. But it still has a problem with some outliers, so t-SNE just redefine the joint probability as equation 8:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{8}$$

Similarly, the pairwise similarities in the low-dimensional map q_{ij} are given by:

$$q_{ij} = \frac{e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq l} e^{-\|\mathbf{y}_k - \mathbf{y}_l\|^2}} \tag{9}$$

The new cost function is shown as follow:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{10}$$

2.2.2 Crowding problem

In order to solve "crowding problem", in high-dimensional space, t-SNE uses Gaussian distribution to convert distance to probability distribution, in low-dimensional space, t-SNE uses more long-tailed distribution to convert distance to probability distribution. Therefore, the medium and low distances in the high dimension can have a larger distance after the mapping. Equation 11 is the q_{ij} after using t-distribution:

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}} \tag{11}$$

According to equation 8, equation 10, and equation 11, we have:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1} \tag{12}$$

2.2.3 Algorithm

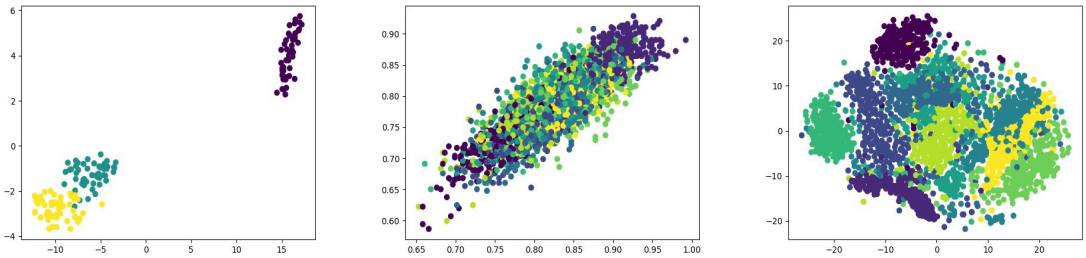
Algorithm 1: t-Distributed Stochastic Neighbor Embedding

Input: high-dimensional data $\mathbf{X}$
Compute perplexity $Prep$
Initialize iterations T, learning rate η, momentum $\alpha(t)$
Output: low-dimensional data $\mathbf{Y}$
begin
  computes pairwise affinities $p_{j|i}$
  computes $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  initialize $\mathbf{Y}$ from $\mathcal{N}(0, 10^{-4}I)$
for *t = 1, 2,* $\cdots, T$ **do**
  computes low-dimensional affinities q_{ij};
  computes gradient $\frac{\partial C}{\partial \mathbf{Y}}$;
  update $Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$;
end

3. Experiments

3.1. Data sets

Date sets used in this project are MNIST and Iris. The size of MNIST data set is 2500×784, the size of Iris data set is 150×4.



(a) Iris (b) Raw MNIST (c) t-SNE MNIST

Figure 1. Visualizations of Iris data sets

```
./test_mnist
# Reading Data ...
# tSNE Start ...
# Computing P-values ...
# Time Spent: 0ms
# Start Gradient Descent ...
# Iter: 100, err: 8.54428, time: 14s
# Iter: 200, err: 0.361679, time: 14s
# Iter: 300, err: 0.358529, time: 14s
# Iter: 400, err: 0.357205, time: 13s
# Iter: 500, err: 0.35648, time: 14s
# Iter: 600, err: 0.356172, time: 14s
# Iter: 700, err: 0.355964, time: 14s
# Iter: 800, err: 0.355953, time: 14s
# Iter: 900, err: 0.355953, time: 13s
# Iter: 1000, err: 0.355953, time: 14s
```

(a) Parallel

```
g++ -D NO_PARALLEL -Wall -O2 test.cpp -o test_mnist_nop
./test_mnist_nop
# Reading Data ...
# tSNE Start ...
# Computing P-values ...
# Time Spent: 3989ms
# Start Gradient Descent ...
# Iter: 100, err: 8.54428, time: 42s
# Iter: 200, err: 0.361679, time: 38s
# Iter: 300, err: 0.358529, time: 37s
# Iter: 400, err: 0.357205, time: 37s
# Iter: 500, err: 0.35648, time: 37s
# Iter: 600, err: 0.356172, time: 37s
# Iter: 700, err: 0.355964, time: 37s
# Iter: 800, err: 0.355953, time: 37s
# Iter: 900, err: 0.355953, time: 37s
# Iter: 1000, err: 0.355953, time: 37s
```

(b) No parallel

Figure 2. Performance comparison on SCC

3.2. Parallelization implementation

In this project, we use OpenMP as parallelization tool to accelerate t-SNE algorithm. For the independent data in gradient descent we can use OpenMP to do parallel computation.

3.3. Results

3.3.1 Visualization

In figure 1, subfigure (a) is visualization result of Iris data set after doing t-SNE algorithm. We can see in subfigure (b) the data set has three classes, ans t-SNE algorithm can better separate data points of different categories. Subfigure (b) is raw data of MNIST data set, and subfigure (c) is results after t-SNE. We can see that the raw data are clustered in a small range; after t-SNE, the data point of the same class is still gathered together, but the data point of different classes are gradually scattered.

3.3.2 Performance comparison

Figure 2 is a result of performance comparison between parallel t-SNE algorithm and normal t-SNE algorithm using MNIST data set on the SCC. We can see that the running time of parallel t-SNE for every 100 iterations is about 14 seconds; on the other hand, the running time of normal t-SNE for every 100 iterations is about 38 seconds which is more than double that of the parallel t-SNE algorithm.

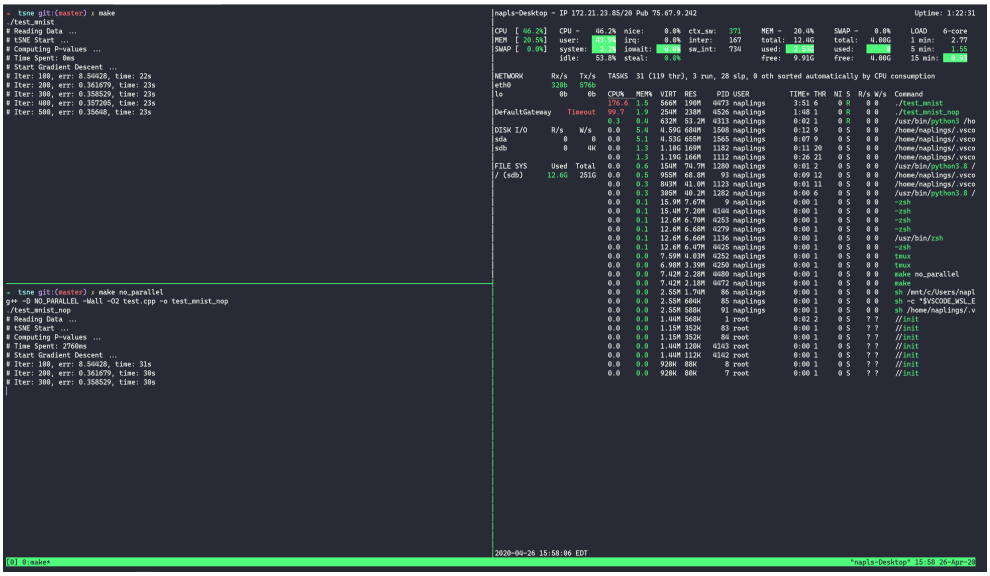


Figure 3. Performance comparison locally

4. Conclusion

In this project, we implement t-SNE algorithm to do dimensionality reduction and visualization, and we also implement parallel t-SNE using OpenMP. By comparing the visualization results with the raw data of data set, we find that t-SNE is much better than SNE because t-SNE solves the ”crowding problem”. From performance comparison part, we find that parallel method accelerate the program by accelerating the summation of the independent data in many functions in our code. The running time is reduced by 50% on the SCC and about 30% locally.

Through the entire project, we have learned more about the principle and specific implementation of the dimensionality reduction method of t-SNE. And by using parallel software to accelerate the algorithm, we have a further understanding of parallel software and methods.

References

[1] ReferenceLaurens van der Maaten and Geoffrey Hinton. “Visualizingdata using t-SNE”. In:Journal of machine learning research9.Nov (2008), pp. 2579–2605.