Parallel t-Distributed Stochastic Neighbor Embedding

Yuhao Zhou & Fengxu Tu

EC 526

Spring 2020

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Introduction

t-Distributed Stochastic Neighbor Embedding (t-SNE) [1] is a dimensionality reduction algorithm which is used to visualize high-dimensional data.

Method

- Calculate the pairwise similarity of high-dimensional data.
- Construct a low-dimensional pairwise similarity map.
- Minimize Kullback-Leibler divergence

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Measure of Similarity

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

where σ_i is the variance of the Gaussian that is centered on data point x_i, and set $p_{i|i} = 0$.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Low-dimensional Similarity

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

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Cost function

Kullback-Leibler divergence

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

Gradient of cost function:

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

So we use gradient descent with momentum to optimize the cost function.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

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

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Parallelization

We use OpenMP to do the parallelization in our implementation.

- **Pairwise Similarity Calculation**
  - No data dependency between data points
- **Gradient Calculation**
  - Cannot Parallelize Iterations, each iteration is based on the result of previous iteration
  - Parallelize the gradient calculation for each data point in one iteration

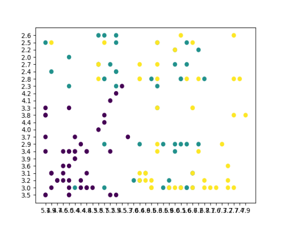
t-Distributed Stochastic Neighbor Embedding (t-SNE)

Implementation

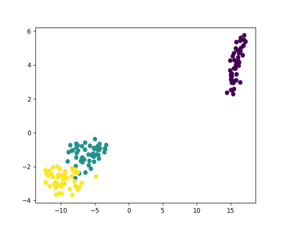
- **Dependencies**:
  - C++ Template Library: Eigen3
  - Visualization: Python3, NumPy, Matplotlib
- **Testing**:
  - Dataset: Iris, MNIST Subset
  - Options: With or without OpenMP
  - Test Envs: Local and SCC
  - Local Env: WSL2 Ubuntu 20.04 on 6-cores CPU

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Visualization



(a) Raw data

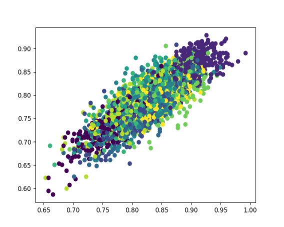


(b) After t-SNE

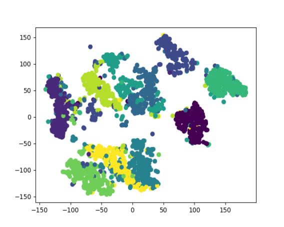
Figure: Iris Dataset $\in \mathbb{R}^{4\times 150}$, 3 classes

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Visualization



(a) Raw data



(b) After t-SNE

Figure: MNIST subset $\in \mathbb{R}^{784\times2500}$, 10 classes

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Performance

- Local Environment

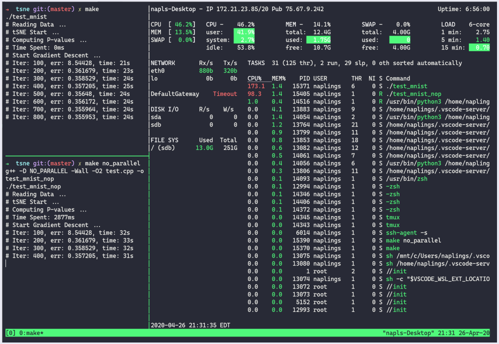


Figure: Performance comparison

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Performance

- SCC

```
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
```

(a) Single Thread

```
./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
```

(b) OpenMP

Figure: Performance comparison

Reference

Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.