Sitong Feng (sf3049)

Parallel t-SNE

Introduction

t-Distributed Stochastic Neighbor Embedding, or t-SNE, is a nonlinear dimensionality reduction algorithm, first designed to visualize high dimensional datasets. t-SNE archives better results than other available dimensionality reduction algorithms (e.g. PCA) but is very computationally heavy. I will explain in length on how parallel implementation will be suitable to optimize t-SNE.

Dive deep into t-SNE

t-SNE achieves its purpose in these steps:

1. Measure pairwise similarities between high-dimensional objects and embed the distance as probability densities, centered on a Gaussian distribution.

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2\right)}$$

The σi indicates the “perplexity”, or a preset number of neighboring points for which t-SNE aims to preserve distances, and usually ranges between 5 and 50. Then symmetrize the conditionals to get joint probability:

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

2. Map the high dimensional data points to lower dimensions(usually 2D or 3D) while preserving the local similarities. Use Student’s t-distribution instead of Gaussian distribution to avoid dense data in the tail.

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

3. Use gradient descent to minimize the Kullback–Leibler divergence, a cost function that measures the similarity between P and Q, corresponding to step 1 and 2.

$$C = KL(\mathbf{P} \;||\; \mathbf{Q}) = \sum_{ij} p_{ij}\log\frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4\sum_{j\neq i}\left(p_{ij} - q_{ij}\right)\left(\mathbf{y}_i - \mathbf{y}_j\right)\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}$$

Sitong Feng (sf3049)

Parallel Optimization

By analyzing the steps above, we can first see the pair-wise similarities ($O(N^2)$) of P can be computed in parallel, or the process for computing a probability distribution centered on every point can be parallel. Similarly, Q can also be computed in parallel. But since Q needs to be computed multiple times to optimize the cost function(in gradient descent), the parallel optimization can be considered on the pairs of one Q or on multiple Qs. Since the optimization with gradient descent depends on the context of the dataset, the more precise parallelization choice will be explored with varying datasets.

Some considerations involve experimenting with the parameters of the algorithms in parallel, but it is not a top priority. Other general optimizations include the Barnes-Hut approximation, which is a tree-based algorithm that reduces the pairwise computation to $O(N \log N)$.

Performance Evaluation and Work in Future

The optimization of the parallel implementation will be benchmarked with a sequential implementation of t-SNE in Haskell. They will also be performed on different samples of datasets for fair assessments. Then the algorithm will be treated with large datasets to prove its scalability with parallelism. The work in the future entails more fine-tuning of the algorithm with different parameters, developing a pipeline for analyzing textual data from preprocessing words with word embedding, t-SNE, to visualizing the results of parallel implementation for data exploration.

Reference

[1] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, 9, 2579–2605, Nov. 2008.

[2] L. van der Maaten, "Accelerating t-SNE using Tree-Based Algorithms," *Journal of Machine Learning Research,* 15, 3221-3245, Oct. 2014.