Swift Neighbor Embedding of Sparse Stochastic Graphs

Nikos Pitsianis[†‡], Dimitris Floros[†], Alexandros-Stavros Iliopoulos[‡], and Xiaobai Sun[‡]

[†]Department of Electrical and Computer Engineering,
Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece
[‡]Department of Computer Science, Duke University, Durham, NC 27708, USA

July 14, 2019

Contents

Summary

SG-t-SNE-Π is a high-performance software for swift embedding of a large, sparse, stochastic graph into a d-dimensional space ($d = 1, 2, 3$) on a shared-memory computer. Sparse graphs or network data emerge in numerous real-world applications and research fields, such as biological networks, commercial product networks, food webs, telecommunication networks, and word co-occurrence networks; see [1, 2] and references therein. Fundamental to many graph analysis tasks, graph embedding maps the vertices of a graph to a set of code/feature vectors in a code space. Often, a high-dimensional graph embedding is accompanied by a d-dimensional embedding ($d = 1, 2, 3$) for visual inspection, interpretation of analysis results, interactive exploration, and hypothesis generation.

SG-t-SNE-Π is based on the algorithm SG-t-SNE [3] and was used to obtain the published results of the latter. The algorithm is built upon precursors for k-nearest neighbor graph embedding, in particular Stochastic Neighbor Embedding (SNE) [4], t-Distributed Stochastic Neighbor Embedding (t-SNE) [5], and their variants [6, 7]. The precursor algorithms typically require the data to be provided in a metric space and the graph be regular with constant degree k. We remove this limitation and enable low-dimensional embedding of arbitrary large, sparse, stochastic graphs, which arise with real-world social and commercial networks. The use of the precursor methods was practically limited up to two-dimensional (2D) embeddings. We advocate 3D embedding and make it fast and practical on modern laptop and desktop computers, which are affordable and available to researchers and developers by and large.

The SG-t-SNE-Π library is implemented in `C++`. In addition, we provide two particular types of interfaces. The first is to support the conventional t-SNE, with its typical interface and wrappers [6], which converts data points in a metric space to a stochastic kNN graph. The second is a MATLAB interface for SG-t-SNE-Π.

1 Overview

We introduce SG-t-SNE-Π, a high-performance software for swift embedding of a large, sparse, stochastic graph $\mathcal{G} = (V, E, \mathbf{P}_c)$ into a d-dimensional space ($d = 1, 2, 3$) on a shared-memory computer. The algorithm SG-t-SNE and the software t-SNE-Π were first described in Reference [3]. The algorithm is built upon precursors for embedding a k-nearest neighbor (kNN) graph, which is distance-based and regular with constant degree k. In practice, the precursor algorithms are also limited up to 2D embedding or suffer from overly long latency in 3D embedding. SG-t-SNE removes the algorithmic restrictions and enables d-dimensional embedding of arbitrary stochastic graphs, including, but not restricted to, kNN graphs. SG-t-SNE-Π expedites the computation with high-performance functions and materializes 3D embedding in shorter time than 2D embedding with any precursor algorithm on modern laptop/desktop computers.

1.1 Precursor algorithms

The original t-SNE [5] has given rise to several variants. Two of the variants, t-SNE-BH [6] and FIt-SNE [7], are distinctive and representative in their approximation approaches to re-

ducing algorithmic complexity. They are, however, limited to kNN graph embedding. Specifically, at the user interface, a set of $n = |V|$ data points, $\mathcal{X} = \{\mathbf{x}_j\}_{j=1}^n$, is provided in terms of their feature vectors $\mathbf{x}_j$ in an L-dimensional vector space equipped with a metric/distance function. The input parameters include d for the embedding dimension, k for the number of near-neighbors, and u for the perplexity. A t-SNE algorithm maps the data points $\mathcal{X}$ to data points $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^n$ in a d-dimensional space.

There are two basic algorithmic stages in a conventional t-SNE algorithm. In the preprocessing stage, the kNN graph is generated from the feature vectors $\{\mathbf{x}_j\}$ according to the metric function and input parameter k. Each data point is associated with a graph vertex. Next, the kNN graph is cast into a stochastic one, $\mathbf{P}_{\rm c} = [p_{j|i}]$, and symmetrized to $\mathbf{P} = [p_{ij}]$,

$$p_{j|i}(\sigma_i) = \frac{a_{ij}\exp\bigl(-d_{ij}^2/2\sigma_i^2\bigr)}{\sum_l a_{il}\exp(-d_{il}^2/2\sigma_i^2)}, \quad \mathbf{P} = (\mathbf{P}_{\rm c} + \mathbf{P}_{\rm c}^{\rm T})/2. \tag{1}$$

where $\mathbf{A} = [a_{ij}]$ is the binary-valued adjacency matrix of the kNN graph, with zero diagonal elements (i.e., the graph has no self-loops), and d_{ij} is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. The Gaussian parameters σ_i are determined by the point-wise equations related to the same perplexity value u,

$$-\sum_j a_{ij} p_{j|i}(\sigma_i)\log(p_{j|i}(\sigma_i)) = \log(u). \tag{2}$$

The next stage is to determine and locate the embedding coordinates $\mathcal{Y}^* = \{\mathbf{y}_j^*\}_{j=1}^n$ by minimizing the Kullback-Leibler divergence

$$\mathcal{Y}^* = \arg\min_{\mathcal{Y}} \mathrm{KL}(\mathbf{P}\|\mathbf{Q}(\mathcal{Y})), \tag{3}$$

where matrix $\mathbf{Q}(\mathcal{Y}) = [q_{ij}]$ is made of the ensemble $\mathcal{Y}$ regulated by the Student t-distribution,

$$q_{ij} = \frac{1}{Z}\frac{1}{1+\|\mathbf{y}_i - \mathbf{y}_j\|^2}, \qquad Z = \sum_{k\neq\ell}\frac{1}{1+\|\mathbf{y}_k - \mathbf{y}_\ell\|^2}. \tag{4}$$

In other words, the objective of (3) is to find the optimal stochastic matching between $\mathbf{P}$ and $\mathbf{Q}$ defined, respectively, over the feature vector set $\mathcal{X}$ and the embedding coordinate set $\mathcal{Y}$. The optimal matching is obtained numerically by applying the gradient descent method. A main difference among the precursor algorithms lies in how the gradient of the objective function is computed.

1.2 Approximation of the gradient

The computation per iteration step is dominated by the calculation of the gradient. Van der Maaten reformulated the gradient into two terms [6]:

$$\frac{\partial(\mathrm{KL})}{\partial \mathbf{y}_i} = \frac{4}{Z}\underbrace{\sum_{i\neq j} p_{ij}q_{ij}(\mathbf{y}_i - \mathbf{y}_j)}_{\text{attraction term}} - \frac{4}{Z}\underbrace{\sum_{i\neq j} q_{ij}^2(\mathbf{y}_i - \mathbf{y}_j)}_{\text{repulsion term}}, \tag{5}$$

The attractive interaction term can be cast as the sum of $2d$ matrix-vector products with the sparse matrix $\mathbf{PQ} = [p_{ij}q_{ij}]$. The vectors are composed of the embedding coordinates, one in each dimension. The repulsive interaction term can be cast as the sum of $2d$ matrix-vector products with the dense matrix $\mathbf{QQ} = [q_{ij}q_{ij}]$. For clarity, we simply refer to the two terms as the **PQ** term and the **QQ** term, respectively.

The **QQ** (repulsion) term is in fact a broad-support, dense convolution with the Student t-distribution kernel on non-equispaced, scattered data points. As the matrix is dense, a naive method for calculating the term takes $O(n^2)$ arithmetic operations. The quadratic complexity limits the practical use of t-SNE to small graphs. Two types of existing approaches reduce the quadratic complexity to $O(n \log n)$, they are typified by t-SNE-BH and FIt-SNE. The algorithm t-SNE-BH, introduced by van der Maaten [6], is based on the Barnes-Hut algorithm. The broad-support convolution is factored into $O(\log n)$ convolutions of narrow support, at multiple spatial levels, each narrowly supported algorithm takes $O(n)$ operations. FIt-SNE, presented by Linderman et al. [7], may be viewed as based on non-uniform fast Fourier transforms. The execution time of each approximate algorithm becomes dominated by the **PQ** (attraction) term computation. The execution time also faces a steep rise from 2D to 3D embedding.

1.3 SG-t-SNE-Π

With the algorithm SG-t-SNE we extend the use of t-SNE to any sparse stochastic graph $\mathcal{G} = (V, E, \mathbf{P}_\mathrm{s})$. The key input is the stochastic matrix $\mathbf{P}_\mathrm{s} = [p_{j|i}]$, $\sum_j p_{j|i} = 1$, associated with the graph, where $p_{j|i}$ is not restricted to the form of (1). We introduce a parametrized, non-linear rescaling mechanism to explore the graph sparsity. We determine rescaling parameters γ_i by

$$\sum_j a_{ij}\, \phi\Big(p_{j|i}^{\gamma_i}\Big) = \lambda, \tag{6}$$

where $\lambda > 0$ is an input parameter and ϕ is a monotonically increasing function. We set $\phi(x) = x$ in the present version of SG-t-SNE-Π. Unlike (2), the rescaling mechanism (6) imposes no constraint on the graph, its solution exists unconditionally. For the conventional t-SNE as a special case, we set $\lambda = 1$ by default. One may still make use of and exploit the benefit of rescaling ($\lambda \neq 1$).

With the implementation SG-t-SNE-Π, we accelerate the entire gradient calculation of SG-t-SNE and enable practical 3D embedding of large sparse graphs on modern desktop and laptop computers. We accelerate the computation of both **QQ** and **PQ** terms by utilizing the matrix structures and the memory architecture in tandem.

1.3.1 Accelerated accumulation of attractive interactions

The matrix **PQ** in the attractive interaction term of (5) has the same sparsity pattern as matrix **P**, regardless of iterative changes in **Q**. Sparsity patterns are generally irregular. Matrix-vector products with irregular sparse matrix invoke irregular memory accesses and incur non-equal, prolonged access latencies on hierarchical memories. We moderate memory accesses by permuting the rows and columns of matrix **P** such that rows and columns with

similar nonzero patterns are placed closer together. The permuted matrix becomes block-sparse with denser blocks, resulting in better data locality in memory reads and writes.

The permuted matrix $\mathbf{P}$ is stored in the Compressed Sparse Blocks (CSB) storage format [8]. We utilize the CSB routines for accessing the matrix and calculating the matrix-vector products with the sparse matrix $\mathbf{PQ}$. The elements of the $\mathbf{PQ}$ matrix are formed on the fly during the calculation of the attractive interaction term.

1.3.2 Accelerated accumulation of repulsive interactions

We factor the convolution in the repulsive interaction term of (5) into three consecutive convolutional operations. We introduce an internal equispaced grid within the spatial domain of the embedding points at each iteration, similar to the approach used in FIt-SNE [7]. The three convolutional operations are:

`S2G`: Local translation of the scattered (embedding) points to their neighboring grid points.[1]

`G2G`: Convolution across the grid with the same t-distribution kernel function, which is symmetric, of broad support, and aperiodic.

`G2S`: Local translation of the gridded data to the scattered points.

The `G2S` operation is a gridded interpolation and `S2G` is its transpose; the arithmetic complexity is $O(w^d n)$, where w is the interpolation window size per side. Convolution on the grid takes $O(n_g \log(n_g))$ arithmetic operations, where n_g is the number of grid points, i.e., the grid size. The grid size is determined by the range of the embedding points at each iteration, with respect to the error tolerance set by default or specified by the user. In the current implementation, the local interpolation method employed by SG-t-SNE-Π is accurate up to cubic polynomials in d separable variables ($d = 1, 2, 3$).

Although the arithmetic complexity is substantially reduced in comparison to the quadratic complexity of the direct way, the factored operations suffer either from memory access latency or memory capacity issues, which were not recognized or resolved in existing t-SNE software. The scattered translation incurs high memory access latency. The aperiodic convolution on the grid suffers from excessive use of memory when the grid is periodically extended in all sides at once by zero padding. The exponential memory growth with d limits the embedding dimension or the graph size.

We resolve these memory latency and capacity issues in SG-t-SNE-Π. Prior to `S2G`, we relocate the scattered data points to the grid bins. This binning process has two immediate benefits. It improves data locality in the subsequent interpolation. It also establishes a data partition for parallel, multi-threaded execution of the scattered interpolation. We omit the parallelization details. For `G2G`, we implement aperiodic convolution by operator splitting, without using extra memory.

[1] The local interpolation employed by SG-t-SNE-Π is different from that used in FIt-SNE. The interpolation used in the latter is piecewise over non-overlapping grid cells.

1.3.3 Rapid intra-term and inter-term data relocations

In sparse or structured matrix computation of $O(n \log(n))$ arithmetic complexity, the execution time is dominated by memory accesses. We have described in sections 1.3.1 and 1.3.2 how we use intra-term permutations to improve data locality and reduce memory access latency in computing the attraction and the repulsion terms of (5). In addition, we permute and relocate in memory the embedding data points between the two terms, at every iteration step. The inter-term data relocation is carried out at multiple layers, exploiting block-wise memory hierarchy. The data permutation overhead is well paid-off by the much shortened time for arithmetic calculation with the permuted data. We use Π in the software name SG-t-SNE-Π to signify the importance and the role of the permutations in accelerating t-SNE algorithms, including the conventional one, and enabling 3D embeddings.

1.4 Supplementary material

Supplementary material and performance plots are found at http://t-sne-pi.cs.duke.edu.

References

[1] István A. Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, Michael A. Calderwood, Marc Vidal, and Albert-László Barabási. Network-based prediction of protein interactions. *Nature Communications*, 10(1):1240, 2019.

[2] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[3] Nikos Pitsianis, Alexandros-Stavros Iliopoulos, Dimitris Floros, and Xiaobai Sun. Spaceland embedding of sparse stochastic graphs. In *Proceedings of IEEE High Performance Extreme Computing Conference*, 2019. (to appear).

[4] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 857–864, 2003.

[5] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[6] Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(Oct):3221–3245, 2014.

[7] George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods*, 16(3):243–245, 2019.

[8] Aydin Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using

compressed sparse blocks. In *Proceedings of Annual Symposium on Parallelism in Algorithms and Architectures*, pages 233–244, 2009.

2 Getting started

2.1 System environment

SG-t-SNE-Π is developed for shared-memory computers with multi-threading, running Linux or macOS operating system. The source code is (to be) compiled by a `C++` compiler supporting Cilk. The current release is tested with the `GNU g++` compiler 7.4.0 and the `Intel icpc` compiler 19.0.4.233.

2.2 Prerequisites

SG-t-SNE-Π uses the following open-source software:

- `FFTW3` 3.3.8
- `METIS` 5.1.0
- `FLANN` 1.9.1
- `Intel TBB` 2019
- `Doxygen` 1.8.14

On `Ubuntu`:

```
sudo apt-get install libtbb-dev libflann-dev libmetis-dev libfftw3-dev doxygen
```

On `macOS`:

```
sudo port install flann tbb metis fftw-3
```

2.3 Installation

2.3.1 Basic instructions

To generate the SG-t-SNE-Π library, test and demo programs:

```
./configure
make all
```

To specify the `C++` compiler:

```
./configure CXX=<compiler-executable>
```

To test whether the installation is successful:

```
bin/test_modules
```

To generate the documentation:

```
make documentation
```

2.3.2 Support of the conventional t-SNE

SG-t-SNE-Π supports the conventional t-SNE algorithm, through a set of preprocessing functions. Issue

```
make tsnepi
```

to generate the `bin/tsnepi` binary, which is fully compatible with the existing wrappers provided by van der Maaten [6].

2.3.3 MATLAB interface

To compile the SG-t-SNE-Π `MATLAB` wrappers, use the `--enable-matlab` option in the `configure` command. The default `MATLAB` installation path is `/opt/local/matlab`; otherwise, set `MATLABROOT`:

```
./configure --enable-matlab MATLABROOT=<matlab-path>
```

2.4 Usage demo

We provide two data sets of modest size for demonstrating stochastic graph embedding with SG-t-SNE-Π:

```
tar -xvzf data/mobius-graph.tar.gz
bin/demo_stochastic_matrix mobius-graph.mtx

tar -xvzf data/pbmc-graph.tar.gz
bin/demo_stochastic_matrix pbmc-graph.mtx
```

The MNIST data set can be tested using existing wrappers provided by van der Maaten [6].

3 License and community guidelines

The SG-t-SNE-Π library is licensed under the GNU general public license v3.0. To contribute to SG-t-SNE-Π or report any problem, follow our contribution guidelines and code of conduct.

4 Contributors

Design and development:
Nikos Pitsianis[†‡], Dimitris Floros[†], Alexandros-Stavros Iliopoulos[‡], Xiaobai Sun[‡]
[†] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki, 54124 Greece
[‡] Department of Computer Science, Duke University, Durham, NC 27708 USA

5 Acknowledgements

Alpha test participants:
Xenofon Theodoridis