openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding

Pavlin G. Poličar[1,*], **Martin Stražar**[1], **and Blaž Zupan**[1,2]

[1]Faculty of Computer and Information Science, University of Ljubljana, SI-1000 Ljubljana, Slovenia,
[2]Department of Molecular and Human Genetics, Baylor College of Medicine, Houston TX 77030, U.S.A.

*To whom correspondence should be addressed.

Abstract

Summary: Point-based visualisations of large, multi-dimensional data from molecular biology can reveal meaningful clusters. One of the most popular techniques to construct such visualisations is t-distributed stochastic neighbor embedding (t-SNE), for which a number of extensions have recently been proposed to address issues of scalability and the quality of the resulting visualisations. We introduce openTSNE, a modular Python library that implements the core t-SNE algorithm and its extensions. The library is orders of magnitude faster than existing popular implementations, including those from scikit-learn. Unique to openTSNE is also the mapping of new data to existing embeddings, which can surprisingly assist in solving batch effects.
Availability: openTSNE is available at https://github.com/pavlin-policar/openTSNE.
Contact: pavlin.policar@fri.uni-lj.si, blaz.zupan@fri.uni-lj.si

The abundance of high-dimensional data sets in molecular biology calls for techniques for dimensionality reduction, and in particular for methods that can help in the construction of data visualizations. Popular approaches for dimensionality reduction include principal component analysis, multidimensional scaling, and uniform manifold approximation and projections (1). Among these, t-distributed stochastic neighbor embedding (t-SNE) (2) lately received much attention as it can address high volumes of data and reveal meaningful clustering structure. Most of the recent reports on single-cell gene expression data start with an overview of the cell landscape, where t-SNE embeds high-dimensional expression profiles into a two-dimensional space (3, 4). Fig. 1.a presents an example of one such embedding.

Despite its utility, t-SNE has been criticized for poor scalability when addressing large data sets, lack of global organization *t-SNE focuses on local clusters that are arbitrarily scattered in the low-dimensional space* and absence of theoretically-founded implementations to map new data into existing embeddings (5, 6). Most of these shortcomings, have recently been addressed. Linderman et al. (7) sped-up the method through an interpolation-based approximation, reducing the time complexity to be merely linear dependent on the number of samples. Kobak and Berens (8) proposed several techniques to improve global positioning, including estimating similarities with a mixture of Gaussian kernels. While no current popular software library supports mapping of new data into reference embedding, van der Maaten (9) proposed a related approach using neural networks.

We introduce `openTSNE` , a comprehensive Python library that implements t-SNE and all its recently proposed extensions. The library is compatible with the Python data science ecosystem (e.g., `numpy`, `sklearn`, `scanpy`). Its modular design encourages extendibility and experimentation with various settings and changes in the analysis pipeline. For example, the following code uses multiscale similarity kernels to construct the embedding from Fig. 1.b.

```
adata =
    anndata.read_h5ad("macosko_2015.h5ad")
affinities = openTSNE.affinity.Multiscale(
  adata.obsm["pca"], perplexities=[50,
      500], metric="cosine")
init = openTSNE.initialization.pca(
  adata.obsm["pca"])
embedding = TSNEEmbedding(init, affinities)
embedding.optimize(n_iter=250,
    exaggeration=12, momentum=0.5,
    inplace=True)
embedding.optimize(n_iter=750,
    momentum=0.8, inplace=True)
```

Here, we first read the data, define the affinity model based on two Gaussian kernels with varying perplexity, use a PCA-based initialization, and run the typical two-stage t-SNE optimization. Notice that the code for the standard t-SNE used for Fig. 1.a is similar but uses only a single kernel (`perplexities=[30]`).

The proposed `openTSNE` library is currently the only Python t-SNE implementation that supports adding new samples into constructed embedding. For example, we can reuse the `embedding` created above to map new data into existing embedding space in the following code,

```
data =
    anndata.read_h5ad("shekhar_2016.h5ad")
adata, data = find_shared_genes(adata, data)
genes = select_genes(adata.X, n=1000)
embedding.affinities =
    affinity.PerplexityBasedNN(
  adata[:, genes].X, perplexity=30,
      metric="cosine")
new_embedding = embedding.transform(data[:,
    genes].X)
```

which loads and prepare the new data, defines the affinity model, and computes the embeddings. `openTSNE` embeds new data points independently of one another, without changing the reference embedding. The example of combining two data sets with mouse retina cells is shown on Fig. 1.c,

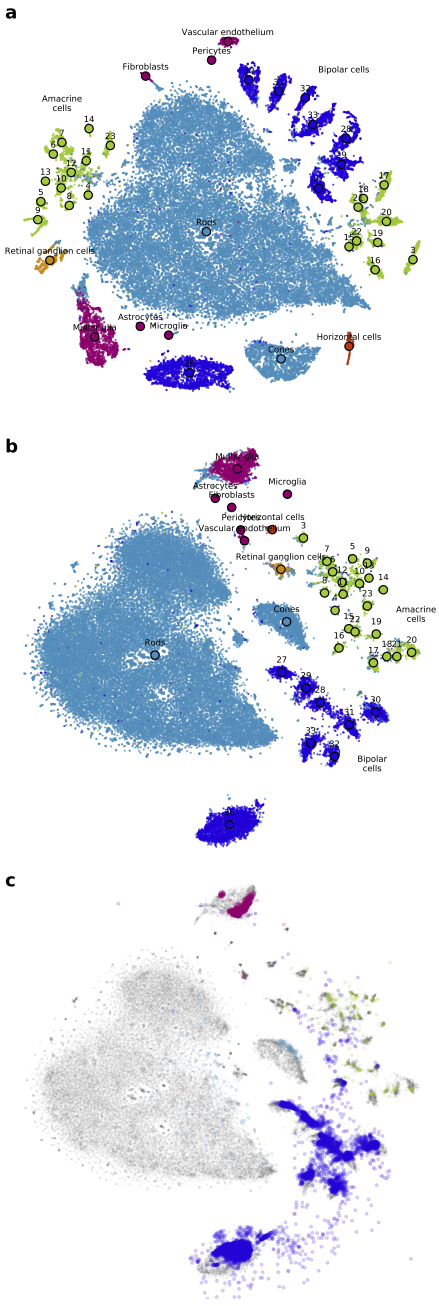

Fig. 1. Example application of `openTSNE` on mouse retina cells from Macosko et al. (3) and Shekhar et al. (4). a) Standard t-SNE embedding using random initialization and perplexity of 30. b) t-SNE embedding using multi-scale affinities leads to better global cluster organization. Cluster annotations in both a) and b) are from Macosko *et al.* c) Embedding of independent mouse retinal cells from Shekhar *et al.* (points in color) to a reference t-SNE visualization of data from Macosko *et al.* (points in gray) places cells in the clusters that match classifications from original publications and alleviates batch effects.

where the cells from the secondary data set are matched to the cells in the reference embedding. This procedure can therefore be used to handle batch effects (10), a key obstacle in molecular biology when dealing with the data from different sources (11). The code used to plot the embeddings is not shown for brevity, but is, together with other examples, available on `openTSNE` 's GitHub page.

Our Python implementation introduces computational overhead: `openTSNE` is about 25% slower than `FIt-SNE` (7), a recent t-SNE implementation in C++. However, `openTSNE` is still orders of magnitude faster than other Python implementations, including those from `scikit-learn` and `MulticoreTSNE` (see Benchmarks in `openTSNE` documentation on GitHub). An example data set with 200,000 cells is processed in more than 90 minutes with scikit-learn and in less than 4 minutes with `openTSNE` . The framework includes controlled execution (callback-based progress monitoring and control), making it suitable for interactive data exploration environments such as Orange (12). A pure Python implementation offers distinct advantages that include integration with Python's rich data science infrastructure and ease of installation through PyPI and *conda*.

Funding and Acknowledgements

The support for this work was provided by the Slovenian Research Agency Program Grant P2-0209, and by European Regional Development Fund and the Slovenian Ministry of Education through BioPharm project. We want to thank George Linderman and Dmitry Kobak for helpful discussions on extensions of t-SNE.

Bibliography

1. Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.
2. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9:2579–2605, 2008.
3. E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, J. J. Trombetta, D. A. Weitz, J. R. Sanes, A. K. Shalek, A. Regev, and S. A. McCarroll. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5), 2015.
4. K. Shekhar, S. W. Lapan, I. E. Whitney, N. M. Tran, E. Z. Macosko, M. Kowalczyk, X. Adiconis, J. Z. Levin, J. Nemesh, M. Goldman, S. A. McCarroll, C. L. Cepko, A. Regev, and J. R. Sanes. Comprehensive Classification of Retinal Bipolar Neurons by Single-Cell Transcriptomics. *Cell*, 166(5):1308–1323, 2016.
5. Jiarui Ding, Anne Condon, and Sohrab P Shah. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature Communications*, 9(1): 2002, 2018.
6. Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using umap. *Nature Biotechnology*, 37(1):38, 2019.
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, 2019.
8. Dmitry Kobak and Philipp Berens. The art of using t-SNE for single-cell transcriptomics. *bioRxiv*, page 453449, 2018.
9. Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391, 2009.
10. Pavlin G Poličar, Martin Stražar, and Blaž Zupan. Embedding to reference t-SNE space addresses batch effects in single-cell classification. *BioRxiv*, page 671404, 2019.
11. Jeffrey T Leek, Robert B Scharpf, Héctor Corrada Bravo, David Simcha, Benjamin Langmead, W Evan Johnson, Donald Geman, Keith Baggerly, and Rafael A Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10), 2010.
12. Martin Stražar, Lan Žagar, Jaka Kokošar, Vesna Tanko, Aleš Erjavec, Pavlin G Poličar, Anže Starič, Janez Demšar, Gad Shaulsky, Vilas Menon, et al. scorange—a tool for hands-on training of concepts from single-cell data analytics. *Bioinformatics*, 35(14):i4–i12, 2019.