SCALABLE VISUALIZATION FOR HIGH-DIMENSIONAL SINGLE-CELL DATA

JUHO KIM

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
Urbana, Illinois, 61801, USA
Email: juhokim2@illinois.edu

NATE RUSSELL

Institute of Genomic Biology, University of Illinois at Urbana-Champaign
Urbana, Illinois, 61801, USA
Email: ntrusse2@illinois.edu

JIAN PENG

Department of Computer Science, University of Illinois at Urbana-Champaign
Urbana, Illinois, 61801, USA
Email: jianpeng@illinois.edu

Single-cell analysis can uncover the mysteries in the state of individual cells and enable us to construct new models about the analysis of heterogeneous tissues. State-of-the-art technologies for single-cell analysis have been developed to measure the properties of single-cells and detect hidden information. They are able to provide the measurements of dozens of features simultaneously in each cell. However, due to the high-dimensionality, heterogeneous complexity and sheer enormity of single-cell data, its interpretation is challenging. Thus, new methods to overcome high-dimensionality are necessary. Here, we present a computational tool that allows efficient visualization of high-dimensional single-cell data onto a low-dimensional (2D or 3D) space while preserving the similarity structure between single-cells. We first construct a network that can represent the similarity structure between the high-dimensional representations of single-cells, and then, embed this network into a low-dimensional space through an efficient online optimization method based on the idea of negative sampling. Using this approach, we can preserve the high-dimensional structure of single-cell data in an embedded low-dimensional space that facilitates visual analyses of the data.

1. Introduction

Many traditional biological experiments have been conducted on bulk-cell populations[1] with an assumption that cells in the same group share homogeneous properties. However, some evidence[1-3] shows that heterogeneity can exist even within a small group of cells. The assumption based on homogeneity of each cell group can mislead averages and does not properly explain small but critical changes in individual cells. Each cell can have different biological properties such as cell sizes, gene expression levels, RNA transcripts, and bio marker expressions. These variations can be very important to answer previously unsolved questions in stem cell research, cancer biology, and immunology. Single-cell data analysis has contributed to understand the various and important behaviors of individual cells[1-15].

The recent development of single-cell technologies has also improved the analysis to be more reliable and reasonable. For example, mass cytometry[4,16] can measure up to 60 parameters at the same time for tens of thousands of individual cells. In addition, single-cell RNA sequencing techniques[17,18] also have been widely used, which deal with hundreds of or thousands of parameters per cell.

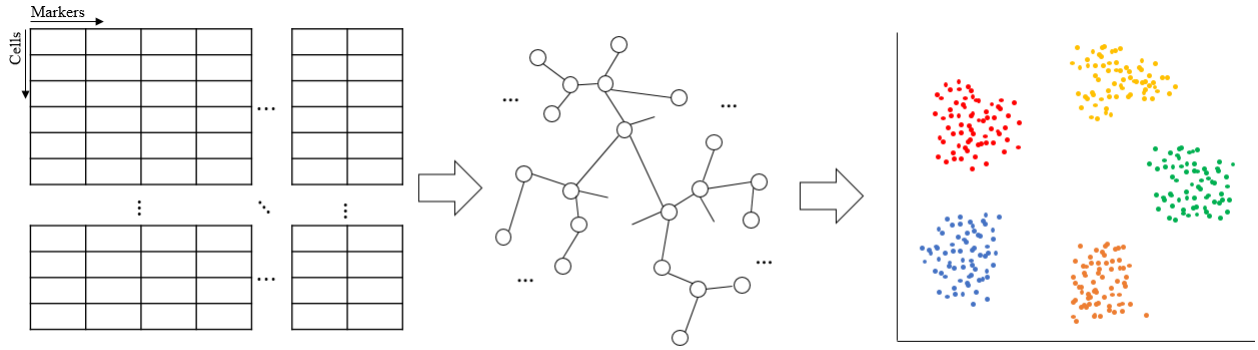
Even though the advanced single-cell technologies can provide quality data, such data sets are still difficult to analyze. Traditionally, single-cell data are analyzed in a biaxial scatter plot for two variables at once[19]. However, this method requires the order of dimension squared to represent all pairwise relationships between variables, which is computationally expensive. In addition, scatter plots cannot capture multivariate relationships between more than two variables. Thus, new computational methods have been developed for analyzing single-cell data. For instance, SPADE[6] tries to find hierarchies of high-dimensional single-cell data showing cellular heterogeneity by clustering of down-sampled cytometry data, constructing minimum spanning trees, and up-sampling. However, this method considers not each cell itself but cell groups and their behaviors on average. X-shift[12] is recently developed to discover cell subsets and visualize them based on a weighted k-nearest neighbor density estimation.

Another approach to deal with the high-dimensionality of single-cell data is to use dimensionality reduction techniques. Some researchers applied principle component analysis (PCA)[20] to find low-dimensional projections of single-cell data[21,22]. Although PCA is possibly the most popular method of dimensionality reduction, it is a linear projection method. Thus, it cannot capture nonlinear structures in single-cell data. In order to address this issue, advanced methods based on nonlinear dimensionality reduction have been developed. Both viSNE[8] and ACCENSE[10] are based on an algorithm called t-Distributed Stochastic Neighbor Embedding (t-SNE)[23]. viSNE applies t-SNE to mass cytometry data and reveals biologically meaningful relationships from bone marrow and leukemia data. ACCENSE combines the results of t-SNE with kernel-based density estimation and finds subpopulations of given single-cell data sets. However, the runtime complexity of t-SNE is $O(n^2)$, and that of its accelerated version, Barnes-Hut-SNE[24] is $O(n \log n)$ where n is the number of cells. Thus, both methods require excessive computational time for large-scale single-cell data sets with hundreds of thousands or millions of cells.

In this paper, we propose a scalable embedding-based visualization method for large-scale and high-dimensional single-cell data based on a new graph embedding algorithm, LargeVis[25]. The proposed method constructs a k-nearest neighbor (k-NN) network to find the structure of similarities between high-dimensional single-cell data. This process is accelerated by an approximate k-NN construction method based on random projection trees[26] and neighbor exploring[30]. This approach optimizes a probabilistic utility function to embed the high-dimensional single-cell data into a low-dimensional space (2D or 3D). For efficient training, the utility function is approximated using negative sampling[28] that was introduced in word2vec[28]. The runtime complexity of our method is linear with regard to the number of cells, which is faster than previous single-cell visualization tools such as viSNE[8] and ACCENSE[10].

2. Methods

We propose a new approach for visualizing high-dimensional single-cell data via efficient dimensionality reduction based on LargeVis[25]. The algorithm consists of two steps: constructing an approximate k-NN network to find the similarity structure between high-dimensional single-cell data and embedding the constructed network into a 2D or 3D space while preserving the high-dimensional structure in an easily visualized low-dimensional space. Pairwise similarity between single-cell data points is determined by the distance between them in their marker expression representation space. The core assumption is that numerical proximity in the marker space is proportional to cell similarity.



(a) High-dimensional single-cell data (b) Construction of k-nearest neighbor network (c) Two-dimensional representation

Figure 1. Outline of high-dimensional single-cell data visualization: constructing a k-nearest neighbor network and embedding the network into a 2D space.

2.1. *Notation*

We denote a set of high-dimensional single-cell data as $\mathcal{X} = \{x_i | x_i \in \mathbb{R}^p,\ i \in [n],\ p > 3\}$, where p is the dimension of measurements and n is the number of cells in the data; and the embedded representations of cells are denoted as $\mathcal{Y} = \{y_i | y_i \in \mathbb{R}^2\ or\ \mathbb{R}^3,\ i \in [n]\}$ in a low-dimensional space.

2.2. *Construction of a k-nearest neighbor network*

Constructing a k-nearest neighbor (k-NN) network is a very crucial step in many applications of machine learning such as a distance-based similarity search, manifold learning, and topological data analysis. Finding the exact k-NN network for large-scale single-cell data is time-consuming because it requires $O(n^2)$ time to compute all pairwise distances between all cells in the data set. Approximate methods for constructing a k-NN network have been developed, all of which have a tradeoff between speed and accuracy. Common approaches include locality sensitivity hashing[29], neighbor exploring methods[27], and partitioning methods based on random projection trees[26], k-d trees[31] and k-means trees[31].

As suggested by LargeVis[25], we develop a fast method to construct an approximate k-NN network. We first partition the whole high-dimensional space into two subspaces and generate a tree having only a root node. A set of single-cells in each partitioned subspace belongs to child nodes of the root node. Then, for the two subspaces that each set of single-cells in the child nodes belongs to, we partition each subspace into two sub-subspaces and generate two child nodes for each child node of the root node. The single-cells in each sub-subspace are assigned to each generated child nodes' child node. By continuing to partition the space iteratively, we can build a tree that assigns a group of single-cells belonging to partitioned small subspaces to its nodes. When the number of cells in a certain node is equal to or less than a predefined threshold, we stop the iterations. The single-cells in each leaf node are considered to be a candidate of approximate nearest neighbors. The generated tree is called a random projection tree.

By generating many random projection trees, we can increase the accuracy of the construction of a k-NN network, but it is time consuming. Instead of building many random projection trees, we use a neighbor search method in order to enhance both the accuracy and the efficiency. Specifically, we search the neighbor j of the neighbor of each node i assuming that its neighbor's neighbor is likely to be its neighbor also[30]. If the number of neighbors of node i is less than k, the method pushes some searched neighbor's neighbor j into the set of nearest neighbors of the node i. By iteratively doing this procedure, we can improve the accuracy of the construction and finally find our approximate k-NN network. Regarding the accuracy of the k-NN network construction, one can refer to the paper of largeVis[25], which dealt with several benchmark tests for the accuracy. The k-NN network construction process has linear time complexity because we build only a few random projection trees and because searching a certain node's neighbor's neighbor requires just a few iterations.

We then calculate the weight of each pairwise edge that represents the similarity structure of the constructed network using the Gaussian kernel, which was also used by t-SNE[23,24]. The conditional probability that the edge from data x_i to x_j is observed is first computed by:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{(i,k)\in E} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$
$$p_{i|i} = 0 \qquad (1)$$

where the parameter σ_i is determined by setting the perplexity, and E is the set of all edges in the k-NN network. To make the network symmetric, the weights are defined as:

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

where n is the number of input single-cell data. Since the number kn is much smaller than the number of all pairs (n^2), the constructed k-NN network is sparse. The sparsity of the k-NN network can make us compute w_{ij} within linear time complexity. Through the steps, our method can find the similarity structure of high-dimensional single-cell data within linear time complexity $O(kn)$.

2.3. *Network embedding into a low-dimensional space*

Embedding the constructed k-NN network is intended to preserve local and global network topology such that neighbors in the network are near each other in a low-dimensional space. First, for two nodes v_i and v_j, LargeVis[25] defines the probability that they come from the same neighborhood, i.e. the probability that we can observe the edge between two nodes in the k-NN network, as:

$$p(e_{ij} = 1|y_i, y_j) = f(\text{dist}(y_i, y_j)) \tag{3}$$

where f is a transformation function to map the distance between y_i and y_j into a probability value.

The function f satisfies the idea that when the distance between two low-dimensional points is small, the probability observing the connection between them is high. After considering some candidates like a multinomial logistic model and a sigmoid function, we chose $f(x) = \frac{1}{1+\alpha x^2}$ ($\alpha > 0$) due to its computational simplicity. The selected function f does not require any normalization across the data set, thus only $O(n)$ runtime is needed for objective evaluation and gradient calculation in the embedding optimization (see below). In addition, we can control the thickness of the tail of the function f by controlling α. When α becomes smaller, its tail gets thicker. When $\alpha = 1$, f is Student's t-distribution with degree of freedom one except a scaling factor $\frac{1}{\pi}$. On the other hand, t-SNE[23] uses the Gaussian kernel p_{ij} of (1) and a t-distributed kernel $q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$ to measure its high-dimensional and low-dimensional similarity, respectively. By minimizing the Kullback-Leibler divergence between two similarities through gradient descent, t-SNE finds its low-dimensional embedding. The gradient of its cost function contains the normalization term of q_{ij}. Computing the term requires $O(n^2)$. To avoid inefficiency, accelerated t-SNE[24] uses Barnes-Hut algorithm[32] and reduces its time complexity from $O(n^2)$ to $O(n \log n)$. Two versions of t-SNE are more expensive than our approach.

Like LargeVis[25], we chose Euclidean distance as a distance metric in a low-dimensional space because computing Euclidean distance between embedded single-cell data is simple. In addition, we can map each calculated distance to one of the various probability function values since the range of Euclidean distance is $[0, \infty)$.

To embed the high-dimensional data, we define a log likelihood utility function (4) that considers both the probabilities of all edge connections E of the constructed k-NN network and the probabilities of all negative edges E^C. Negative edges mean that pairwise single-cell connections that are not observed in the k-NN network. This idea originally comes from noise-contrastive estimation (NCE)[33], which considers estimation that differentiates its observed data from noise using nonlinear logistic regression. Using the idea of NCE, we want to discriminate the same type of cells

from different types of cells. Specifically, by maximizing the first term of (4), we can make similar single-cells become closer to each other in a low-dimensional space, and by maximizing the second part of (4), we can make dissimilar single-cells move away from each other.

$$J = \sum_{(i,j)\in E} w_{ij} \log p\left(e_{ij} = 1 | y_i, y_j\right) + \sum_{(i,j)\in E^C} \gamma \log(1 - p\left(e_{ij} = 1 | y_i, y_j\right)) \quad (4)$$

However, considering all negative edges is computationally expensive or even intractable when input data are very large. Thus, instead of using all negative edges, we use the idea of negative sampling[28]. This approach considers only a few samples drawn from a noise distribution. We assumed $P_n(j) \sim d_j^{3/4}$ as the noisy distribution where $d_j$is the degree of node j, which was used in word2vec[28]. By letting M the number of negative samples, we can redefine the utility function as:

$$J = \sum_{(i,j)\in E} w_{ij} \log p\left(e_{ij} = 1 | y_i, y_j\right) + \sum_{k=1}^{M} \mathbb{E}_{j_k \sim P_n(j)} \gamma \log(1 - p\left(e_{ij_k} = 1 | y_i, y_{j_k}\right)) \quad (5)$$

Then, we optimized (5) by applying asynchronous stochastic gradient descent (ASGD)[34]. It is a powerful optimization technique which can be efficiently parallelized and can make our algorithm more scalable. ASGD can be used in this context because the network constructed by the first step is sparse and there are few memory access conflicts between the threads we used. The learning rate is determined by $\rho_t = \rho(1 - t/T)$ where T is the total number of edge samples[25], and the initial learning rate ρ_0 is determined by considering the properties of input single-cell data. The time complexity of each SGD step of (5) is $O(M)$. For a large number of data set, the number of SGD iterations is usually proportional to the number of the given data set, n. Thus, the time complexity of the optimization is $O(Mn)$, which is linear with respect to the number of samples.

3. Experiments and Discussion

3.1. *Data and data processing*

We used mass cytometry data that are provided by X-shift[12]. They consist of 10 data sets that contain mice bone marrow samples stained with surface markers, and each of them has 51 parameters. Instead of using all of them, we used 39 surface marker expressions[12,35] that were utilized for mass cytometry experiments of the immune system reference framework[35]. In addition, the data was processed through noise thresholding and asinh transformation, i.e. $y = \text{asinh}(\max(x - 1, 0)/5)$ like X-shift[12] and viSNE[8] applied. The data sets also offer 24 gating annotations of each cell, which were used to distinguish cells in visualization and compare the clustering performance of viSNE and our method.

3.2. *Experimental setting*

We compared our method with viSNE[8] because it is a state-of-the-art method of single-cell visualization based on nonlinear embedding like our approach. Before implementing both algorithms, we set the parameters of each method. viSNE is based on Barnes-Hut-SNE[24], which has two parameters: perplexity and theta that controls the tradeoff between speed and accuracy. In our

experiments, we set the two as 30 and 0.5, respectively. Our method allows for more control and therefore has more parameters: number of trees, number of neighbors, perplexity, number of negative samples, rho, gamma, and alpha. We set the parameters considering our input data set. The first three parameters are related to constructing a k-NN network. The number of trees and neighbors can determine the shapes of a k-NN network, and perplexity is related to computing edge weights of section 2.2. The other parameters are related to network embedding. The number of negative samples is M of (5), rho is the initial learning rate, gamma is the weight of negative edges, and alpha determines the thickness of the tail of f. Table 1 shows the parameters we tuned for our visualization.

Table 1. Parameters for constructing a k-NN network and for network embedding

Parameters for constructing a k-NN network		
Number of trees 20 – 100	Number of neighbors 20 – 150	Perplexity 10 – 50
Parameters for network embedding		
Number of negative samples 5 – 10	Rho 1 – 10	Gamma 1 – 10	Alpha < 1

All experiments for measuring the computation time were performed on a machine with Intel Xeon E5-2650 CPUs running at 2.30GHz. 40 threads were used except the experiments about the effectiveness of multiple threads.

3.3. *Results*

3.3.1. *Visualization*

Figure 2 represents the visualization for mice bone marrow replicate 7 data set[12]. Overall, the same type of cells forms a dense subset. The number of a certain class of cells such as HSC in the data set was so small that they were difficult to distinguish from other cells and to find in our visualization. Except these cells, we can see clearly that the same type of cells gathers together and different types of cells move away from each other in a two-dimensional space. In addition, we can find some similar cells to stay together in Figure 2. For example, similar cell types like Intermediate Monocytes (red) and Classical Monocytes (yellow) appear close to each other. Two types of B cells (purple and light green) are also stay near each other.

In addition, we applied viSNE to the same data set. viSNE also represented cell subpopulations very well. The same type of cells was grouped together, and it can clearly distinguish different types of cells. In the experiments, our method tended to form denser and rounder clusters than viSNE but to have more randomly scattered samples. Due to the space limit, the visualization results of viSNE are shown through our web-based visualization tool (see section 4). We also compare our method with other embedding methods such as PCA in the tool.

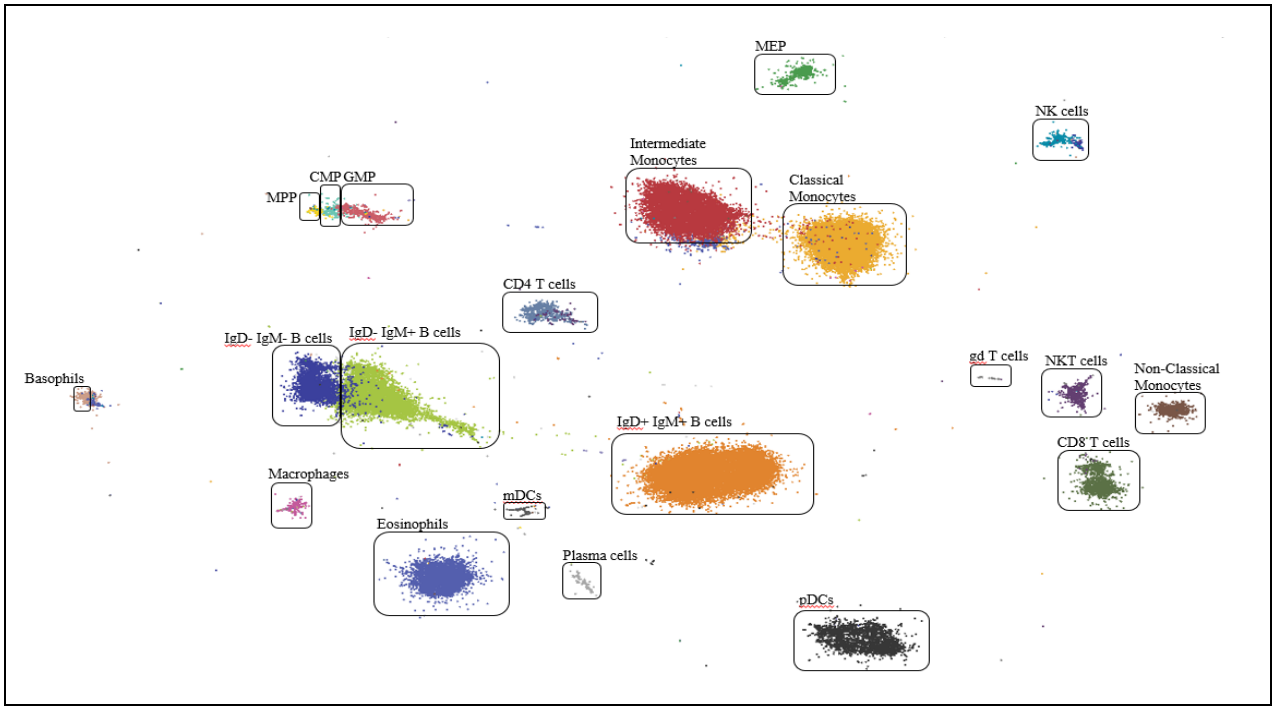


Figure 2. Visualization of our method for bone marrow replicate 7 data set.

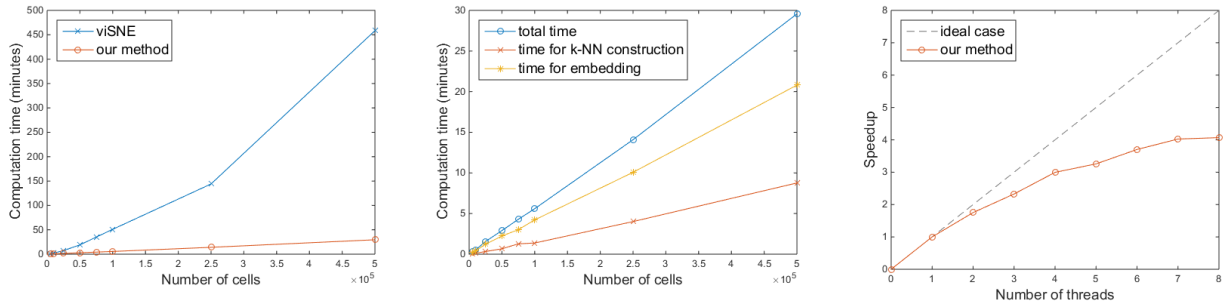
3.3.2. *Computation time*

One of the main goals of our method is to make visualization of high-dimensional single-cell data be faster and more scalable. Thus, we compared the computation time between viSNE[8] and our method for various cases. In addition, to test the scalability and parallelizability, we measured the effectiveness of speedup with respect to the number of threads.

To measure the computation time and evaluate the scalability with respect to the size of the data set, we constructed 8 single-cell data sets that contained 5,000, 10,000, 25,000, 50,000, 75,000, 100,000, 250,000, and 500,000 data, respectively. For each data set, cells were uniformly sampled from the union of 10 data sets (total number: 841,644). Each data set contained 39 parameters and were preprocessed by noise thresholding and asinh transformation before sampling. Figure 3(a) shows that our method was faster than viSNE for all 8 sampled data sets and our method is easier to make scalable. The total computation time of our method consists of two computation times: one is for constructing a k-NN network and the other is for embedding the network. Figure 3(b) shows how much time we needed for each step.

In addition, we tested the parallelization of our method in the multi-core setting. Since our method uses asynchronous stochastic gradient descent (ASGD)[34] for training, it can be more accelerated by using multiple threads. We measured the computation time of our method when dealing with the union of all 10 single cell data sets with respect to the number of threads. By increasing the number of threads from 1 to 8, we measured the effectiveness of the multiple threads for our method. When we used 8 threads simultaneously, the speedup rate was 4.1 times faster than single-thread implementation in Figure 3(c). The results show that our method can be easily

parallelized and can be made more scalable through a multi-core system.



(a) (b) (c)

Figure 3. (a) Comparison of the computation time of viSNE and our method with respect to the number of single-cell data samples. (b) Separate analysis of the computation time for constructing a k-NN network and for embedding with regard to the number of single-cell data samples. (c) Effectiveness of the multiple threads for speedup of our method.

3.3.3. *Clustering*

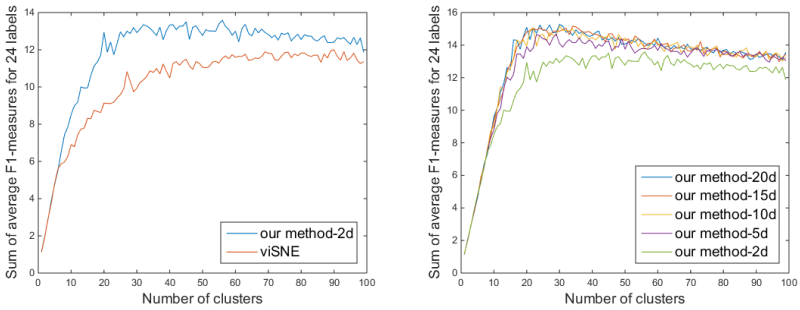
In this section, we compared the quality of embedding by comparing the performance of clustering. In our experiments, we first applied one of the off-the-shelf clustering algorithms, k-means clustering[20] to the embedded vectors by viSNE[8] and those by our method. Next, we measured the performance of clustering using hand-gated annotations of each cell. Specifically, we followed the process of X-shift[12] to compare the clustering result and hand-gated labels and to calculate F1-measures. As the number of clusters changed from 2 to 100, we computed F1-measures for each cluster that a label was assigned to by the Hungarian algorithm[36]. This process was applied to our 10 data sets, and we obtained an average F1-measure sum. As another performance measure, we obtained maximum F1-measures for each data set across all the number of clusters and took a median.

As the input of clustering, we used the two-dimensional vectors embedded by viSNE[8] and our method. We compared an average F1-measure sum of both methods and a median of maximum F1-measures. Figure 4(a) shows that the clustering performance of our method was better than that of viSNE across all the number of clusters with respect to an average F1-measure sum. In addition, we compared a median of maximum F1-measures of viSNE and our method. Our two-dimensional embedding obtained 14.68 while viSNE obtained 13.23 as its median. Our method also outperformed viSNE for this metric.

Since our method is developed mainly for visualization, two or three dimensional vectors are usually used as a result of embedding. However, the algorithm can embed high-dimensional single-cell data into another arbitrary low-dimensional space other than a two- or three-dimensional space. The vectors embedded in a higher-dimensional space than a space for visualization can lose less intrinsic information about original high-dimensional single-cell data. Thus, they can be used to enhance the performance of clustering. We clustered the data by using 5-, 10-, 15- and 20-dimensional representations obtained by our embedding.

Figure 4(b) shows that the performance of clustering was improved when we used the vectors

with higher dimensions than two. The performances as we used 10-, 15-, and 20-dimensional vectors are similar to each other and better than the performance as we used two- or 5-dimensional vectors.



(a) (b)

Figure 4. (a) Comparison of the clustering performance of viSNE and our method when using two-dimensional vectors with respect to the number of clusters. (b) Comparison of the clustering performances when we changed the dimension of our embedding.

4. Interactive Visualization

To better aide analysis, we also introduce an interactive web browser based visualization tool featured in Figure 5. It allows researchers to examine their own data quickly by enabling functionality like mouse-over, zoom, pan, brushing, and linking on the embedded data. Users can color data by quantities of marker values as well as qualitative gate information. One can select arbitrary groups of single-cell data points, tag them, and save them for downstream analysis. We provide code, documentation, and video demonstrations to reproduce experiments and apply our methods to new single-cell data through the link[a]. All code is made available under an MIT license.

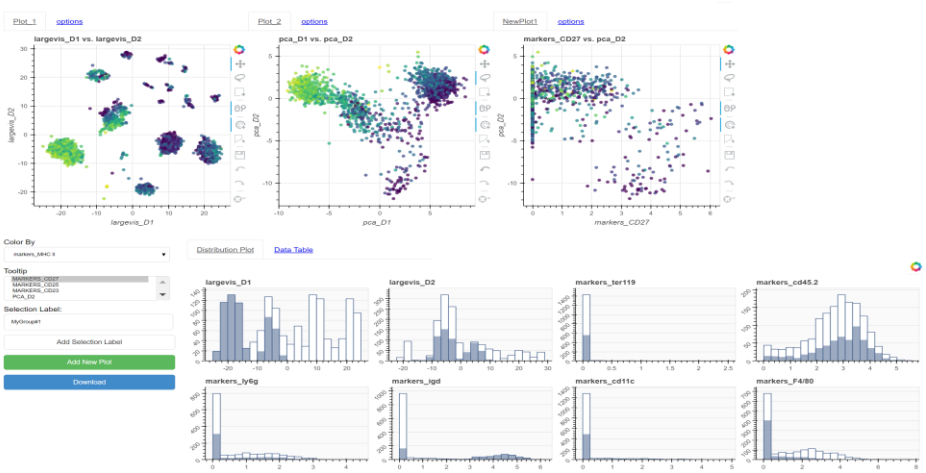


Figure 5. Screen shot of web browser based visualization developed in python. The left scatter plot depicts the result of our proposed method and on the middle, a PCA projection of the data. The right plot describes embedded expressions

[a] https://github.com/nate-russell/SVHD-Single-Cell

of a specific marker with respect to a certain projection. Color assignment and data selection labeling are also available through widgets at the bottom left. Some data statistics and the table to the right show all provided marker data and meta data regarding the single-cell data.

5. Conclusion

In this paper, we introduced a new visualization method for large-scale and high-dimensional single-cell data based on LargeVis[25], which consists of two parts: constructing an approximate k-NN network and embedding the constructed network into a low-dimensional space. Since the both steps have linear time complexity, our method is scalable and readily for analyzing large-scale single-cell data sets with hundreds of thousands or even millions of single cells. Specifically, our experiment results showed that the proposed method is much faster than viSNE[8], a state-of-the-art single-cell visualization method. In addition, through the experiments about clustering, we showed that the quality of our embedding is better than that of viSNE on cell identity mapping with respect to F1-measures. We also provide a web based interactive visualization tool and all necessary code and documentation to extend this approach to new data.

Acknowledgments

This study was supported by a Sloan Research Fellowship and a National Center for Supercomputing Applications (NCSA) Fellowship of University of Illinois at Urbana-Champaign.

References

1. O. Stegle, S. A. Teichmann, and J. C. Marioni, *Nat. Rev. Genet*. **16**, 133-145 (2015).
2. F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle, *Nat. Biotechnol*. **33**, 155-160 (2015).
3. C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokhare, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn, *Nat. Biotechnol*. **32**, 381-386 (2014).
4. O. Ornatsky, D. Bandura, V. Baranov, M. Nitz, M. A. Winnik, and S. Tanner, *J. Immunol. Methods*. **361**, 1-20 (2010).
5. S. C. Bendall, E. F. Simonds, P. Qiu, E. D. Amir, P. O. Krutzik, R. Finck, R. V. Bruggner, R. Melamed, A. Trejo, O. I. Ornatsky, R. S. Balderas, S. K. Plevritis, K. Sachs, D. Pe'er, S. D. Tanner, and G. P. Nolan, *Science.* **332**, 687-696 (2011).
6. P. Qiu, E. F. Simonds, S. C. Bendall, K. D. Gibbs Jr, R. V. Bruggner, M. D. Linderman, K. Sachs, G. P. Nolan, and S. K. Plevritis, *Nat. Biotechnol.* **29**, 886-891 (2011).
7. S. C. Bendall, G. P. Nolan, M. Roederer, P. K. Chattopadhyay, *Cell*. **33**, 323-332 (2012).
8. E.-A. D. Amir, K. L. Davis, M. D. Tadmor, E. F. Simonds, J. H. Levine, S. C. Bendall, D. K. Shenfeld, S. Krishnaswamy, G. P. Nolan, and D. Pe'er, *Nat. Biotechnol*. **31**, 545-552 (2013).
9. S. C. Bendall, K. L. Davis, E.-A. D. Amir, M. D. Tadmor, E. F. Simonds, T. J. Chen, D. K. Shenfeld, G. P. Nolan, and D. Pe'er, *Cell*. **157**, 714-725 (2014).
10. K. Shekhar, P. Brodin, M. M. Davis, and A. K. Chakraborty, *Proc Natl Acad Sci*. **111**, 202-207 (2014).
11. M. Setty, M. D. Tadmor, S. Reich-Zeliger, O. Angel, T. M. Salame, P. Kathail, K. Choi, S. C.

Bendall, N. Friedman, and D. Pe'er, *Nat. Biotechnol*. **34**, 637-645 (2016).
12. N. Samusik, Z. Good, M. H. Spitzer, K. L. Davis, and G. P. Nolan, *Nat. Methods*. **13**, 493-496 (2016).
13. B. Anchang, T. D. P. Hart, S. C. Bendall, P. Qiu, Z. Bjornson, M. Linderman, G. P. Nolan, and S. K. Plevritis, *Nat. Protocols*. **11**, 1264-1279 (2016).
14. A. P. Patel, I. Tirosh, J. J. Trombetta, A. K. Shalek, S. M. Gillespie, H. Wakimoto, D. P. Cahill, B. V. Nahed, W. T. Curry, R. L. Martuza, D. N. Louis, O. Rozenblatt-Rosen, M. L. Suvà, A. Regev, and B. E. Bernstein, *Science.* **344**, 1396-1401 (2014).
15. Q. Deng, D. Ramskold, B. Reinius, and R. Sandberg, *Science*. **343**, 193-196 (2014).
16. D. R. Bandura, V. I. Baranov, O. I. Ornatsky, A. Antonov, R. Kinach, X. Lou, S. Pavlov, S. Vorobiev, J. E. Dick, and S. D. Tanner, *Anal. Chem.* **81**, 6813-6822 (2009).
17. F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B. B. Tuch, A. Siddiqui, K. Lao, and M. A. Surani, *Nat. Methods*. **6**, 377-382 (2009).
18. C. Trapnell, *Genome Res*. **25**, 1491-1498 (2015).
19. L. A. Herzenberg, J. Tung, W. A. Moore, and D. R. Parks, *Nat. Immunol*. **7**, 681-685 (2006).
20. C. Bishop, *Springer*. (2006).
21. H. C. Fan, G. K. Fu, and S. P. A. Fodor, *Science.* **347**. 1258367 (2015).
22. D. A. Lawson, N. R. Bhakta, K. Kessenbrock, K. D. Prummel, Y. Yu, K. Takai, A. Zhou, H. Eyob, S. Balakrishnan, C. Wang, P. Yaswen, A. Goga, and Z. Werb, *Nat*. **526**, 131-135 (2015).
23. L. J. P. van der Maaten, and G. E. Hinton, *J. Mach. Learn. Res*. **9**, 2579-2605 (2008).
24. L. J. P. van der Maaten, *J. Mach. Learn. Res*. **15**, 3221-3245 (2014).
25. J. Tang, J. Liu, M. Zhang, and Q. Mei, *Proc. 25th Int. Conf. WWW*. (2016).
26. S. Dasgupta and Y. Freund, *Proc. 40th ACM STOC*. 537-546 (2008).
27. W. Dong, M. Charikar, and K. Li, *Proc. 20th Int. Conf. WWW*. 577-586 (2011).
28. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, *Proc. 26th Adv. NIPS.* 3111-3119 (2013).
29. M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, *Proc. 20th ACM SoCG*. 253-262 (2004).
30. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, *Proc. 24th Int. Conf. WWW*. 1067-1077 (2015).
31. M. Muja and D. G. Lowe, *IEEE Trans Pattern Anal Mach Intell*. **36**, 2227-2240 (2014).
32. J. Barnes and P. Hut, *Nat*. **324**, 446-449 (1986).
33. M. U. Gutmann and A. Hyvarinen, *J. Mach. Learn. Res*. **13**, 307-361 (2012).
34. B. Recht, C. Re, S. Wright, and F. Niu, *Proc. 24th Adv. NIPS*. 693-701 (2011).
35. M. H. Spitzer, P. F. Gherardini, G. K. Fragiadakis, N. Bhattacharya, R. T. Yuan, A. N. Hotson, R. Finck, Y. Carmi, E. R. Zunder, W. J. Fantl, S. C. Bendall, E. G. Engleman, G. P. Nolan, *Science*. **349**, 1259425 (2015).
36. J. Munkres, *J. Soc. Ind. Appl. Math*. **5**, 32-38 (1957).