Using Dimensionality Reduction to Optimize t-SNE

Rikhav Shah and Sandeep Silwal

University of California, Berkeley Massachusetts Institute of Technology

Introduction

A widely used practice to more easily visualize the cluster structure present in data is to first reduce the dimension of data to a low integer, typically two. A popular tool to do this is the t-distributed Stochastic Neighborhood Embedding (t-SNE) algorithm [7].

Given a set of N points $x_1, \cdots, x_N \in \mathbb{R}^d$, t-SNE first computes the similarity score p_{ij} between x_i and x_j defined as $p_{ij} = (p_{i|j} + p_{j|i})/(2N)$ where for a fixed i, $p_{j|i} \propto \exp(\|x_i - x_j\|^2/\sigma_i^2)$ for some parameter σ_i. Intuitively, the value p_{ij} measures the 'similarity' between points x_i and x_j. t-SNE then aims to learn the lower dimensional points $y_1, \cdots, y_N \in \mathbb{R}^2$ such that if $q_{ij} \propto (1 + \|y_i - y_j\|_2^2)^{-1}$, then q_{ij} minimizes the Kullback–Leibler divergence of the distribution $\{q_{ij}\}$ from the distribution $\{p_{ij}\}$. For a more detailed explanation of the t-SNE algorithm, see [7].

A drawback of t-SNE is its large computational cost, which largely can be written as $O(\#\text{ of distances computed} \cdot d)$ where d is the dimension of the input data. This drawback severely limits its potential use in unsupervised learning problems.

Practical developments have included parallel computation, GPU, and multicore implementations mostly focused on reducing the number of pairwise distances computed between the input points [1, 9, 11, 6]. We instead focus on reducing the value of d in the runtime cost.

In this work, we show that the distance computation can be optimized by first projecting the input data into a much smaller random subspace before running t-SNE. Our motivation for this approach comes from the field of metric embeddings where it is known that random projections preserve many geometric structures of the input data. [3, 4, 8]. One of the key results of this field is the Johnson–Lindenstrauss (JL) Lemma which roughly states that given any set of N points, a random projection of these points into dimension $O(\log N/\epsilon^2)$ preserves all pairwise distances up to multiplicative error $(1 \pm \epsilon)$ [3].

Our contributions are the following:

- We empirically show that projecting to a very small dimension (even as low as 5% of the original dimension in some cases) and then performing t-SNE preserves local cluster information just as well as t-SNE with no prior dimensionality reduction step.
- We empirically show that performing dimensionality reduction first and then running t-SNE is significantly faster than just running t-SNE.

Experimental Setup and Justification

We devise an *accuracy score* to measure the quality of the cluster structure of a low-dimensional embedding. First we note that ideal clusters are ones in which every element has the same label as others in the same cluster. Ideally, such a clustering would be created without looking at the labels ahead of time. A datapoint is said to be "correctly clustered" in the embedding if its label matches that of its nearest neighbor. The accuracy of an embedding then is given by the fraction of datapoints that are correctly clustered.

We construct embeddings by first applying a random projection into d' dimensions, then using t-SNE to arrive at a two dimensional embedding. The random projections are performed by using an appropriately sized random matrix with i.i.d. Gaussian entries $\mathcal{N}(0, 1/d)$ where d is the original dimension of the input data. We repeat this process for several values of d' exponentially spaced and ranging from 7 to d.

We use a highly optimized implementation of t-SNE called openTSNE [10] as well as the standard scikit-learn implementation.

We use the following four datasets in our experiments: MNIST, Kuzushiji-MNIST (KMnist) (data set of Japanese cursive characters in the same format as MNIST) [2], Fashion-MNIST (data set of fashion images in the same format as MNIST) [12], and the Small Norb data set (data set of images of toys) [5]. All of our experiments were done in Python 3.7 using one core of a MacBook Pro using a 2.7 GHz Intel Core $\text{i}5$ processor. We only report the time to perform t-SNE since the t-SNE runtime is many orders of magnitude larger than all the other steps. Our code can be accessed from the following GitHub reprository: `https://github.com/ssilwa/optml`

Results

We first compare the time taken to perform t-SNE and the accuracy scores achieved if a dimensionality reduction step is used before using t-SNE versus the case where no dimensionality reduction is used. We do so by plotting the ratio of the runtime and accuracy scores. For a given dimension, a higher accuracy score ratio is better and signifies that projecting down to that dimension using a random projection does not deteriorate the performance of t-SNE. Likewise, a lower time ratio is better. In figures 1a-1d, we plot the ratio of the accuracy scores and the time taken when we use the openTSNE implementation. The x axis is the dimension of the random projection (ranging all the way up to the actual dimension of the input data). In all four datasets, we see that as the dimension increases, both the accuracy ratio and the ratio of the time taken approach 1. However, we also observe that the accuracy score ratios approach 1 much faster, indicating a "sweet spot" where the dimension is high enough for the random projection to preserve geometric structure, while still low enough for t-SNE to run relatively quickly.

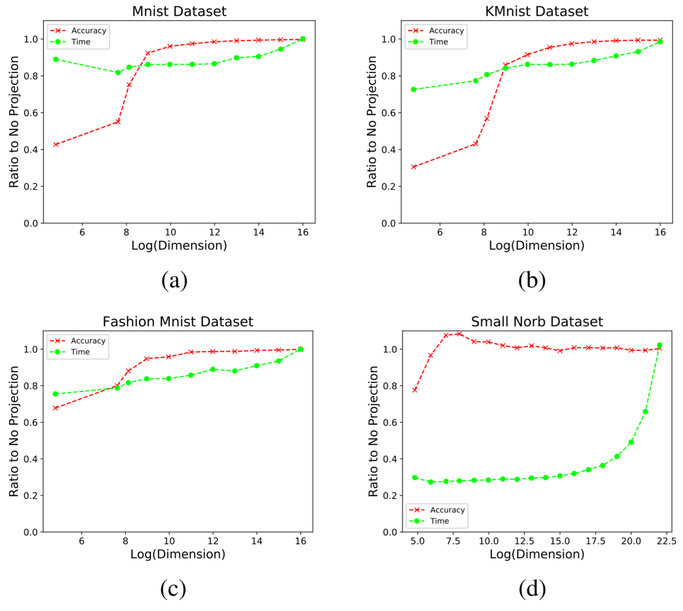

Fig. 1: Green: Ratio of time taken to run t-SNE after dimensionality reduction to time taken to run t-SNE without dimensionality reduction. Red: Ratio of accuracy score after dimensionality reduction to accuracy score without dimensionality reduction. The t-SNE implementation used was openTSNE. x axis shows the dimension after dimensionality reduction. The base of the logarithm is 1.5.

Likewise, we show the results of the same experiments using the scikit-learn implementation. Since the scikit-learn implementation is significantly slower than the openTSNE implementation, we subsample the d values used and do not test the Small Norb dataset. However, even using this different implementation, we again observe the same trends as in the openTSNE implementation.

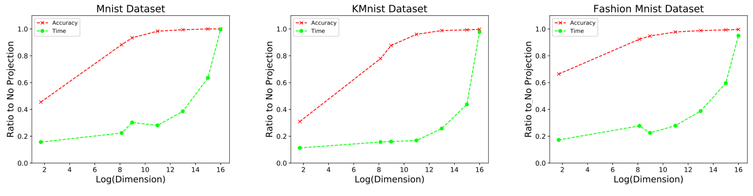

Fig. 2: Green: Ratio of time taken to run t-SNE after dimensionality reduction to time taken to run t-SNE without dimensionality reduction. Red: Ratio of accuracy score after dimensionality reduction to accuracy score without dimensionality reduction. The t-SNE implementation used was the scikit implementation. x axis shows the dimension after dimensionality reduction. The base of the logarithm is 1.5. Since the scikit implementation is very slow, we used fewer datapoints for the datasets shown and did not test on the Small Norb dataset.

Remarks

In practice, SVD methods such as principal component analysis (PCA) are more widely used than random projections for the general task of dimensionality reduction. We show that for our four datasets, PCA indeed outperforms random projections. That is, we empirically observe that using PCA, rather than a random projection before t-SNE, allows us to project to a much smaller dimension while still retaining a high accuracy score ratio, (compared to the case where no dimensionality reduction is used). This is shown in Figure 3 where regardless of the dataset, a projection to a dimension of $d = 25$ is sufficient to get an accuracy score ratio close to 1. An advantage of random projections however, is that it is *data oblivious* (the dimensionality reduction does not depend on the data) and it has provable guarantees in many cases, such as the JL lemma.

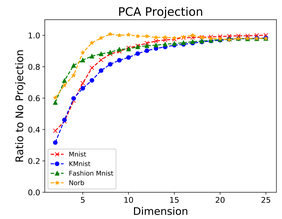

Fig. 3: Ratio of the accuracy score using PCA for dimensionality reduction instead of random projections and then performing t-SNE. We see that across all datasets, the accuracy scores become comparable to the scores when using t-SNE with no prior dimensionality reduction even for very small dimensions. This suggests that in practice, PCA might be a better choice than random projections to perform dimensionality reduction before t-SNE.

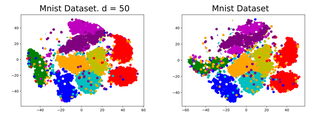

Fig. 4: Plotting the output of t-SNE. Left: a dimensionality reduction to $d = 50$ is performed before t-SNE on the MNIST dataset. Right: no dimensionality reduction is performed. Colors represent the label of the points. Observe that the same cluster structures appear in both plots.

References

[1] David M Chan et al. "GPU accelerated t-distributed stochastic neighbor embedding". In: *Journal of Parallel and Distributed Computing* 131 (2019), pp. 1–13.

[2] Tarin Clanuwat et al. *Deep Learning for Classical Japanese Literature*. Dec. 3, 2018. arXiv: cs.CV/1812.01718 [cs.CV].

[3] Sanjoy Dasgupta and Anupam Gupta. "An Elementary Proof of a Theorem of Johnson and Lindenstrauss". In: *Random Struct. Algorithms* 22.1 (Jan. 2003), pp. 60–65. ISSN: 1042-9832. DOI: 10.1002/rsa.10073. URL: http://dx.doi.org/10.1002/rsa.10073.

[4] Piotr Indyk and Assaf Naor. "Nearest-neighbor-preserving Embeddings". In: *ACM Trans. Algorithms* 3.3 (Aug. 2007). ISSN: 1549-6325. DOI: 10.1145/1273340.1273347. URL: http://doi.acm.org/10.1145/1273340.1273347.

[5] Yann LeCun, Fu Jie Huang, and Léon Bottou. "Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR'04. Washington, D.C., USA: IEEE Computer Society, 2004, pp. 97–104. URL: http://dl.acm.org/citation.cfm?id=1896300.1896315.

[6] Laurens van der Maaten. "Accelerating t-SNE using Tree-Based Algorithms". In: *Journal of Machine Learning Research* 15 (2014), pp. 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

[7] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[8] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. "Performance of Johnson-Lindenstrauss Transform for K-means and K-medians Clustering". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: ACM, 2019, pp. 1027–1038. ISBN: 978-1-4503-6705-9. DOI: 10.1145/3313276.3316350. URL: http://doi.acm.org/10.1145/3313276.3316350.

[9] N. Pezzotti et al. "GPGPU Linear Complexity t-SNE Optimization". In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. DOI: 10.1109/TVCG.2019.2934307.

[10] Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. "openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding". In: *bioRxiv* (2019). DOI: 10.1101/731877. eprint: https://www.biorxiv.org/content/early/2019/08/13/731877.full.pdf. URL: https://www.biorxiv.org/content/early/2019/08/13/731877.

[11] Dmitry Ulyanov. *Multicore-TSNE*. https://github.com/DmitryUlyanov/Multicore-TSNE. 2016.

[12] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].