Fast Optimization for t-SNE

Laurens van der Maaten
Department of Computer Science and Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
Pattern Recognition & Bioinformatics Lab, Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
lvdmaaten@gmail.com

Abstract

The paper presents an alternative optimization technique for t-SNE that is orders of magnitude faster than the original optimization technique, and that produces results that are at least as good.

1 Introduction

An important problem in the visualization of objects is how to construct a low-dimensional scatter plot in which each object is modeled by a single point, and in which the pairwise distances between points reflect the pairwise dissimilarities between their corresponding objects. Many techniques for dimensionality reduction and multidimensional scaling have been designed with this goal in mind; see, e.g., [1, 2, 3] for an overview. Recently, a technique called t-Distributed Stochastic Neighbor Embedding (t-SNE) has gained popularity. In particular, it has been successfully applied to visualize, among others, documents [4], optimization procedure trajectories [5], breast cancer CADx data [6], linguistic data [7], paintings [8], and data on malicious software [9, 10].

One of the main disadvantages of t-SNE is that it is computationally expensive; the visualization of a typical data set with $5,000$ objects can take up to 30 minutes, even using advanced implementations running on high-end hardware. The high computational expenses of performing t-SNE limit its applicability: in many domains, data analysts want to quickly construct a visualization, change one or more parameters underlying the objects' pairwise similarities, and quickly construct a new visualization to evaluate the result of the parameter change, etc. Such interactions with the visualization technique are hampered by long waiting times for the construction of the visualizations.

In this paper, we aim to address the high computational costs of t-SNE by presenting an alternative optimization technique for t-SNE that is orders of magnitude faster than the original t-SNE optimizer proposed in [11], whilst constructing solutions that are just as good in terms of the objective function.

2 t-Distributed Stochastic Neighbor Embedding

The input of t-SNE consists of a collection of pairwise (Euclidean) distances δ_{ij} between the n input objects, that are converted into conditional probabilities as $p_{j|i} = \frac{\exp(-\delta_{ij}^2/2\sigma_i)}{\sum_{k \neq i} \exp(-\delta_{ik}^2/2\sigma_i)}$ (where $p_{i|i} = 0$ and σ_i is set in such a way as to obtain conditional probability distributions with a fixed perplexity), which are in turn symmetrized to obtain a joint probability matrix $\mathbf{P}$ over pairs of points with entries $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. Pairwise distances between points $\mathbf{y}_i$ in the map of the input objects are converted into a joint probability matrix $\mathbf{Q}$ in a similar way, but instead of Gaussian densities, densities under a Student-t distribution are employed

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

The map points $\mathbf{y}_i$ are laid out in such a way as to minimize the Kullback-Leibler divergence between the joint probability distributions $\mathbf{P}$ and $\mathbf{Q}$, i.e., the objective function $C(\mathbf{Y}) = \text{const} - \sum_i \sum_{j \neq i} p_{ij} \log q_{ij}$. In the original paper [11], the optimization is performed using a gradient descent optimizer that uses momentum and a delta-bar-delta scheme [12] to update the learning rate.

3 New Optimization Technique

The gradient of the t-SNE cost function is given by

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

Denoting the graph Laplacian of the matrix $\mathbf{P}$ by $\mathbf{L}_P = \text{diag}(\sum_j p_{ij}) - \mathbf{P}$ and the graph Laplacian of $\mathbf{Q}$ by $\mathbf{L}_Q = \text{diag}(\sum_j q_{ij}) - \mathbf{Q}$, the gradient can be rewritten as

$$\frac{\partial C(\mathbf{Y})}{\partial \mathbf{Y}} = 4\mathbf{W} \circ (\mathbf{L}_P - \mathbf{L}_Q)\mathbf{Y}^{\mathrm{T}}, \tag{3}$$

where $\circ$ represents an element-wise product, and where $\mathbf{W}$ represents a matrix with weights $w_{ij} = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$. An important advantage of this way of writing the gradient is that the main computation consists of a matrix multiplication of two matrices, which facilitates the use of highly optimized BLAS implementations such as those in Intel MKL or CUBLAS.

Setting the t-SNE gradient to zero, we can investigate several splits in an attempt to identify a fixed point iteration for t-SNE. For instance, we can consider

$$4\mathbf{W} \circ (\mathbf{L}_P - \mathbf{L}_Q)\mathbf{Y}^{\mathrm{T}} = 0 \tag{4}$$

$$(\mathbf{W} \circ \mathbf{L}_P)\mathbf{Y}^{\mathrm{T}} = (\mathbf{W} \circ \mathbf{L}_Q)\mathbf{Y}^{\mathrm{T}} \tag{5}$$

$$\mathbf{Y} = \mathbf{Y}(\mathbf{W} \circ \mathbf{L}_P)^{-1}(\mathbf{W} \circ \mathbf{L}_Q). \tag{6}$$

We also investigated several other splits, but we focus on this split in the paper as it works best in practice. Unfortunately, the update rule derived above is not a fixed point iteration, and neither is any of the other splits we investigated. In other words, there is no guarantee that the update decreases $C(\mathbf{Y})$ at every iteration. However, the update rule derived above does suggest a good search direction that can be used instead of the gradient direction, viz. the search direction

$$\mathbf{D} = \mathbf{Y}(\mathbf{W} \circ \mathbf{L}_P)^{-1}(\mathbf{W} \circ \mathbf{L}_Q) - \mathbf{Y}, \tag{7}$$

where we note that the matrix inversion does not need to be computed explicitly; instead, we can use an off-the-shelf linear system solver to compute the above search direction. A similar search direction is also used in a dimensionality reduction technique called "elastic embedding" [13].

Although the update rule derived above does not always decrease the value of $C(\mathbf{Y})$, the corresponding search direction never becomes orthogonal (or obtuse) to the true gradient, i.e., the directional derivative of the search direction always remains negative. Hence, as a result of Zoutendijk's theorem [14], we are guaranteed to converge to a local optimum of $C(\mathbf{Y})$ if we use the search direction in combination with a linesearch that satisfies the Wolfe conditions, i.e., a linesearch that satisfies

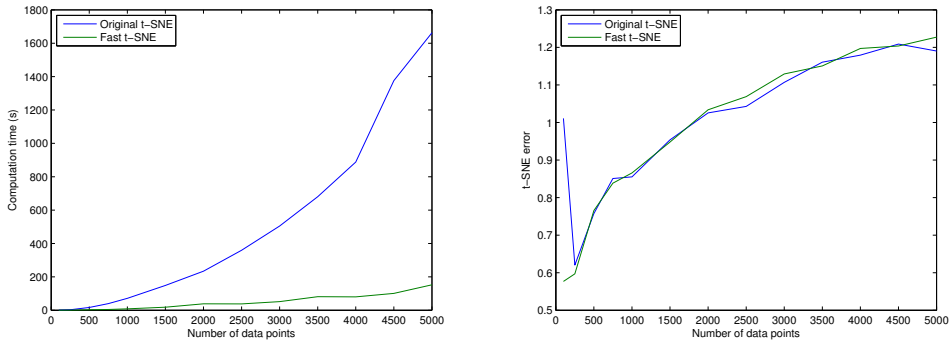
$$C(\mathbf{Y} + \alpha\mathbf{D}) \leq C(\mathbf{Y}) + c_1 \alpha \sum_i \sum_j d_{ij} \frac{\partial C(\mathbf{Y})}{\partial y_{ij}}, \qquad \text{(Armijo condition)}, \tag{8}$$

$$\sum_i \sum_j d_{ij} \frac{\partial C(\mathbf{Y} + \alpha\mathbf{D})}{\partial y_{ij}} \geq c_2 \sum_i \sum_j d_{ij} \frac{\partial C(\mathbf{Y})}{\partial y_{ij}}, \qquad \text{(curvature condition)}, \tag{9}$$

where α is the step size we aim to determine, and c_1 and c_2 are free parameters.

4 Experiments

We performed experiments on the MNIST data set, in which we compare the optimizer that uses the search direction $\mathbf{D}$ and a Wolfe linesearch algorithm with the optimizer proposed in the original



(a) Computation time.

(b) Lowest cost function.

Figure 1: Required computation time and lowest cost function value achieved as a function of the number of instances; for both optimizers.

t-SNE paper [11]. Both optimizers were implemented in Matlab. Profiling of both implementations revealed that over 95% of all computation time is in computations that are implemented by BLAS and LAPACK; hence, we may safely ignore the overhead from the Matlab calls and can readily compare computation times.

In the experiments with both implementations, we initialized the maps $\mathbf{Y}$ using PCA. The original t-SNE optimizer uses $1,000$ iterations of gradient descent with a momentum of 0.5 during the first 250 iterations and a momentum of 0.8 afterwards, as well as a delta-bar-delta scheme [12] for the step size η with a starting value of $\alpha = 500$. We did not employ the "early exaggeration" trick [11] to facilitate a fair comparison of the cost function values over time. The t-SNE optimizer proposed in this paper uses an off-the-shelf Wolfe linesearch algorithm that uses bracketing with a cubic interpolation/extrapolation method that employs both function and gradient values [15] (using a maximum number of 10 function evaluations per linesearch). The Armijo update parameter was set to $c_1 = 0.02$, whereas the curvature parameter was set to $c_2 = 0.9$. The optimization is continued until the step size α becomes smaller than 10^{-5}, until the improvement of the cost function value is smaller than 50^{-4}, or until a preset maximum of 50 iterations is reached. A Matlab implementation of the new optimizer is available from `http://homepage.tudelft.nl/19j49/t-SNE.html`.

We performed experiments for various number of instances (ranging from 100 to $5,000$), and we heuristically set the perplexity-values for each experiment to a suitable value. In order to make the comparison as fair as possible, we excluded the PCA computations as well as the computation of the matrix $\mathbf{P}$ from the measurement of computation times, hence, we only measure the computation time required for the actual optimization. Computation times were measured on an Intel Core i7 2.66 GHz mobile processor.

The required computation time for both optimizers as a function of the number of instances is shown in Figure 1(a). The figure reveals the strong performance of the new optimizer: whereas the computation time consumed by the original optimizer roughly grows quadratically with the number of instances, the new optimizer appears to scale nearly linear in the number of instances (even though its worst-case performance is $\mathcal{O}(n^3)$). In Figure 1(b), we show the lowest cost function value achieved by both optimizers as a function of the number of instances[1]. The figure reveals that, despite being orders of magnitude faster on larger data sets, the new optimizer does construct solutions of the same quality in terms of the t-SNE cost function.

In Figure 4, we plot the value of the t-SNE cost function as a function of computation time for both optimizers. The plot presents results for the visualization of $5,000$ MNIST digits. It underlines the rapid convergence of the new optimizer compared to the original t-SNE algorithm.

[1]Note that cost values tend to increase with the number of instances.

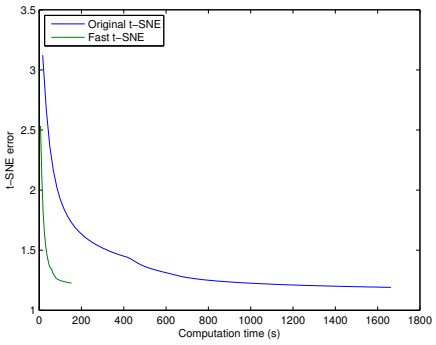


Figure 2: Value of the t-SNE cost function as a function of computation time (for the visualization of $5,000$ MNIST digits); for both optimizers.

5 Concluding Remarks

We proposed a new optimizer for t-SNE that uses an alternative search direction in combination with a Wolfe linesearch algorithm. The new optimizer is orders of magnitude faster than the original t-SNE optimizer, whilst producing similar results. All computation time in the new optimizer is spent on matrix multiplication and linear system solving, as a result of which we expect large additional speed-ups are possible by implementing the algorithm in CUDA. We leave such an implementation to future work.

Acknowledgements

The author thanks Miguel Carreira-Perpiñán, Geoffrey Hinton, Fei Sha, and Lawrence Saul for helpful discussions on optimization and dimensionality reduction.

References

[1] C.J.C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4):1–95, 2010.

[2] J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York, NY, 2007.

[3] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[4] S. Lacoste-Julien, F. Sha, and M.I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems*, volume 21, pages 897–904, 2009.

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Mar):625–660, 2010.

[6] A.R. Jamieson, M.L. Giger, K. Drukker, H. Li, Y. Yuan, and N. Bhooshan. Exploring nonlinear feature space dimension reduction and data representation in breast CADx with Laplacian Eigenmaps and t-SNE. *Medical Physics*, 37(1):339–351, 2010.

[7] Y. Mao, K. Balasubramanian, and G. Lebanon. Dimensionality reduction for text using domain knowledge. In *Proceedings of the* 23^{rd} *International Conference on Computational Linguistics*, pages 801–809, 2010.

[8] L.J.P. van der Maaten and E.O. Postma. Texton-based analysis of paintings. In *SPIE Optical Engineering and Applications*, volume 7798-16, 2010.

[9] I. Gashi, V. Stankovic, C. Leita, and O. Thonnard. An experimental study of diversity with off-the-shelf antivirus engines. In *Proceedings of the IEEE International Symposium on Network Computing and Applications*, pages 4–11, 2009.

[10] O. Thonnard, W. Mees, and M. Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, pages 11–21, 2009.

[11] L.J.P. van der Maaten and G.E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2431–2456, 2008.

[12] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[13] M.Á. Carreira-Perpiñán. The elastic embedding algorithm for dimensionality reduction. In *Proceedings of the* 27^{th} *International Conference on Machine Learning*, pages 167–174, 2010.

[14] G. Zoutendijk. *Methods of Feasible Directions*. Elsevier Publishing Company, Amsterdam, The Netherlands, 1960.

[15] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.