Eurographics Conference on Visualization (EuroVis) 2016
E. Bertini, N. Elmqvist, and T. Wischgoll
(Guest Editors)

Short Paper

Visualizing Time-Dependent Data Using Dynamic t-SNE

Paulo E. Rauber [1,2], Alexandre X. Falcão[2], Alexandru C. Telea[1]

[1] Johann Bernoulli Institute, University of Groningen, the Netherlands
[2] Institute of Computing, University of Campinas, Brazil

Abstract

Many interesting processes can be represented as time-dependent datasets. We define a time-dependent dataset as a sequence of datasets captured at particular time steps. In such a sequence, each dataset is composed of observations (high-dimensional real vectors), and each observation has a corresponding observation across time steps. Dimensionality reduction provides a scalable alternative to create visualizations (projections) that enable insight into the structure of such datasets. However, applying dimensionality reduction independently for each dataset in a sequence may introduce unnecessary variability in the resulting sequence of projections, which makes tracking the evolution of the data significantly more challenging. We show that this issue affects t-SNE, a widely used dimensionality reduction technique. In this context, we propose dynamic t-SNE, an adaptation of t-SNE that introduces a controllable trade-off between temporal coherence and projection reliability. Our evaluation in two time-dependent datasets shows that dynamic t-SNE eliminates unnecessary temporal variability and encourages smooth changes between projections.

Categories and Subject Descriptors (according to ACM CCS): Human-centered computing – Information visualization; Computing methodologies – Dimensionality reduction and manifold learning

1. Introduction

Time-oriented data visualization is a widely researched subject. According to [AMST11], current techniques can be categorized as abstract or spatial, univariate or multivariate, linear or cyclic, instantaneous or interval-based, static or dynamic, and two or three-dimensional. Our work is concerned with abstract, multivariate, and instantaneous time-oriented visualization.

The visualization of (high-dimensional) multivariate data is another vast subject [LMW*15]. In this context, dimensionality reduction (DR) has been successfully applied to compute projections: representations of high-dimensional datasets in low-dimensional spaces (typically 2D) that retain the structure of the original data. Such structure is related to local density, relationships between observations, and presence of clusters [LWBP14, LV05]. When compared to other high-dimensional data visualization techniques (*e.g.*, parallel coordinates, subspace clustering), DR is notably more scalable (visually and computationally) with respect to the number of dimensions and observations [LMW*15, VdMH08].

We define a time-dependent dataset as a sequence of datasets captured at particular time steps. In such a sequence, each dataset is a set of observations, and each observation has a corresponding observation across time steps. In simple terms, each observation *evolves* with time (or any other discrete parameter). Consider the task of visualizing a time-dependent dataset. If a DR technique is applied independently for each time step, the resulting sequence of projections may present variability that does not reflect significant changes in the *structure* of the data. We refer to this issue as *temporal incoherence*, which significantly impairs the visualization of temporal trends. In this paper, we will show that this issue affects t-SNE [VdMH08], a widely used DR technique, which achieves high-quality results in many applications [VdMH08]. Furthermore, temporal incoherence will affect any DR technique that is sensitive to relatively small changes in their inputs [GFVLDB13].

In this context, we propose dynamic t-SNE: an adaptation of t-SNE that allows a controllable trade-off between temporal coherence and spatial coherence (defined as preservation of structure at a particular time step). Previous work on this trade-off has been restricted to the context of dynamic graph drawing [XKHI13, LS08], even though there are many examples of time-dependent high-dimensional data visualizations based on DR [JFSK16, BWS*12, ABPdO12]. As will become clear, our approach can be easily extended to other optimization-based DR techniques.

This paper is organized as follows. Section 2 introduces our notation and briefly summarizes t-SNE. Section 3 explains the necessity for a controllable bias towards temporal coherence, and presents our proposed solution. Section 4 presents a preliminary evaluation of this proposal. Finally, Section 5 summarizes our contributions.

2. t-SNE

A dataset $\mathcal{D} = \mathbf{x}_1, \dots, \mathbf{x}_N$ is a sequence of observations, which are D-dimensional real vectors. The goal of t-SNE is to compute a sequence of points (projection) $\mathcal{P} = \mathbf{p}_1, \dots, \mathbf{p}_N$ where the *neighborhoods* from $\mathcal{D}$ are preserved, considering that each $\mathbf{p}_i \in \mathbb{R}^d$ corresponds to $\mathbf{x}_i \in \mathbb{R}^D$. Typically, $d = 2$ and $D \gg d$.

We will let $d_{i,j} = ||\mathbf{x}_i - \mathbf{x}_j||$ denote the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Analogously, $r_{i,j} = ||\mathbf{p}_i - \mathbf{p}_j||$.

Firstly, consider a random process where the probability of choosing the next $\mathbf{x}_j$ given the current $\mathbf{x}_i$ is defined as

DOI: 10.2312/eurovisshort.20161164

$$P(X' = j \mid X = i) = \frac{\exp\left(-\frac{d_{i,j}^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d_{i,k}^2}{2\sigma_i^2}\right)},$$

except for $i = j$, when $P(X' = j \mid X = i) = 0$. Each parameter $\sigma_i > 0$ is chosen in such a way that the *perplexity* $\kappa = 2^{H[X'|X=i]}$ matches a pre-defined value, where $H[X]$ denotes the entropy of X. In simple terms, $P(X' = j \mid X = i)$ is high whenever $\mathbf{x}_j$ is near $\mathbf{x}_i$ relative to the *observation density* near $\mathbf{x}_i$.

Consider also a distinct random process where the probability of choosing a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \times \mathcal{D}$ is defined as

$$P(X' = i, X = j) = \frac{P(X' = j \mid X = i) + P(X' = i \mid X = j)}{2N}.$$

Intuitively, $P(X' = i, X = j)$ is high whenever $P(X' = j \mid X = i)$ or $P(X' = i \mid X = j)$ is high.

In $\mathbb{R}^d$, the probability of choosing a pair $(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{P} \times \mathcal{P}$ in yet another random process is defined as

$$P(Y' = i, Y = j) = \frac{(1 + r_{i,j}^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + r_{k,l}^2)^{-1}},$$

except for $i = j$, when $P(Y' = i, Y = j) = 0$. Clearly, $P(Y' = i, Y = j)$ is high whenever $\mathbf{p}_i$ and $\mathbf{p}_j$ are close.

T-SNE aims at minimizing the following cost C with respect to $\mathcal{P}$ [VdMH08]:

$$C = \sum_i \sum_{j \neq i} P(X' = i, X = j) \log \left[\frac{P(X' = i, X = j)}{P(Y' = i, Y = j)} \right].$$

For our purposes, it suffices to note that C corresponds to the Kullback-Leibler divergence between $P(X', X)$ and $P(Y', Y)$, which heavily penalizes $P(X' = i, X = j) \gg P(Y' = i, Y = j)$, *i.e.*, placing *neighbors* in $\mathcal{D}$ far apart in $\mathcal{P}$.

The gradient of C with respect to a point $\mathbf{p}_i \in \mathcal{P}$ is given by

$$\nabla_{\mathbf{p}_i} C = 4 \sum_j (\mathbf{p}_i - \mathbf{p}_j) \frac{P(X' = i, X = j) - P(Y' = i, Y = j)}{1 + r_{i,j}^2}.$$

Geometrically, $\nabla_{\mathbf{p}_i} C$ is a combination of vectors pointing in the direction $\mathbf{p}_i - \mathbf{p}_j$, for every j. Each vector $\mathbf{p}_i - \mathbf{p}_j$ is also weighted by whether $\mathbf{p}_j$ should be moved closer to $\mathbf{p}_i$ to preserve neighborhoods from $\mathcal{D}$, and by whether $\mathbf{p}_j$ is close to $\mathbf{p}_i$.

The cost C is usually minimized with respect to $\mathcal{P}$ by (momentum-based) gradient descent: from an arbitrary initial $\mathcal{P}$, for a number of iterations, each $\mathbf{p}_i \in \mathcal{P}$ is moved in the direction $-\nabla_{\mathbf{p}_i} C$. For more details, we refer to [VdMH08].

3. Dynamic t-SNE

Consider the task of creating a sequence of projections $\mathcal{P}[1], \dots, \mathcal{P}[T]$ for a time-dependent dataset $\mathcal{D}[1], \dots, \mathcal{D}[T]$, where each $\mathbf{x}_i[t] \in \mathcal{D}[t]$ corresponds to $\mathbf{x}_i[t+1] \in \mathcal{D}[t+1]$. Although we will say that the sequence of datasets represents a *time*-dependent process, this task is meaningful whenever there is correspondence between observations at different *steps*.

We will let $C[t]$ denote the usual t-SNE cost for dataset $\mathcal{D}[t]$ and projection $\mathcal{P}[t]$, as defined in the previous section. It is possible to apply t-SNE individually for each dataset in a sequence using at least four different strategies:

1. Initializing $\mathcal{P}[t]$ independently and randomly, for all t.
2. Initializing $\mathcal{P}[t]$ with the same random sequence, for all t.
3. Initializing $\mathcal{P}[1]$ randomly, and $\mathcal{P}[t+1]$ with the $\mathcal{P}[t]$ that results from minimizing $C[t]$, for all $t > 1$, or reversely.
4. Combining datasets from all time steps into a single dataset $\mathcal{D}$, and computing a single projection $\mathcal{P}$.

However, each of these strategies has significant drawbacks.

Strategies 1 and 2 often result in a sequence of projections with major changes in positioning of corresponding points in adjacent time steps (temporal incoherence). This issue cannot be corrected by rigid transformations (*e.g.*, rotations, translations), and makes tracking the evolution of the data more challenging (see Sec. 4.1).

Strategy 3 is viable in some cases. However, it lacks a mechanism to enforce temporal coherence after initialization. At the other extreme, the initial bias may be difficult for gradient descent to overcome, because of the diminished effect on $\nabla_{\mathbf{p}_i[t]} C[t]$ of a point that is distant from $\mathbf{p}_i[t]$. Furthermore, because t-SNE usually takes many iterations to converge, the optimization of $C[t]$ starts at a likely *advantaged* state when compared to the optimization of $C[t']$, for all $t' \ll t$. In this case, the evolution due to the optimization process can be mistaken for temporal evolution. As an extreme example, consider a particular sequence of 100 *identical* datasets, each with 2000 observations in $\mathbb{R}^{512}$. Figure 1 shows some projections that result from strategy 3, which are clearly misleading. Notice how there is significant *apparent* evolution between time steps 1 and 50 (10^3 and 5×10^4 gradient descent iterations, respectively). In fact, the configuration still changes between 5×10^4 and 10^5 iterations, albeit more slowly. Running t-SNE for this many iterations (for each projection) is impractical, and tweaking the parameters to achieve faster convergence is not trivial. Although it suffices to realize that there is no actual temporal evolution in this time-dependent dataset, the experimental details are described in Sections 4 and 4.2. In summary, the major issue with strategy 3 is the lack of control over how the optimization is biased.

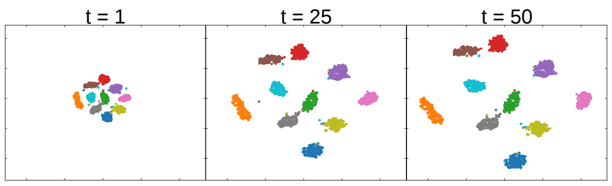

Figure 1: *Strategy 3 results on a sequence of* identical *datasets (last CNN hidden layer fixed at epoch* 1*, MNIST test subset).*

Strategy 4 can be dismissed in many cases. Firstly, when the distance matrix for $\mathcal{D}$ and all σ_i are given as inputs, and the target dimension d is seen as a constant, t-SNE has time complexity $O(N^2)$ for N observations. Thus, strategy 4 quickly becomes intractable. Secondly, it also introduces significant *clutter*, which cannot be eliminated by filtering points per time step, since that introduces misleading void spaces. Finally, depending on context, combining *structures* across different steps may be inappropriate.

Dynamic t-SNE, our proposal, is an alternative that overcomes the drawbacks of the previous strategies. The dynamic t-SNE cost C tries to preserve the neighborhoods from $\mathcal{D}[t]$ in $\mathcal{P}[t]$, for each t, but also penalizes each point for unnecessarily moving between time steps. This new cost introduces a hyperparameter $\lambda \geq 0$ that *controls* the bias for temporal coherence, and is defined as

$$C = \sum_{t=1}^{T} C[t] + \frac{\lambda}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T-1} || \mathbf{p}_i[t] - \mathbf{p}_i[t+1] ||^2.$$

Intuitively, each point is penalized in proportion to the total squared length of the line segments formed by its movement through time. This penalty is similar to the one proposed by [LS08] for dynamic graph drawing using multidimensional scaling (MDS). Although it would be possible to penalize each movement in $\mathbb{R}^d$ in proportion to the corresponding movement in $\mathbb{R}^D$, that would have undesirable consequences. For instance, any transformation that moved observations significantly while preserving most pairwise distances would justify significant changes in the projection. This is undesirable because the t-SNE cost depends solely on distances, which makes a projection convey only relative positioning.

It is easy to show that the gradient $\nabla_{\mathbf{p}_i[t]} C$ of C with respect to a point $\mathbf{p}_i[t] \in \mathcal{P}[t]$ is given by

$$\nabla_{\mathbf{p}_i[t]} C = \nabla_{\mathbf{p}_i[t]} C[t] + \frac{\lambda}{N} \mathbf{v}_i[t],$$

where $\nabla_{\mathbf{p}_i[t]} C[t]$ is the usual t-SNE cost gradient (with respect to $\mathbf{p}_i[t]$) when the dataset $\mathcal{D}[t]$ is considered separately, and $\mathbf{v}_i[t]$ is defined as

$$\mathbf{v}_i[t] = \begin{cases} 2\mathbf{p}_i[t] - (\mathbf{p}_i[t-1] + \mathbf{p}_i[t+1]) & \text{if } 1 < t < T, \\ \mathbf{p}_i[t] - \mathbf{p}_i[t+1] & \text{if } t = 1, \\ \mathbf{p}_i[t] - \mathbf{p}_i[t-1] & \text{if } t = T. \end{cases}$$

Just as $\nabla_{\mathbf{p}_i[t]} C[t]$, each vector $\mathbf{v}_i[t]$ also has a geometrical interpretation. For $1 < t < T$, the vector $\mathbf{v}_i[t]$ has *opposite* direction to any vector that points from $\mathbf{p}_i[t]$ to the midpoint between $\mathbf{p}_i[t-1]$ and $\mathbf{p}_i[t+1]$. Thus, in gradient *descent*, the parameter λ controls the trade-off between moving each $\mathbf{p}_i[t]$ in a direction that tries to preserve neighborhoods from $\mathcal{D}$, and moving each $\mathbf{p}_i[t]$ in a direction that minimizes the total squared length of line segments in the polyline $(\mathbf{p}_i[t-1], \mathbf{p}_i[t], \mathbf{p}_i[t+1])$.

4. Evaluation

We implemented t-SNE and dynamic t-SNE in Python, using Theano [BLP*12], Numpy [VDWCV11], and scikit-learn [PVG*11] [†]. Theano allows writing mathematical expressions that can be automatically translated into optimized (CPU or GPU) code and evaluated. Our implementation uses automatic differentiation, which can be highly valuable for adapting t-SNE to a particular application. For instance, altering the symbolic expression that defines the cost does not require manually finding (possibly involved) partial derivatives analytically, nor changing the optimization process. Dynamic t-SNE requires roughly the same computational time as executing t-SNE independently for each time step (Strategies 1-3). Using an *Intel i7-2600* at 3.4 GHz with a GeForce GTX 590, both (GPU) implementations require approx. 6 minutes *per time step* for the time-dependent dataset in Sec. 4.1.

The remainder of this section presents our preliminary experimental evaluation of dynamic t-SNE. The implementation details and hyperparameter choices are very similar to those of publicly available implementations [VDM16]. We use momentum-based gradient descent for minimizing C, with a learning rate $\eta = 2400$ and momentum coefficient $\mu = 0.5$, which change to $\eta = 250$ and $\mu = 0.8$ at iteration 250. The optimization is run for 1000 iterations, with a perplexity $\kappa = 70$. We sample the initial coordinates of each point from a Gaussian distribution with zero mean and standard deviation 10^{-4}. The binary search for each σ_i lasts 50 iterations. For dynamic t-SNE, every projection $\mathcal{P}[t]$ is initialized equally.

4.1. Multivariate Gaussians

We create the multivariate Gaussians dataset specifically as a controlled experiment for dynamic t-SNE. Firstly, we sample 200 observations from each of 10 distinct (isotropic) 100-dimensional multivariate Gaussian distributions with variance 0.1. We combine the resulting 2000 observations into a single dataset $\mathcal{D}[1]$. Each multivariate Gaussian has a distinct mean, which is chosen uniformly between the standard basis vectors for $\mathbb{R}^{100}$. Given $\mathcal{D}[t]$, the dataset $\mathcal{D}[t+1]$ is created as follows. Each observation $\mathbf{x}[t+1] \in \mathcal{D}[t+1]$ corresponds to an observation $\mathbf{x}[t] \in \mathcal{D}[t]$ moved 10% of the remaining distance closer to the mean of its corresponding multivariate Gaussian. In simple terms, each of the 10 clusters becomes more *compact* as t increases. We consider $T = 10$ datasets.

The sequence of images in Fig. 2a shows dynamic t-SNE results for $\lambda = 0$, which corresponds to strategy 2 (as defined in Sec. 3). Each point $\mathbf{p}_i[t]$ is colored, for illustration purposes, according to the distribution from which $\mathbf{x}_i[1]$ was sampled. Notice the large variability in *visual cluster* positioning between time steps, even after the clusters become well-defined. Because the process that originates the data simply makes the clusters gradually more compact, this variability is misleading. We preserve the scatterplot scale between time steps, which is also a significant source of variability.

In comparison, consider the results shown in Fig. 2b, for $\lambda = 0.1$. Notice how each cluster stays at a similar relative position during all steps, and only becomes more compact in later steps. When the projections are inspected step by step, it becomes easier to notice the movement of projection *outliers*, which is obscured when $\lambda = 0$.

Because each point is penalized for moving between projections, clear *visual separation* between clusters in later projections is also able to induce better separation in earlier projections. In simple terms, given a similar spatial coherence in two alternative projections for time step t, the projection that is more temporally coherent with the projection for time $t+1$ is preferred by the cost function. There is a trade-off: a large λ will induce unwanted bias, whereas a small λ will cause misleading temporal incoherence. The major benefit of dynamic t-SNE is precisely the control over this trade-off. Although the choice of λ depends on context, we recommend first comparing $\lambda = 0$ with the results of an arbitrary low value.

4.2. Hidden layer activations (SVHN CNN)

The time-dependent dataset $\mathcal{D}[0], \dots, \mathcal{D}[T]$ considered in this section is composed of datasets of neural network activations. We developed dynamic t-SNE partially to overcome visualization problems encountered in this context. An activation vector $\mathbf{a}[t] \in \mathcal{D}[t]$ is a D-dimensional observation that represents the outputs of D neurons in a particular layer of an artificial neural network given a particular input. Such activation vector can be seen as an alternative representation of the input, learned by the network through an optimization process. Visualizing activation vectors allows valuable insight into how a network learns and operates, which is considered highly valuable by practitioners [ZF14, YCN*15, EBCV09].

In this particular case, each network input belongs to a subset of 2000 test images from the SVHN dataset [NWC*11], a traditional image classification benchmark, and is assigned to one of

[†] Available in `https://github.com/paulorauber/thesne`.

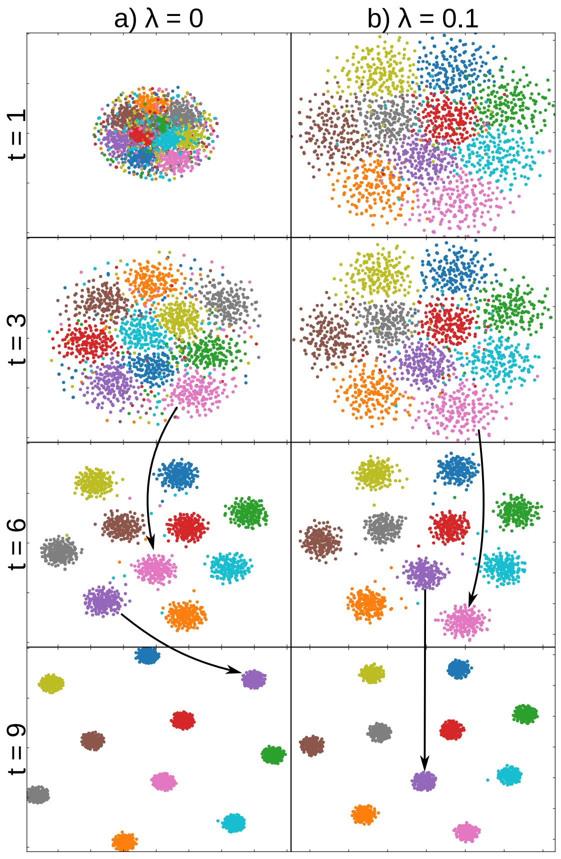

Figure 2: *Dynamic t-SNE results on Multivariate Gaussians.*

ten classes (according to the digit seen in the image), which we use to color the projections. For each t, an activation $\mathbf{a}[t] \in \mathcal{D}[t]$ is a 512-dimensional observation, and corresponds to the last hidden layer activation of a convolutional neural network (CNN) after t epochs of training (given a particular input image). The time-dependent dataset represents the evolution of the learned representations through 100 epochs. Although the interested reader may consult [RFFaT16] for details on the experiments that originated this dataset, these details are not strictly necessary to understand this section. Earlier in the text, the projections shown in Fig. 1 correspond to a similar dataset based on 2000 MNIST [LCB98] test images, and 100 copies of the same dataset after one training epoch.

Figures 3a and 3b compare the results of dynamic t-SNE for $\lambda = 0$ and $\lambda = 0.1$, respectively. Notice that the projections for step $t = 0$, which correspond to network activations before training, are noticeably different from those that follow. Clearly, the early epochs of training have a significant effect on the learned representations, which coincides with most of the increase in validation accuracy (not shown). Although both sequences show significant variation between steps $t = 25$ and $t = 100$, the remarkable distinction is that the projections change *smoothly* when $\lambda = 0.1$. For an example, compare the transition between steps 24 and 25 in Figs. 3a and 3b. This phenomenon can be seen consistently through the whole sequence. The visual separation between clusters does not seem to improve considerably after the early epochs, although it is hard to state whether there is significant variability in the structure of the data. Because $\lambda = 0.1$ does not seem to introduce a misleading bias in comparison to $\lambda = 0$, more evidence could be obtained by increasing λ even further.

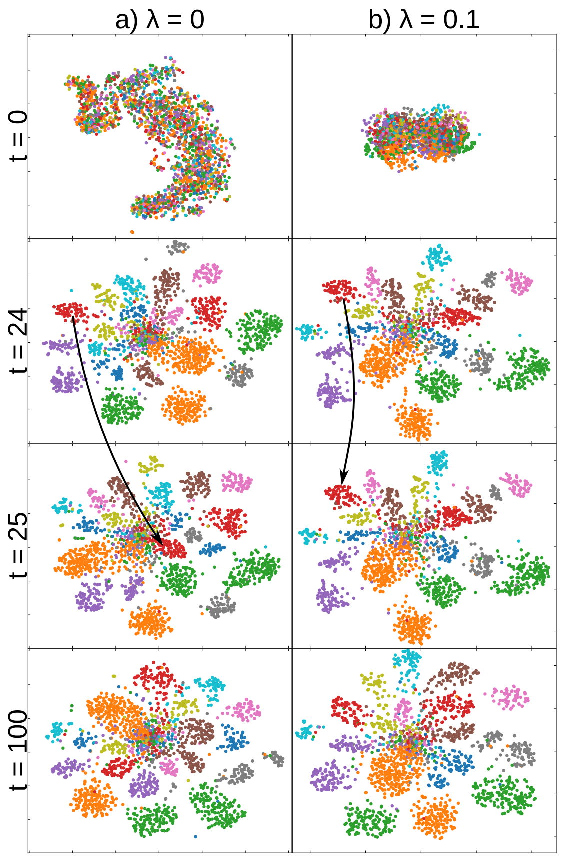

Figure 3: *Dynamic t-SNE results on SVHN CNN.*

5. Conclusion

In this paper, we have shown how dynamic t-SNE can be applied to create sequences of projections with increased temporal coherence, which facilitates tracking the evolution of high-dimensional time-dependent data. The main advantage of dynamic t-SNE over t-SNE is the control over the trade-off between temporal coherence (between successive projections) and spatial coherence (with respect to high-dimensional neighborhoods). This control depends on a single hyperparameter λ, which has a simple interpretation, and does not introduce a significant computational overhead. This approach can be easily adapted for other optimization-based DR techniques. Our preliminary experiments show promising results in eliminating unnecessary variability between projections.

Although we implemented dynamic t-SNE as an adaptation of *traditional* t-SNE, the Barnes-Hut approximation is significantly more computationally efficient [VDM14]. Future works that employ dynamic t-SNE for large datasets should consider a similar optimization. The current implementation has the advantage of being highly flexible with respect to cost functions.

Acknowledgments

We would like to thank CAPES, FAPESP (2012 / 24121-9) and CNPq (302970 / 2014-2, 479070 / 2013-0) for the financial support.

References

[ABPdO12] ALENCAR A. B., BÖRNER K., PAULOVICH F. V., DE OLIVEIRA M. C. F.: Time-aware visualization of document collections. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), ACM, pp. 997–1004. 1

[AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of time-oriented data*. Springer Science & Business Media, 2011. 1

[BLP*12] BASTIEN F., LAMBLIN P., PASCANU R., BERGSTRA J., GOODFELLOW I. J., BERGERON A., BOUCHARD N., BENGIO Y.: Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 3

[BWS*12] BERNARD J., WILHELM N., SCHERER M., MAY T., SCHRECK T.: Timeseriespaths : Projection-based explorative analysis of multivariate time series data. In *Journal of WSCG* (2012), pp. 97–106. 1

[EBCV09] ERHAN D., BENGIO Y., COURVILLE A., VINCENT P.: Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep 4323* (2009). 3

[GFVLDB13] GARCÍA FERNÁNDEZ F. J., VERLEYSEN M., LEE J. A., DÍAZ BLANCO I.: Stability comparison of dimensionality reduction techniques attending to data and parameter variations. In *Eurographics Conference on Visualization* (2013), The Eurographics Association. 1

[JFSK16] JAÌĹCKLE D., FISCHER F., SCHRECK T., KEIM D. A.: Temporal MDS plots for analysis of multivariate data. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 141–150. 1

[LCB98] LECUN Y., CORTES C., BURGES C. J.: The MNIST database of handwritten digits, 1998. Available at http://http://yann.lecun.com/exdb/mnist. 4

[LMW*15] LIU S., MALJOVEC D., WANG B., BREMER P.-T., PASCUCCI V.: Visualizing high-dimensional data: Advances in the past decade. In *Proc. EuroVis – STARs* (2015). 1

[LS08] LEYDESDORFF L., SCHANK T.: Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. *Journal of the American Society for Information Science and Technology 59*, 11 (2008), 1810–1818. 1, 3

[LV05] LEE J. A., VERLEYSEN M.: Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing 67* (2005), 29–53. 1

[LWBP14] LIU S., WANG B., BREMER P.-T., PASCUCCI V.: Distortion-guided structure-driven interactive exploration of high-dimensional data. *CGF 33*, 3 (2014), 101–110. 1

[NWC*11] NETZER Y., WANG T., COATES A., BISSACCO A., WU B., NG A. Y.: Reading digits in natural images with unsupervised feature learning. In *Proc. NIPS* (2011), vol. 2011, p. 5. 3

[PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., ET AL.: Scikit-learn: Machine learning in python. *J Mach Learn Res 12* (2011), 2825–2830. 3

[RFFaT16] RAUBER P. E., FADEL S. G., FALCÃO A. X., TELEA A. C.: The hidden activity of artificial neural networks, 2016. In preparation. 4

[VDM14] VAN DER MAATEN L.: Accelerating t-SNE using tree-based algorithms. *J Mach Learn Res 15*, 1 (2014), 3221–3245. 4

[VDM16] VAN DER MAATEN L.: t-SNE - Laurens van der Maaten, 2016. Available at https://lvdmaaten.github.io/tsne. 3

[VdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *J Mach Learn Res 9*, 2579-2605 (2008), 85. 1, 2

[VDWCV11] VAN DER WALT S., COLBERT S. C., VAROQUAUX G.: The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering 13*, 2 (2011), 22–30. 3

[XKHI13] XU K. S., KLIGER M., HERO III A. O.: A regularized graph layout framework for dynamic network visualization. *Data Mining and Knowledge Discovery 27*, 1 (2013), 84–116. 1

[YCN*15] YOSINSKI J., CLUNE J., NGUYEN A., FUCHS T., LIPSON H.: Understanding neural networks through deep visualization. In *Proc. ICML* (2015). 3

[ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *Proc. ECCV*. Springer, 2014, pp. 818–833. 3