Focused Stochastic Neighbor Embedding

Tuan Le, Sanuj Kumar; New Mexico State University, Rafael Baez Ramirez; University of Texas at El Paso

***Abstract*—Dimensional reduction algorithms reduce complexity in data; making computation cheaper, and resulting models more interpretable. Existing algorithms all differ in capabilities and effectiveness at preserving information. Focused Stochastic Neighbor Embedding aims to show the effects of augmenting existing algorithms with a framework that will improve the method's ability to preserve local structure without sacrificing global structure. We added weights and rederived the new cost function for implementation in python. We compared our augmented algorithm to a baselines of a few others using metrics that are defined to measure the local and global discrepancy and a k-nearest neighbors classifier. Using some of the same data sets as that paper, we ran our algorithm on those and recorded the metrics below. As a result we saw that our algorithm does improve the original algorithm's performance in the local preservation of structure without losing too much global structure.**

1. Introduction

In data science, big data includes a large quantity of information that is believed to be necessary to solve a problem. The issue is two-fold, the data needs to be sampled to be used practically in any algorithm, and usually includes many features for every sample. Dimensional reduction algorithms solve one of these issues by reducing the amount of feature or dimensions in a data set. Data sets that are less complex make computations done with the data less expensive, requiring less resources from a single group or cluster of computers, and also make any resulting models more interpretable. Two dimensions are easily plotted and shown on papers and is easy to understand the results and their interpretation. Three dimensions might be plotted and shown in an interactive model on computers or would need to be displayed physically, though the ability to interpret the results would depend on one's familiarity with the data or problem to interpret the mode. Big data ranges from dimensions in the tens to the hundreds and thousands, it is not feasible for any person to interpret all of the data simultaneously.

The uniqueness of every data set brought many different algorithms, each differing in capabilities and goals. Algorithms that focus on preserving the local structure, others that focus on the overall structure of the data, quick algorithms at the expense of a little accuracy, and so on. But completely preserving data from extremely high dimensions to low is impossible, a simple example are equidistant points that are possible in higher dimensions. On two dimensions, it is simple to draw to three equidistant points in the form of a triangle. But this same example is impossible to draw along a line, in one dimension. Similarly, in three dimensions it is possible to have 4 equidistant points in the form of a triangular pyramid, but is impossible to represent these same points accurately on paper. There are no physical models that include 4 or more dimensions, so we must instead reduce the dimensions of data to an amount that we can interpret. We focused on preserving more local structure by augmenting the cost function of an existing algorithm to focus and a list of given points.

By augmenting the cost function of the algorithm, we also need to re-derive the gradient that is used to implement the algorithm. This brings some human error, though this can be mitigated with new methods of automatic differentiation. We still need to have some form of checking the gradient, but automatic differentiation would help the process of including this method across several different algorithms. This form of differentiation is also uninterpretable, so manual differentiation would still be needed to represent the gradient in mathematical notation and present it. Differentiation was one of the major challenges when developing this method of adding focus to an algorithm. Other challenges included the GAPS methods taking significantly longer than the algorithm to calculate the metrics. LAPS would generally take around half the time of the algorithm to calculate all the scores while the GAPS methods would take around half an hour when calculating 30 neighbors and 5 samples.

With many algorithms, we also need a form of quantifying how well an algorithm is performing with respect to others. Each algorithm has a unique cost function, and while they are all minimized in every algorithm, they cannot be directly compared. Explained thoroughly later, metrics for the local and global preservation of structure were defined in another paper and used here. We expect that adding focus to our algorithm will increase its ability to preserve local structure without sacrificing much global structure.

2. Problem Statement

The goal is to demonstrate the effects of augmenting an algorithm to focus on the preservation of local structure between given points of interest. Given a set of N high-dimensional objects $x_1, x_2, ..., xN$ and a subset of N, which contains the points of interest (POI), compute the following probabilities $p_{j|i}$ for all objects in N, and $q_{j|i}$ for their respective counterparts in the low dimensional embedding:

$$p_{j|i} = \frac{exp(-||-x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||-x_i - x_k||^2/2\sigma_i^2)} \tag{2.1}$$

We also implemented the binary search for the sigma in (2.1) that produces a fixed perplexity P_i that is specified by the user. The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)} \tag{2.2}$$

where $H(P_i)$ is the Shannon entropy of P_i measured in bits

$$H(P_i) = -\sum P_{j|i} log_2 p_{j|i} \tag{2.3}$$

The perplexity can be interpreted as a smooth measure of the effective number of neighbors. And the typical values range from 5 to 50.

$$q_{j|i} = \frac{exp(-||-y_i - y_j||^2}{\sum_{k \neq i} exp(-||-y_i - y_k||^2)} \tag{2.4}$$

Minimize the difference between both probability distributions according to the new cost function:

$$\begin{aligned} C = \lambda(\sum_i \sum_{j \in POI} p_{j|i} log \frac{p_{j|i}}{q_{j|i}}) + \\ \lambda(\sum_{i \in POI} \sum_{j \notin POI} p_{j|i} log \frac{p_{j|i}}{q_{j|i}}) + \\ (\sum_{i \notin POI} \sum_{j \notin POI} p_{j|i} log \frac{p_{j|i}}{q_{j|i}}) \end{aligned} \tag{2.5}$$

Finally return the embedding of the set in a lower dimension space which should have a focus on the neighborhood of the given POI. The resulting embedding should show an improvement in the local preservation of structure compared to the original algorithm. The metrics for the embedding are defined below.

3. Literature Review

The Stochastic Neighbor Embedding (SNE) algorithm established by Geoffrey Hinton and Sam Roweis is the foundation for the method established in this paper. By using asymmetric probabilities that a point i would pick point j as its neighbor, the differences are minimized in both the low and high dimension space Hinton & Roweis (2002). The cost function is defined with the Kullback-Leibler divergence and all points are considered equally whereas ours will focus on a certain list of points of interest.
It is difficult to compare different dimension reduction algorithms without first establishing a metric between all of them. The Local Approximation of Preserved Structure (LAPS) and Global Approximation of Projection Space (GAPS) metrics defined by Aindrila Ghost, et al, provides a way to grade the embedding produced by these algorithms. The LAPS metric searches the neighborhood around a singular point and determines if similar neighbors and factors are near the point in both low and high dimensions. The discrepancies are added and show the overall discrepancy around a given point. The GAPS metric similarly searches the neighborhoods, but across a list of points and also returns a value for the amount of discrepancy in the global structure of the embedding Ghosh et al. (2022).

The binary search described in SNE and later implemented in van der Maaten and Hinton's new t-distributed Stochastic Neighbor Embedding (tSNE) is also used here. The variance σ_i may not be optimal for all points in the data set. As described in their paper van der Maaten & Hinton (2008), a smaller value is used for dense regions while larger values are used for sparse regions.

4. Methodology

As mentioned before, we based our algorithm off of Hinton and Roweis' SNE algorithm; minimizing the difference between two sets of probabilities for both low and high dimension space. After separating the terms into summations that either include or don't include the points of interest, we get the equation (2.3). Whose gradient when y_i is a point of interest is as follows,

$$\begin{aligned} &i \in POI \\ &\frac{\partial C}{\partial y_i} = 2\lambda \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j) \\ &+ (\lambda - 1) \sum_{\substack{i' \neq i \\ i' \in POI}} \sum_{\substack{j' \in POI \\ j' \neq i \\ j' \neq i'}} p_{j'|i'}(-2(y_{i'} - y_i) q(i|i')) \end{aligned} \tag{4.6}$$

and when y_i is not a point of interest,

$$\begin{aligned} &i \notin POI \\ &\frac{\partial C}{\partial y_i} = 2\lambda \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j) \\ &\quad + (\lambda - 1) \sum_{j \neq i} 2(y_i - y_j) q_{j|i} \sum_{\substack{j \in POI \\ j \neq i}} p_{j|i} \\ &\quad + (\lambda - 1) \sum_{j \notin POI} 2(y_j - y_i) p_{i|j}(1 - qi|j) \\ &\quad + (\lambda - 1) \sum_{\substack{i' \neq i \\ i' \notin POI}} \sum_{\substack{j' \notin POI \\ j' \neq i \\ j' \neq i'}} p_{j'|i'} - 2(y_{i'} - y_i) q_{i|i'} \\ &\quad - (\lambda - 1) \sum_{\substack{j \notin POI \\ j \neq i}} 2 p_{j|i}(y_i - y_j) \end{aligned} \tag{4.7}$$

Unlike the original gradient, we no longer have a short and concise gradient as before. For the purposes of the experiments, the point of interest are randomly gathered from the set of all points and stored in a list. However, if the user chooses, they may choose to focus on specific points of interest and pass their own list of indices. Initially the implementation was in python with little to no extra libraries, but the slow speed led to a parallel version that reduced the time required by a factor nearly equal to the amount of cores used. Later a PyTorch version was developed which runs significantly faster and would also allow for quick implementation of automatic differentiation as the project expands to different algorithms to modify. We

would then run our metrics and determine whether this method of focusing on POI would increase the performance of an existing algorithm. Even with a PyTorch implementation, the Breast Cancer data set running on Google Colab's A100 Tensor Core GPU, it takes approximately 4 minutes to calculate the low dimension embedding.

The embedding for all of the data sets are obtained using the predefined perplexities of 10, 20, 30, 40, and 50 are used for the algorithms that allow for perplexity/neighbors while the rest of the algorithms are evaluated at 30. The LAPS method is already defined and the only parameters that are changed are the ones mentioned before. The same can be said for the GAPS method, with the only difference that an additional parameter is passed, which is a list of the indices of the points of interest. The small budget used in the GAPS methods can be incremented for accuracy, though it was left as default due to resource constraints. The LAPS score is calculated for each of the points in the list of POI and then are averaged for each of the data sets. Currently each metric is tested once per embedding at each of the embedding.

We also included a few other algorithms to compare against, namely UMAP, PCA, KernelPCA, Isomap, and MDS. While the augmentation of the cost function shouldn't radically change the capacity of the original algorithm, it is good to gauge an idea of how well it compares to other algorithms before and after. UMAP is one of the stronger algorithms and is a strong algorithm to augment next after checking the results of this test.

5. Experimental Results

We found and started using some of the same data sets used in the LAPS and GAPS paper. The breast cancer data set is self explanatory, it includes 569 samples each with 30 features describing the type of cancer found and is classified as either malignant or benign. Our credit card data set includes 10,000 samples with 23 dimensions that include the demographics of a user, their amount of their credit, the payments they've given, and are labeled whether they will have to default on their payment the following month. Our MAGIC data set also includes 10,000 samples and 10 dimensions that describe the properties of signals to determine whether the sample is a valid signal or background noise. Wine quality contains 3918 samples and 11 dimensions describing physical attributes of a wine and are classified on different levels of quality. The spam and sentiment140 data set are text data containing sentences for their respective topic. The spam data set has 5572 samples, are labeled on whether the message is spam and after preprocessing are given 100 dimensions through a word to vector algorithm, though this is subject to change. Our sentiment data set includes 10,000 samples and are classified as either a positive or negative emotion, and similarly was given 100 dimensions through the word to vector algorithm.

The following are tables describing the metrics on our algorithm and baselines. Any row with a number in it's name is an algorithm that allows for the use of perplexity or neighbors. They are the table of results for the LAPS, GAPS and KNN metrics respectively.

LAPS Scores

	BC	CC	Magic	WQ	Spam	Sentiment
fSNE-10	0.72493	0.76336	0.8042	0.78948	0.68732	0.76998
fSNE-20	0.74765	0.80132	0.83583	0.82266	0.72533	0.79697
fSNE-30	0.7507	0.80341	0.84593	0.83105	0.72901	0.80154
fSNE-40	0.74238	0.80973	0.84909	0.83132	0.73072	0.80052
fSNE-50	0.7358	0.79957	0.83997	0.83078	0.72977	0.79484
SNE-10	0.74285	0.77414	0.80643	0.79043	0.69202	0.78021
SNE-20	0.75786	0.81007	0.8372	0.82925	0.73256	0.80809
SNE-30	0.75553	0.80807	0.84648	0.82963	0.7387	0.80675
SNE-40	0.74669	0.80493	0.84404	0.83153	0.73317	0.8032
SNE-50	0.73702	0.80751	0.84343	0.83219	0.73204	0.799
UMAP-10	0.69756	0.75029	0.77155	0.71753	0.6777	0.73744
UMAP-20	0.73027	0.78983	0.80995	0.77937	0.71492	0.7777
UMAP-30	0.73973	0.80023	0.8186	0.79682	0.72915	0.79004
UMAP-40	0.73295	0.79902	0.82408	0.8022	0.73232	0.79125
UMAP-50	0.72479	0.79921	0.82487	0.80483	0.7309	0.78928
PCA	0.76984	0.8462	0.88479	0.85862	0.75925	0.8235
KernelPCA	0.76984	0.8462	0.88434	0.8574	0.7598	0.8235
Isomap	0.77327	0.83182	0.87959	0.84996	0.76436	0.81898
MDS	0.77339	0.83973	0.88374	0.86462	0.7607	0.83

TABLE I

GAPS Scores

	BC	CC	Magic	WQ	Spam	Sentiment
fSNE-10	0.02377	0.02032	0.01858	0.01942	0.02825	0.02364
fSNE-20	0.02266	0.01983	0.01814	0.01903	0.02719	0.02305
fSNE-30	0.0214	0.01976	0.0177	0.01858	0.02578	0.02266
fSNE-40	0.02103	0.01956	0.0178	0.01827	0.02508	0.02288
fSNE-50	0.02108	0.01926	0.0178	0.0183	0.02499	0.02252
SNE-10	0.02255	0.02003	0.01839	0.01936	0.02805	0.02279
SNE-20	0.02134	0.01961	0.01808	0.01852	0.02545	0.02279
SNE-30	0.02092	0.01981	0.01795	0.01874	0.02478	0.02208
SNE-40	0.02042	0.01957	0.01769	0.01821	0.02465	0.02225
SNE-50	0.02073	0.01908	0.01788	0.01814	0.02444	0.02154
UMAP-10	0.02617	0.02047	0.01913	0.02183	0.02908	0.02502
UMAP-20	0.02476	0.01991	0.01875	0.02032	0.02681	0.02388
UMAP-30	0.02385	0.0194	0.01899	0.02008	0.02728	0.02391
UMAP-40	0.02312	0.0197	0.01848	0.01895	0.02635	0.02396
UMAP-50	0.02324	0.01936	0.01862	0.01957	0.02605	0.02294
PCA	0.02034	0.0186	0.01702	0.01801	0.02301	0.02203
KernelPCA	0.02034	0.0186	0.01702	0.01807	0.02304	0.02203
Isomap	0.01993	0.01881	0.01744	0.01801	0.02515	0.0223
MDS	0.0196	0.01871	0.01735	0.01788	0.02406	0.0221

TABLE II

KNN Scores

	BC	CC	Magic	WQ	Spam	Sentiment
fSNE-10	0.95	0.93	0.83	0.66	0.95	0.71
fSNE-20	0.97	0.95	0.83	0.72	0.94	0.77
fSNE-30	0.98	0.95	0.83	0.66	0.95	0.7
fSNE-40	0.98	0.95	0.82	0.76	0.94	0.83
fSNE-50	0.98	0.95	0.83	0.67	0.94	0.74
SNE-10	0.96	0.97	0.78	0.73	0.94	0.74
SNE-20	0.97	0.95	0.8	0.64	0.95	0.76
SNE-30	0.97	0.93	0.83	0.7	0.94	0.73
SNE-40	0.98	0.94	0.82	0.72	0.94	0.78
SNE-50	0.98	0.92	0.84	0.69	0.94	0.77
UMAP-10	0.99	0.94	0.83	0.74	0.97	0.72
UMAP-20	0.96	0.95	0.88	0.69	0.97	0.73
UMAP-30	0.97	0.95	0.86	0.73	0.97	0.73
UMAP-40	0.99	0.94	0.87	0.72	0.96	0.77
UMAP-50	0.99	0.92	0.87	0.76	0.96	0.78
PCA	0.98	0.92	0.79	0.55	0.99	0.71
KernelPCA	0.98	0.93	0.79	0.55	0.99	0.71
Isomap	0.98	0.93	0.76	0.66	0.96	0.65
MDS	0.94	0.91	0.86	0.63	0.96	0.64

TABLE III

The best scores for each of the metrics is highlighted in gold, seconds highest; silver, and third highest; bronze. The following are the results plotted by data set.

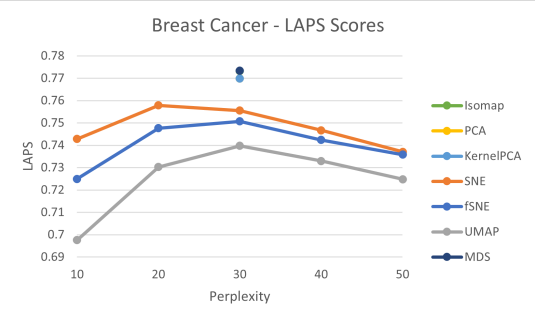

Fig. 1

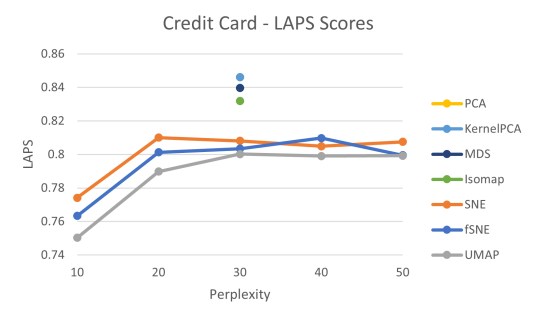

Fig. 2

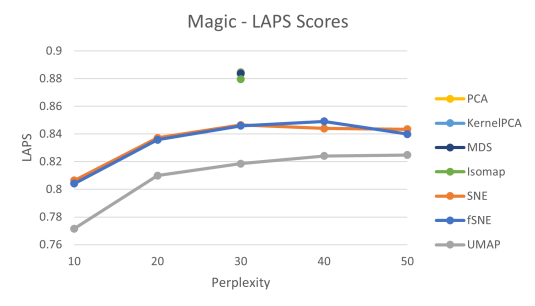

Fig. 3

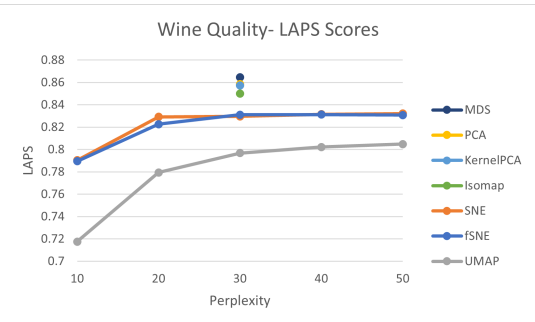

Fig. 4

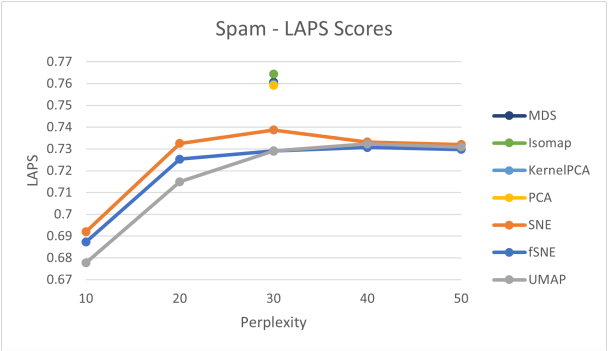

Fig. 5

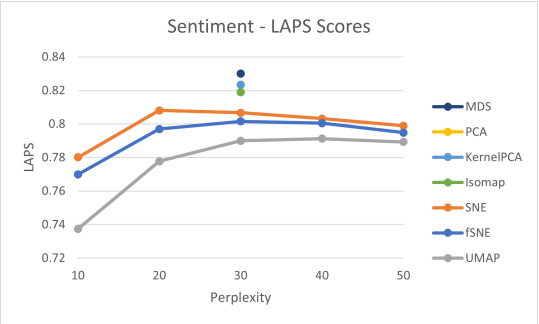

Fig. 6

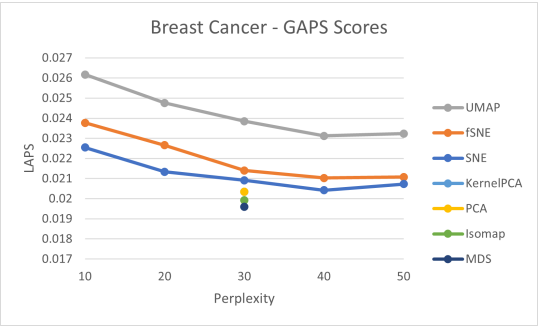

Fig. 7

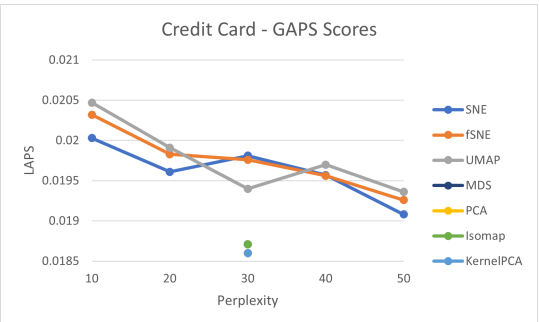

Fig. 8

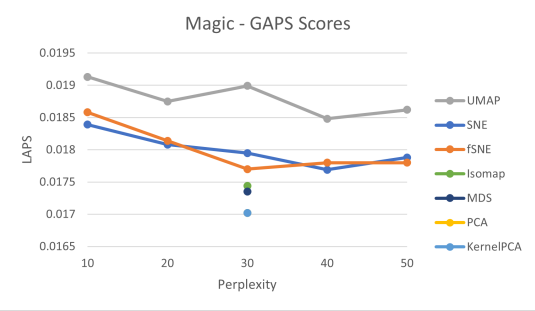

Fig. 9

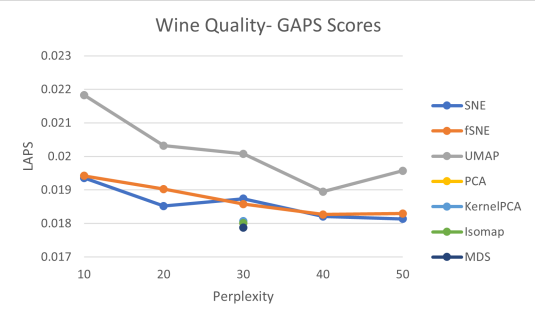

Fig. 10

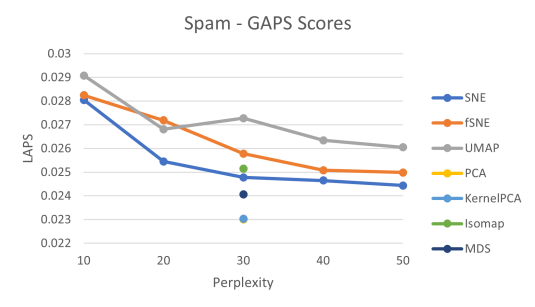

Fig. 11

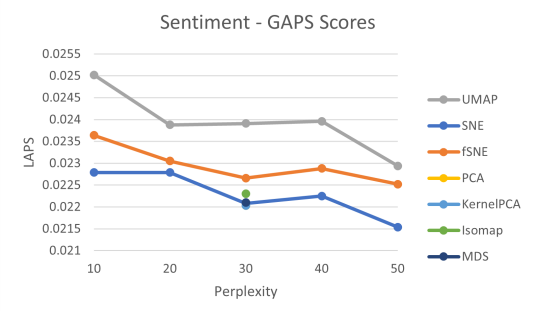

Fig. 12

A general trend can be seen where fSNE performs better than SNE, albeit a small change.

6. Conclusions and Future Works

As shown by the tables and figures, our current implementation of fSNE does improve the results of SNE in the LAPS metrics with only a small decrease in performance in the GAPS metrics. In other words, adding a focus to SNE did improve the algorithms ability to focus on the local structure of a list of given points without sacrificing too much global structure. The inclusion of the UMAP algorithm also displays that this change is not drastic enough to completely change the capacity of the algorithm, only augment it. Adding this ”focus” to more capable algorithms could lead to a stronger algorithm overall. More tests need to be done to accurately describe the amount of change produced by this ”focus”, mainly due to the non-deterministic nature of most of these algorithms. Additional metrics could also be used and added for the same reason.

7. References

Ghosh, A., Nashaat, M., Miller, J., & Quader, S. (2022, may). Interpretation of structural preservation in low-dimensional embeddings. *IEEE Transactions on Knowledge and Data Engineering*, *34*(05), 2227-2240. doi: 10.1109/TKDE.2020.3005878

Hinton, G. E., & Roweis, S. (2002). Stochastic neighbor embedding. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems* (Vol. 15). MIT Press. Retrieved from `https://proceedings.neurips.cc/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf`

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605. Retrieved from `http://www.jmlr.org/papers/v9/vandermaaten08a.html`