Making low-dimensional embeddings more informative: dtSNE and JEDI algorithms

Aleksandr Odnakov [1] **Alina Bogdanova** [1] **Emil Alkin** [1] **Ilia Khristoforov** [1] **Kira Kuznetsova** [1]

Abstract

This report presents a study of two unsupervised machine learning algorithms - dtSNE and JEDI — in comparison to the popular algorithms tSNE and LLE. The paper begins by exploring the theoretical premises of dtSNE and JEDI, followed by the implementation and experimenting. Both algorithms are tested on 5 synthetic datasets, MNIST dataset and evaluated by 4 metrics. It is clear that dtSNE method consistently provides better results for local densities, while global densities are slightly decreasing. JEDI also shows great results at representing complex data.

Github repo: dtSNE and JEDI

1. Introduction

Low-dimensional embeddings are great tools in data science. They allow to explore the structure and special relationships in data, such as clusters, outliers, and general trends. Application of such methods is usually the first step in exploration of data.

There are lots of such methods, but the most interesting are much more complex, because usually data is non-linear. New embeddings are trying to reconstruct local structures of data in exchange of global distances. It allows to find the most interesting relations for the researcher. The most widely used method is tSNE. tSNE is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales. It is done by implementing Student t-distribution.

While this method is effective for visualization and analysis, this method does distort local and inter-cluster distances. In other words, this method loses data structure properties that can be important. The other problem with tSNE method is inability to factor out prior knowledge. Basically, when we use this method, we usually get the structure which is already known by science. So, the ability to see the relations beyond the ones we know can open new abilities for scientists.

Two stated problems can be solved by two improvements of usual tSNE – dtSNE and JEDI. dtSNE takes local densities in respect, while JEDI takes backround knowledge in respect in the form of pairwise distances between samples.

In summary, main tasks were solved by the team are:

- Implementation of dtSNE and JEDI methods as a Python library;
- Implementation of different metrics for evaluation of these methods;
- Creating synthetic datasets for testing proposed methods;
- Choosing LLE, tSNE and UMAP as baselines and comparing new methods to them;
- Analysis of acquired results.

2. Related work

Embeddings of high dimensional data into a low dimensional space, in particular to 2 or 3 dimensions, have in recent years become an essential tool of unsupervised analysis of high dimensional data in modern science.
Below is a description of the main methods used for this task.

2.1. Locally Linear Embedding

Locally Linear Embedding (LLE) is a method of Non Linear Dimensionality reduction proposed by Sam T. Roweis and Lawrence K. Saul in 2000 in their paper (Roweis & Saul, 2000a).
This algorithm works as follows:

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Aleksandr Odnakov <Aleksandr.Odnakov@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

1. Finding the K nearest neighbours.
   The first step is find K nearest neighbours, K is the only hyperparameter of the algorithm.

2. Finding weights.
   Each point $\mathbf{x}_i$ is presented as a linear combination of its neighbors.
   $$\mathbf{x}_i = \sum_j w_{ij}\mathbf{x}_j$$
   Then the weights are the optimizer of the cost function:
   $$L(w_{ij}, j = 1 \dots n) = \|\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j\|^2 - \alpha w_{ij}^2$$

3. Final reconstruction.
   After finding the weight matrix $\mathbf{W}$, it is used to represent the geometric information in $\mathbf{X}$ so that $\mathbf{X}$ does not appear in the final reconstruction step. The goal of this step is to find matrix $\mathbf{Y}$ which minimizes:
   $$\Phi(\mathbf{Y}) = \sum_i \|\mathbf{y}_i - \sum_{i \neq j} w_{ij}\mathbf{y}_j\|^2$$

2.2. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2003) is a machine learning algorithm used for dimensionality reduction, visualization, and clustering of high-dimensional datasets. The algorithm maps each high-dimensional data point to a two- or three-dimensional point while preserving the similarity relationships between them.

The basic idea behind SNE is to convert the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The algorithm then attempts to optimize a cost function that measures the similarity between the high-dimensional and low-dimensional data points.

Here's how SNE works:

1. SNE starts by computing the similarity between data points in the high-dimensional space. It measures the similarity between points using a Gaussian kernel that decays with the distance between points. The kernel function is defined as:
   $$p_{i|j} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2/2\sigma_i^2\right)}$$
   where $p_{i|j}$ represents the probability of choosing point i as a neighbor of point j, σ is a parameter that controls the spread of the Gaussian kernel, and $||.||$ represents the Euclidean distance between two points.

   In order to find all σ_i, we need a hyperparameter Perplexity, which can be interpreted as a smooth measure of the effective number of neighbors in high-dimensional space. Perplexity is defined as
   $$Perp(P_i) = 2^{H(P_i)}$$
   where $H(P_i)$ is a Shannon entropy of P_i:
   $$H(P_i) = -\sum_i p_{j|i} \log_2 p_{j|i}$$
   With pre-defined perplexity value, binary search over σ_i is used to find corresponding standard deviations, since $Perp(P_i)$ is monotounous function with respect σ_i.

2. Next, SNE computes the similarity between low-dimensional points, which are initially assigned random coordinates. It measures the similarity between points using a similar Gaussian kernel, but with a fixed variance of $1/2$:
   $$q_{i|j} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{k \neq i} \exp\left(-||y_i - y_k||^2\right)}$$
   where $q_{i|j}$ represents the probability of choosing point i as a neighbor of point j in the low-dimensional space, y_i and y_j represent the coordinates of the low-dimensional points.

3. SNE then tries to optimize a cost function that measures the difference between the high-dimensional and low-dimensional similarities. The cost function is defined as the Kullback-Leibler divergence between the two probability distributions:
   $$C = KL(P||Q) = \sum_i \sum_j p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}$$
   The algorithm then uses gradient descent to minimize the cost function with respect to the coordinates of the low-dimensional points.

4. The optimization process is repeated for a number of iterations until the low-dimensional representation of the data converges to a stable configuration. The gradient that is used in the process of gradient descent can be expressed via this formula:
   $$\frac{\partial C}{\partial y_i} = 2\sum_j (p_{i|j} - q_{i|j} + p_{j|i} - q_{j|i})(y_i - y_j)$$
   In order to speed up the optimization and to avoid poor local minima, a relatively large momentum term is

added to the gradient, and the gradient update with a momentum term is given by:

$$Y^{(t)} = Y^{(t-1)} + \gamma\frac{\partial C}{\partial Y} + \delta\left(Y^{(t-1)} - Y^{(t-2)}\right))$$

2.3. Symmetric SNE

First modification to original SNE is called Symmetric SNE (van der Maaten & Hinton, 2008). First step to achieve it, is to define pairwise similarities in the high-dimensional space p_{ij}:

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

and in low-dimensional space q_{ij}:

$$q_{ij} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{k\neq l}\exp\left(-||y_k - y_l||^2\right)}$$

So that Kullback-Leibler divergence becomes:

$$C = KL(P||Q) = \sum_i\sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}}$$

And its derivative becomes:

$$\frac{\partial C}{\partial y_i} = 4\sum_j(p_{ij} - q_{ij})(y_i - y_j)$$

Main advantages of symmetric SNE is faster computation time, simpler derivative and author argues that this version is just as good as original one.

2.4. t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (tSNE) is further extension of Symmetric SNE that uses a Student-t distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space. It helps to preserv global structure better, it places higher weight on distant points due to heavier tails of t-distrubution than SNE does, which can lead to better preservation of the overall shape of the data.

Changing to t-distribution in lower-dimensional embedding, q_{ij} becomes:

$$q_{i|j} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_{k\neq l}\left(1 + ||y_k - y_l||^2\right)^{-1}}$$

However, this change complicated gradient of cost function:

$$\frac{\partial C}{\partial y_i} = 4\sum_j(p_{ij} - q_{ij})(y_i - y_j)\left(1 + ||y_i - y_j||^2\right)^{-1}$$

As a result of this change, tSNE puts emphasis on modeling dissimilar datapoints by means of large pairwise distances, and modeling similar datapoints by means of small pairwise distance.

3. Algorithms and Models

3.1. Density preserving tSNE

Density preserving tSNE (dtSNE) is a one more extension of tSNE which aims to target and preserve local densities of the points (Fischer et al., 2023). Authors provide theoretical proof of why their modification indeed manages to reflect density of point's neighborhood.

In order to achieve that, they propose scaling factor for low-dimensional distances for each pair of points i, j:

$$\gamma_{ij} = \frac{\left((\sigma_i + \sigma_j)^2\right)^{-1}}{\max_{k,l}\left((\sigma_k + \sigma_l)^2\right)^{-1}}$$

And then further modifying corresponding low-dimensional similarities:

$$q_{i|j} = \frac{\left(1 + \gamma_{ij}||y_i - y_j||^2\right)^{-1}}{\sum_{k\neq l}\left(1 + \gamma_{kl}||y_k - y_l||^2\right)^{-1}}$$

As we see, scaling factor is symmetric for each pair of points in low-dimensional embedding. To reflect this symmetry in high-dimensional original space, authors symmetrize σ-s:

$$\sigma_{ij}^2 = \left(\frac{\sigma_i + \sigma_j}{2}\right)$$

This also helps with theoretical analysis.

After this change to σ-s, similarities in high-dimensional space will look like this:

$$p_{i|j} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_{ij}^2\right)}{\sum_{k\neq i}\exp\left(-||x_i - x_k||^2/2\sigma_{ik}^2\right)}$$

and then they also symmetrized like in tSNE:

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

Here, algorithm 1 represents final dtSNE algorithm from the original paper.

3.2. JEDI

For some specific applications it is useful to take user's input and/or domain knowledge into account. To do so we can use JEDI algorithm proposed by (Heiter et al., 2021). This algorithm was developed to obtain low-dimensional embeddings while factoring out prior knowledge.

Given a set of n samples X from a high dimensional space, our goal is to find a low dimensional representation Y that captures the local structure in X while factoring out prior knowledge Z about the samples. Here, we consider both high dimensional data X and prior Z to be given as distance

Algorithm 1 dtSNE

input Data X, perplexity k, iterations T, learning rate μ, momentum δ
output Embedding Y

compute P // Use symmetrized σ_{ij}
compute γ_{ij} // Scaling factor γ_{ij}
$Y^{(0)} \leftarrow PCA(X, 2)$ // Initial embedding
$Y^{(0)} \leftarrow 0.001 \frac{Y^{(0)}}{\text{std}\left(Y^{(0)}\right)}$
for $t = 1 \dots T$ **do**
  compute Q // Use scaling γ_{ij}
  compute $\frac{\partial KL(P||Q)}{\partial Y}$
  $Y^{(t)} \leftarrow Y^{(t-1)} + \mu \frac{\partial C}{\partial Y} + \delta \left(Y^{(t-1)} - Y^{(t-2)}\right))$
end for
return $Y^{(T)}$

Algorithm 2 JEDI

input Data X, prior knowledge matrix Z, embedding dimension *no_dims*, PCA dimension *initial_dims*, level of symmetry α, level of skewness β, perplexity k, iterations T, learning rate μ, momentum δ
output Embedding Y

$X \leftarrow PCA(X, \textit{initial\_dims})$
compute D_X
compute D_Z
$Y^{(0)} \leftarrow \text{Normal}(\textit{no\_dims})$ // Initial embedding
for $t = 1 \dots T$ **do**
  compute Q
  compute $\frac{\partial C}{\partial Y} = \frac{\partial\left(KL(P||Q) - JS_\alpha^\beta(P'||Q)\right)}{\partial Y}$
  $Y^{(t)} \leftarrow Y^{(t-1)} + \mu \frac{\partial C}{\partial Y} + \delta \left(Y^{(t-1)} - Y^{(t-2)}\right))$
end for
return $Y^{(T)}$

matrices, thus allowing for data from typical spaces such as Euclidean, but also images, up to unstructured data such as texts or graphs, for which distance matrices can be specified using a kernel.

$$D^X \approx D^Y \not\approx D^Z$$

We could formally define this as a multi-objective problem composed of a minimization over the difference between D^X and D^Y and a maximization of the difference between D^Y and D^Z. We use the same approach as in the tSNE algorithm, but replace the cost function by a function that takes into account the matrix Z. So we change the tSNE algorithm as follows. Suppose that X after applying PCA has a distribution P, embeddings have a distribution Q and the prior knowledge matrix Z defines a distribution P'. As a cost function we use not just the Kullback-Leibler divergence between P and Q which penalizes the difference between them. Instead we use the sum of this divergence and Parameterized Jensen Shannon Divergence between P' and Q which penalizes the similarity of P' and Q. For two probability distributions P' and Q we define the parameterized Jensen-Shannon divergence as

$$\begin{aligned}\text{JS}_\beta^\alpha\left(P' \| Q\right) = \alpha D_{\text{KL}}\left(P' \| \beta Q + (1-\beta)P'\right) + \\ + (1-\alpha) D_{\text{KL}}\left(Q \| \beta P' + (1-\beta)Q\right)\end{aligned}$$

where $0 \leq \alpha \leq 1$ determines the level of symmetry and $0 < \beta < 1$ determines the level of skewness.

$$C = KL(P||Q) - \text{JS}_\beta^\alpha\left(P' \| Q\right)$$

4. Experiments and Results

4.1. Datasets

To evaluate DTSNE and JEDI, we compare on both synthetic as well as real world data against the state-of the-art in unsupervised low-dimensional embedding approaches UMAP (McInnes et al., 2018), TSNE (van der Maaten & Hinton, 2008), LLE (Roweis & Saul, 2000b).

We consider benchmarks of gaussian and uniform mixtures (Fischer et al., 2023), the (vectorized) MNIST dataset of handwritten digits (Lecun et al., 1998).

The following 5 synthetic datasets we take from the paper (Fischer et al., 2023):

2D We generated two 2-dimensional dataset with 3 Gaussian clusters each. The Gaussian clusters had unit variance and were centered at $(10, 0), (0, 15)$, and $(-10, 0)$, respectively. For the first dataset, we drew 300 points from each Gaussian and scaled the spread of the clusters by 1, 2, 4 (i.e., multiply the centered data by this number), respectively. For the second dataset we drew 100, 200, 500, samples from the Gaussians, respectively, keeping the scale the same across clusters.

G3-S For this dataset we generated 3 Gaussian clusters living in 50 dimensions, each cluster distribution ci with mean drawn from $U(0, 50)$ (each dimension iid from this uniform) and unit variance. We then draw 200, 400, 600 points from c_1, c_2, c_3, respectively, and scale the spread of the cluster by 2 (i.e., multiply the centered data by 2).

G3-D For this dataset we generated 3 Gaussian clusters living in 50 dimensions, each cluster distribution ci with mean drawn from U(0, 50) and unit variance. We then draw

300 points from each of the cluster distributions and scale the spread of the c_1, c_2, c_3 by 2, 4, 8, respectively.

G10-D To look at data that is not easily projectable, i.e., the inter-cluster distances can be correctly modeled in 2D, for this dataset we generated 10 Gaussian clusters living in 50 dimensions, each cluster distribution c_i again with mean drawn from $U(0, 50)$ and unit variance. We then draw 200 points from each of the cluster distributions and scale the spread of the $c_1, \ldots, c_{10}$ by $1, \ldots, 10$, respectively.

U5-D To look at a different distribution and higher dimensional data, we generated 10 Uniform clusters living in 150 dimensions, each cluster distribution c_i again with mean drawn from $U(0, 50)$ and unit variance. We then draw 200 points from each of the cluster distributions and scale the spread of the $c_1, \ldots, c_{10}$ by $1, \ldots, 10$, respectively.

For JEDI testing we also use the dataset used by authors in (Heiter et al., 2021). The data has 14 dimensions in total, where each sample belongs to one of 4 clusters (A1-A4) in dimension 1-8 and one of four clusters in dimension 9-12 (B1-B4). We first draw cluster centers from $\mathcal{N}(0, 2)$ for each of the eight clusters. Then, the feature values for each sample are generated in three steps as follows:

1. Pick a cluster a from A1-A4 with probability 0.1, 0.2, 0.3, or 0.4, respectively. Add Gaussian noise to the cluster center a with standard deviation 0.1, 0.2, 0.3, or 0.4, respectively. Noise is drawn and added for each dimension independently.

2. Pick a cluster a from B1-B4 with probability 0.1, 0.2, 0.3, or 0.4, respectively. Add Gaussian noise to the cluster center a with standard deviation 0.1, 0.2, 0.3, or 0.4, respectively. Noise is drawn and added for each dimension independently.

3. The remaining 2 dimensions of every sample are Gaussian noise from $N(0, 1)$.

4.2. Metrics

We use metrics considered in papers (Fischer et al., 2023) and (Heiter et al., 2021):

- Global reconstruction score, ρ

  ρ := Pearson correlation between high- and low-dimensional distances

- Local reconstruction score, ρ_{knn}

  ρ_{knn} := Pearson correlation of distances of each point to its $k = 100$ closest neighbours between high- and low-dimensional space.

- Relative density reconstruction score, ρ_r

  We take for each point i the radius r_i of the (smallest) ball enclosing its 100 neighbours, i.e., the distance to its 100th neighbour. We then take fractions of each pair of radii, $\frac{r_i}{r_j}$. Finally, ρ_r := Pearson correlation between them in high- and low-dimensional space

- Neighbourhood overlap score, NOS

  For two distance matrices D, D_0 and neighborhood size k, we define the neighbourhood overlap score (NOS) as $NOS(D, D_0, k) = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n} |\{\text{kNN of } i \text{ in } D\} \cap \{\text{kNN of } i \text{ in } D'\}|$.

Table 1. ρ, global structure

Method	**dtSNE**	LLE	tSNE	UMAP	**JEDI**
2D	.91	.85	.93	**.93**	.92
G3-s	.80	.94	.91	**.98**	.94
G3-d	.75	.95	.94	.95	**.96**
G10-d	.51	.21	**.73**	.59	.58
U5-d	.77	.51	.82	**.88**	.82
MNIST	**.59**	.30	.55	.41	.49

Table 2. ρ_{knn}, local structure

Dataset	**dtSNE**	LLE	tSNE	UMAP	**JEDI**
2D	**.90**	.45	.80	.78	.86
G3-s	**.64**	.04	.61	.54	.64
G3-d	**.66**	-.02	.19	.19	.20
G10-d	**.70**	.01	.19	.15	.20
U5-d	**.60**	-.03	.12	.12	.12
MNIST	**.85**	.48	.77	.54	.79

For JEDI we considered additional set of experiments. We had tested different hyperparameters, such as perplexity, α and β. At the figures 11-13 in Appendix C you can see visual representation of this dataset depending on these hyperparameters.

All conducted experiments you can find in our Github repository (see "dtSNE_and_JEDI_experiments.ipynb", "JEDI_experiments.ipynb" files).

The results of our experimenting are shown in Appendix C and at tables 1, 2 and 3.

5. Conclusions

During the work on the project it was possible to implement both algorithms - dtSNE and JEDI. Both algorithms were tested on the datasets described above and results were obtained describing the preservation of global and local data structure. As can be seen from the pictures in appendix C and from the tables, dtSNE is consistently much better

Table 3. ρ_r, relative density structure

Dataset	**dtSNE**	LLE	tSNE	UMAP	**JEDI**
2D	.90	.44	.93	.95	**.96**
G3-s	**.34**	-.20	.27	-.01	.32
G3-d	**.80**	-.12	.09	.10	.10
G10-d	**.88**	-.04	.07	-.01	.01
U5-d	**.82**	-.12	.04	.03	.04
MNIST	**.76**	.42	.57	.47	.61

than any of the other methods (table 1, 2 and 3). JEDI shows results comparable to the basic methods, but the data representation with JEDI is different from the other methods and can be used for additional findings about the data.

References

Fischer, J., Burkholz, R., and Vreeken, J. Preserving local densities in low-dimensional embeddings. *arXiv preprint arXiv:2301.13732*, 2023.

Heiter, E., Fischer, J., and Vreeken, J. Factoring out prior knowledge from low-dimensional embeddings. *arXiv preprint arXiv:2103.01828*, 2021.

Hinton, G. and Roweis, S. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15: 833–840, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.7959&rep=rep1&type=pdf.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 5500:2323–6, 2000a.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326, 2000b.

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

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Alexander Odnakov (20% of work)

- Implementing dtSNE method
- Github Repository creation
- dtSNE theory explanation for the report

Emil Alkin (20% of work)

- Coding functions for metrics evaluation
- Creating beautiful plots for experiments visualization
- Preparing 'Experiments and Results' section of this report
- Experimenting and preparing tables for this report

Alina Bogdanova (20% of work)

- JEDI implementation
- JEDI theory for the report
- synthetic data and experiments for JEDI

Kira Kuznetsova (20% of work)

- Datasets creation
- Papers review for the Related work part
- JEDI metrics implementation

Ilia Khristoforov (20% of work)

- Requirements checkup
- Presentation design
- Abstract, intro, and conclusions part of this report
- Appendices for this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: We used proposed metrics and datasets, but we programmed them by ourselves. tSNE code was used to create dtSNE. To sum up, it is around 15-20% of all the code.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: Pre-processing is not required.

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: No models were trained, so no splits were done.

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: Best hyperparameters were not considered, however, hyperparameter range for the JEDI algorithm was shown

9. The exact number of evaluation runs is included.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☐ Yes.
    ☐ No.
    ☑ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

- [x] Yes.
- [ ] No.
- [ ] Not applicable.

Students' comment: None

C. Figures

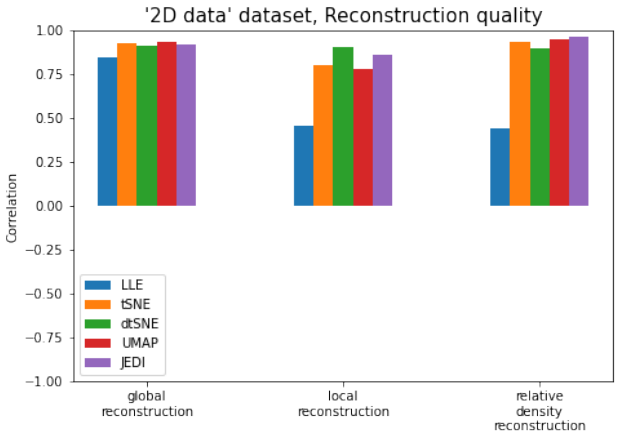

Figure 1. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the 2D DATA dataset.

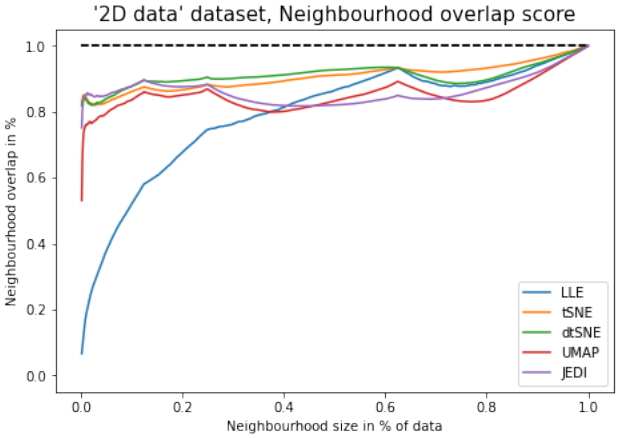

Figure 2. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the 2D DATA dataset.

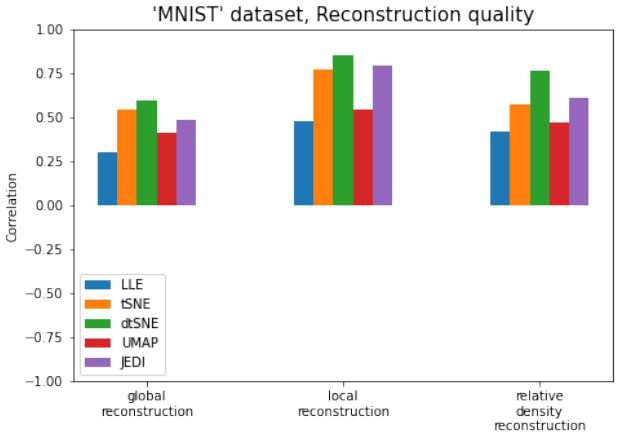

Figure 3. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the MNIST dataset.

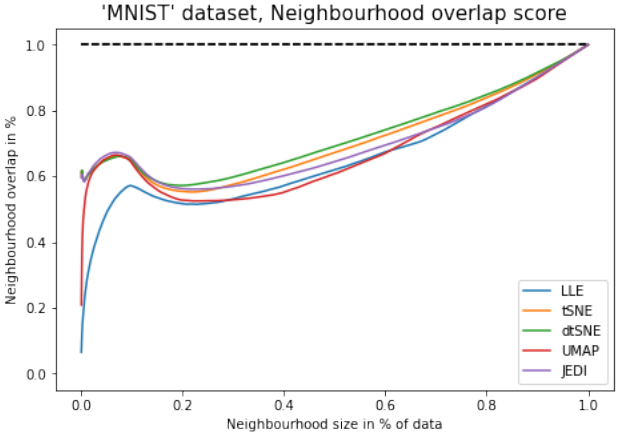

Figure 4. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the MNIST dataset.

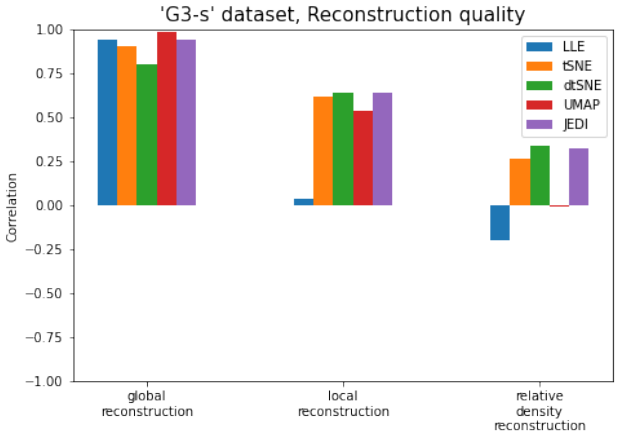

Figure 5. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G3-s dataset.

Figure 7. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G3-d dataset.

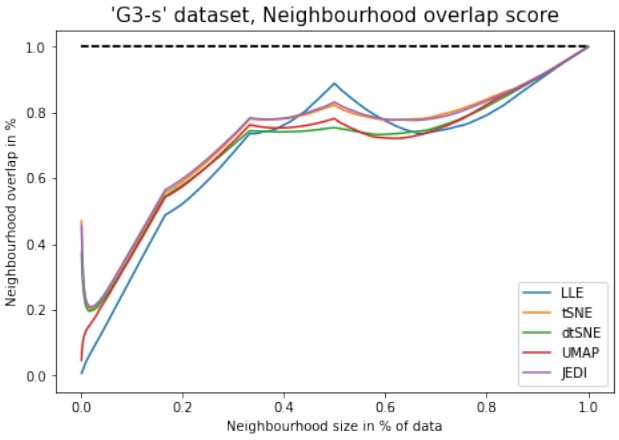

Figure 6. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G3-s dataset.

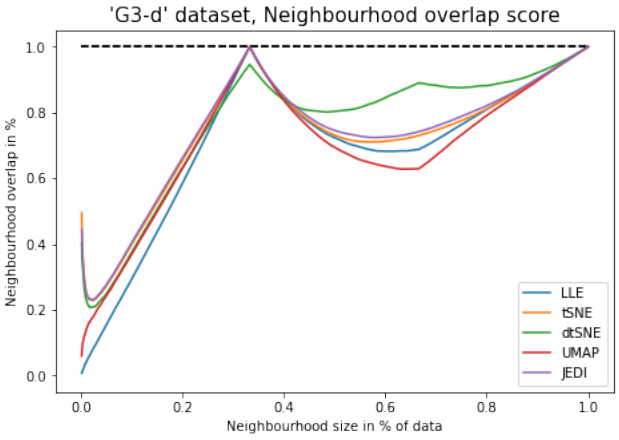

Figure 8. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G3-d dataset.

Figure 9. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G10-d dataset.

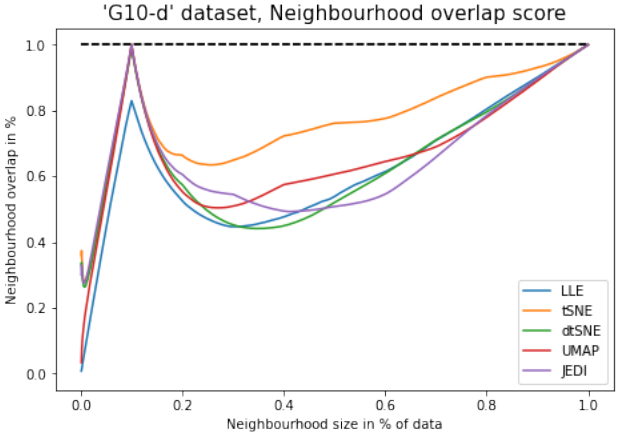

Figure 10. Comparison of ρ, ρ_{knn}, ρ_r metrics evaluated for different algorithms on the G10-d dataset.

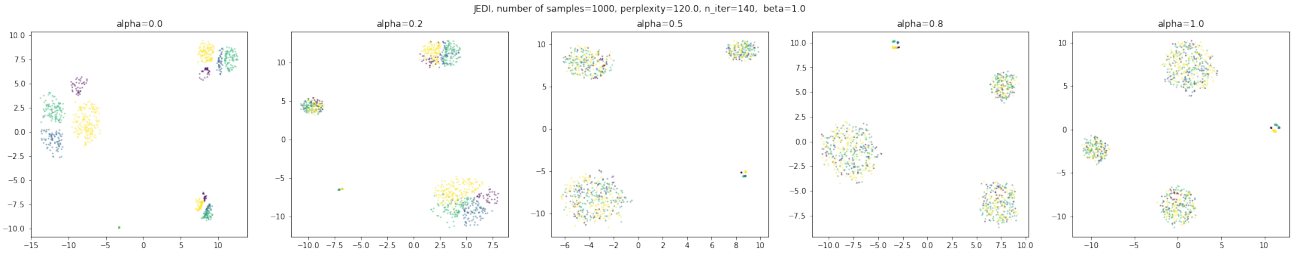

Figure 11. Comparison of JEDI output depending on α.

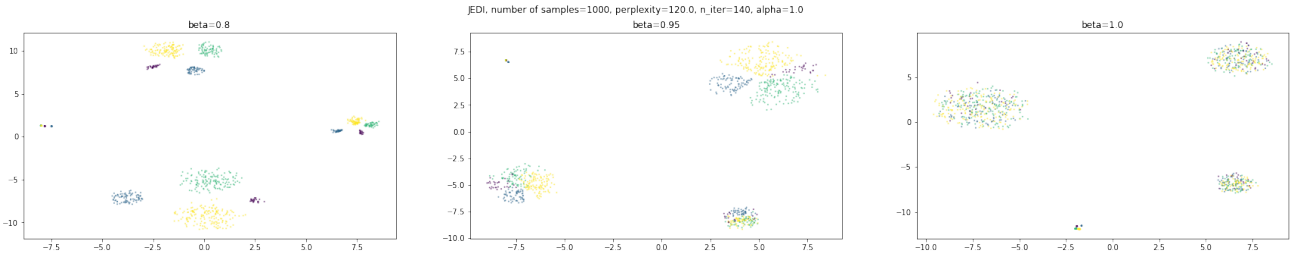

Figure 12. Comparison of JEDI output depending on β.

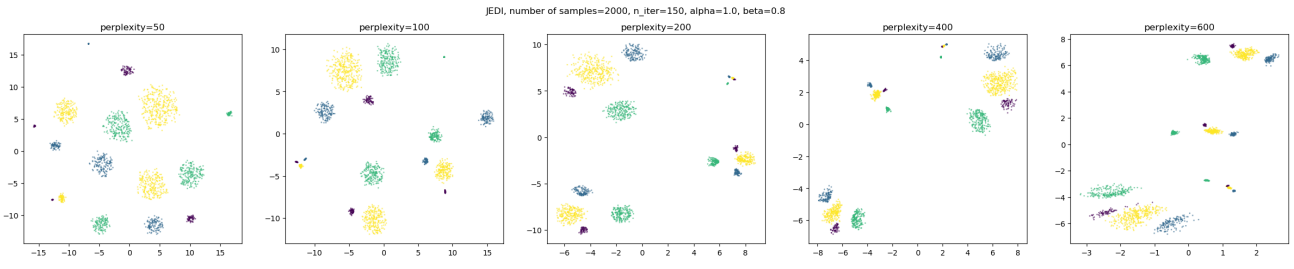

Figure 13. Comparison of JEDI output depending on perplexity.