Distance-based dimensionality reduction for big data literature review

Adrià Casanova Lloveras

February 17, 2025

Contents

1 `Rdimtools`: An R Package for Dimension Reduction and Intrinsic Dimension Estimation [16]

1.1 Abstract

Original: Discovering patterns of the complex high-dimensional data is one of the fundamental pillars of modern data science. Dimension reduction and intrinsic dimension estimation are two thematic programs that facilitate geometric characterization of the data. We present `Rdimtools`, an R package that supports 143 dimension reduction and manifold learning methods and 17 dimension estimation algorithms whose unprecedented extent makes multifaceted scrutiny of the data in one place easier. `Rdimtools` is distributed under the MIT license and is accessible from CRAN, GitHub, and its package website, all of which deliver instruction for installation, self-contained examples, and API documentation.

Apple Intelligence summary: `Rdimtools`, an R package, supports 160 dimension reduction and manifold learning methods, making data analysis easier. It is available on CRAN, GitHub, and its package website.

1.2 Key Points

- **R package**: that supports 143 dimension reduction and manifold learning methods and 17 dimension estimation algorithms.
- **Other libraries**: `drtoolbox` in MATLAB, `scikit-learn` in Python, a C++ template library `tapkee` with a known basis of popularity. In R, packages `dimRed`, `dyndimred`, `intrinsicDimension`.
- **Implementation**: mixture of R and C++ that are integrated by `Rcpp`. For numerical operations, `RcppArmadillo` is heavily used to take advantage of `Armadillo` C++ linear algebra library.
- **3 function families**: `do.{algorithm}`, `est.{algorithm}` and `aux.{algorithm}` for DR, IDE, and auxiliary functions.
- Downloaded 1013 times per month on average from CRAN.
- **Future plan**: support for out-of-memory execution in response to the increased needs for big data analysis.

1.3 Example R Code

```
# Documentation example:
# do.idmap (Interactive Document Map)

library(Rdimtools)

## load iris data
data(iris)
set.seed(100)
```

```r
subid = sample(1:150,50)
X = as.matrix(iris[subid,1:4])
lab = as.factor(iris[subid,5])
## let's compare with other methods
out1 <- do.pca(X, ndim=2)
out2 <- do.lda(X, ndim=2, label=lab)
out3 <- do.idmap(X, ndim=2, engine="NNP")
## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(out1$Y, pch=19, col=lab, main="PCA")
plot(out2$Y, pch=19, col=lab, main="LDA")
plot(out3$Y, pch=19, col=lab, main="IDMAP")
par(opar)
```

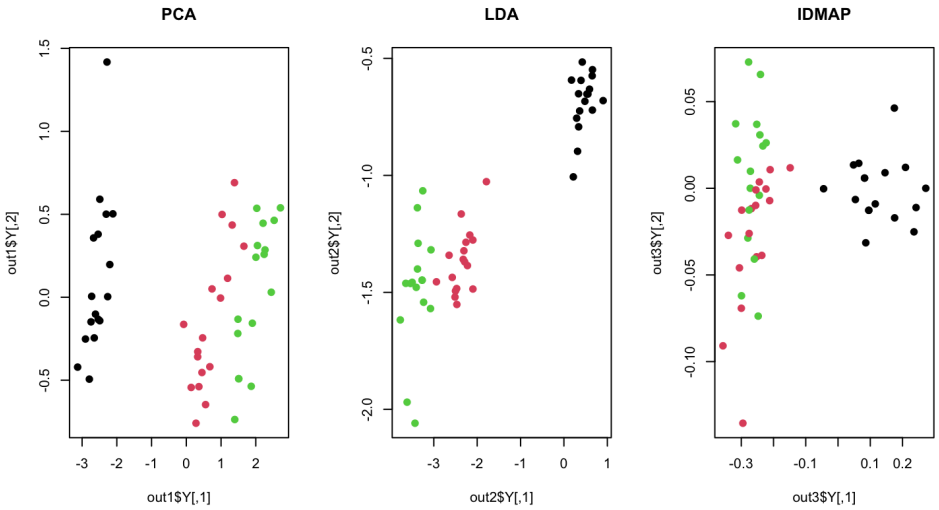

Figure 1: Rdimtools test output.

2 Global versus local methods in nonlinear dimensionality reduction [14]

2.1 Abstract

Original: Recently proposed algorithms for nonlinear dimensionality reduction fall broadly into two categories which have different advantages and disadvantages: global (Isomap), and local (Locally Linear Embedding, Laplacian Eigenmaps). We present two variants of Isomap which combine the advantages of the global approach with what have previously been exclusive advantages of local methods: computational sparsity and the ability to invert conformal maps.

Apple Intelligence summary: Two new Isomap variants are presented, combining global advantages with local computational sparsity and conformal map inversion.

2.2 Key Points

- **Introduction of LMDS** by applying it to Isomap (L-Isomap).
- ***Landmark points*** ($n << N$) reduce the complexity of computing:
  - the distances matrix with Dijkstra's algorithm with Fibonacci heaps (k = neighborhood size) from $\mathcal{O}(kN^2 \log N)$ to $\mathcal{O}(knN \log N)$.
  - MDS from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 N)$.
- If x is a landmark point, then the embedding given by LMDS is consistent with the original MDS embedding.
- If the distance matrix $D_{n,N}$ can be represented exactly by a Euclidean configuration in $\mathbb{R}^l$, and if the landmarks are chosen so that their affine span in that the configuration is l-dimensional (i.e. general position), then LMDS will recover the configuration exactly, up to rotation and translation.

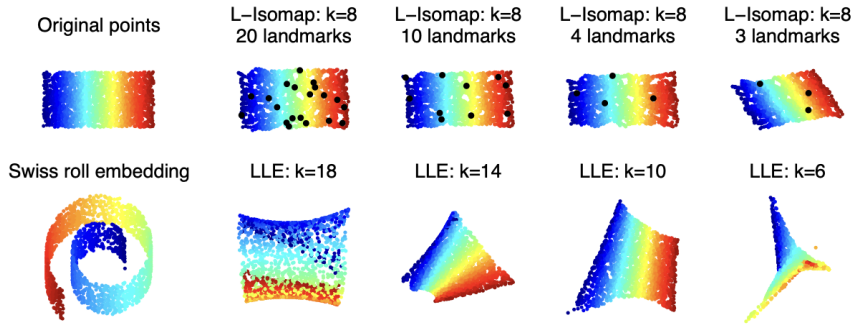

Figure 2: L-Isomap is stable over a wide range of values for the sparseness parameter (the number of landmarks). Results from LLE are shown for comparision[14].

3 `dimRed` and `coRanking` - Unifying Dimensionality Reduction in R [2]

3.1 Abstract

Original: "Dimensionality reduction" (DR) is a widely used approach to find low dimensional and interpretable representations of data that are natively embedded in high-dimensional spaces. DR can be realized by a plethora of methods with different properties, objectives, and, hence, (dis)advantages. The resulting low-dimensional data embeddings are often difficult to compare with objective criteria. Here, we introduce the `dimRed` and `coRanking` packages for the R language. These open source software packages enable users to easily access multiple classical and advanced DR methods using a common interface. The packages also provide quality indicators for the embeddings and easy visualization of high dimensional data. The `coRanking` package provides the functionality for assessing DR methods in the co-ranking matrix framework. In tandem, these packages allow for uncovering complex structures high dimensional data. Currently 15 DR methods are available in the package, some of which were not previously available to R users. Here, we outline the `dimRed` and `coRanking` packages and make the implemented methods understandable to the interested reader.

Apple Intelligence summary: Dimensionality reduction (DR) methods create low dimensional data representations, but comparison is challenging. The `dimRed` (figure 3) and `coRanking` R packages are introduced to address this.

3.2 Key Points

- The difficulty in applying DR is that each DR method is designed to maintain certain aspects of the original data and therefore may be appropriate for one task and inappropriate for another. Most methods also have parameters to tune and follow different assumptions.
- **Software packages for other languages**:
  - Python: `scikit-learn`, which contains a module for DR.
  - Julia: `ManifoldLearning.jl` for nonlinear and `MultivariateStats.jl` for linear DR.
  - Matlab: several toolboxes.
  - C++: `Shogun` toolbox, which offers bindings for many high level languages (including R).
- At the time (2018), no comprehensive package for R.
- None of the former provides means to consistently compare the quality of different methods.
- MDS can be seen as kPCA with kernel $x^T y$, since a distance matrix can be transformed to a matrix of inner products.
- `dimRed` wraps `cmdscale`.

- In contrast to a supervised problem, there is no natural way to directly measure the quality of any output or to compare two methods. Every method optimizes a different error function.
- **Quality criteria implemented in** `coRanking`:
  - **Co-ranking matrix based measures**: the co-ranking matrix Q is the 2d-histogram of the distance ranks. q_{ij} is an integer which counts how many points of distance rank j became rank i. In a perfect DR, this matrix will only have non-zero entries in the diagonal. In R, the co-ranking matrix can be calculated using the the `coRanking::coranking` function. The `dimRed` package contains the functions `Q_local, Q_global, Q_NX, LCMC`, and `R_NX` to calculate the above quality measures in addition to `AUC_lnK_R_NX`. If R_{NX} is high for low values of K, then local neighborhoods are maintained well; if R_{NX} is high for large values of K, then global gradients are maintained well (see fig. 4).
  - Cophenetic correlation.
  - **Reconstruction error**: the fairest one when the method provides an inverse mapping. RMSE $= \sqrt{\frac{1}{n}\sum_{i=1}^{n} d\left(x'_i, x_i\right)^2}$, with $x'_i = f^{-1}(y_i) = f^{-1}(f(x_i))$.
- **Test datasets**: Common ones being the 3d S-curve and the Swiss roll. Real world examples usually have more dimensions and often are much noisier and we cannot be sure if we can observe all the relevant variables. Can be retrieved with `dimRed::loadDataSet`
- **Main functions:** `embed, quality, plot, plot_R_NX`.

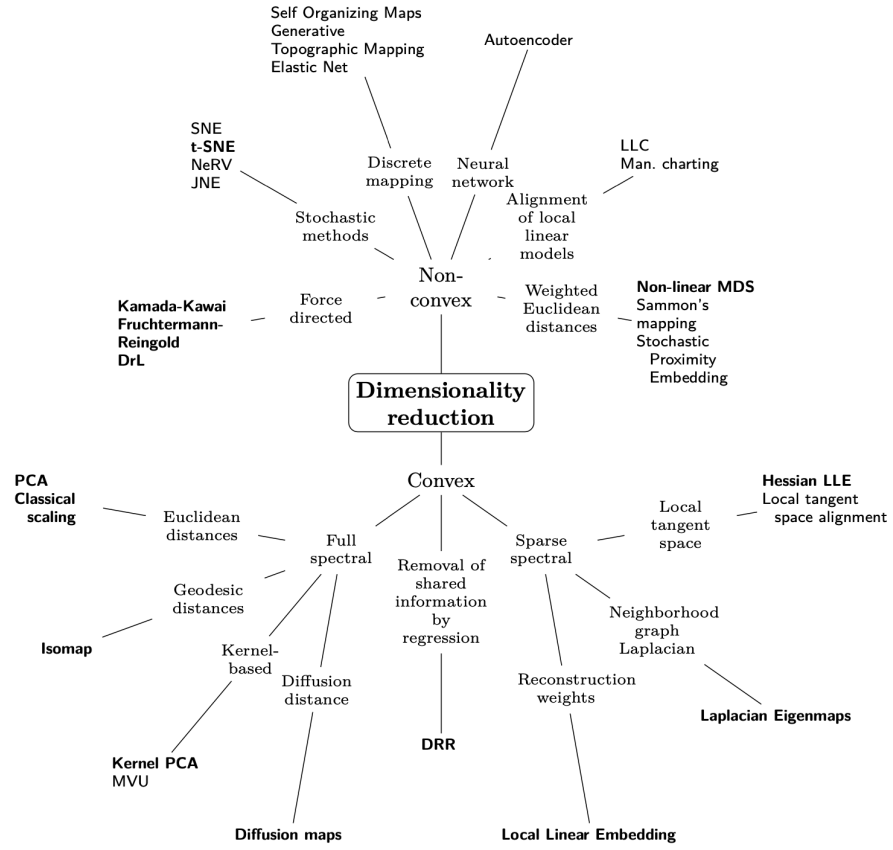

Figure 3: DR methods implemented in `dimRed` [2].

4 Sparse multidimensional scaling using landmark points [1]

4.1 Abstract

Original: In this paper, we discuss a computationally efficient approximation to the classical multidimensional scaling (MDS) algorithm, called Landmark MDS (LMDS), for use when the number of data points is very large. The first step of the algorithm is to run classical MDS to embed a chosen subset of the data, referred to as the 'landmark points', in a low-dimensional space. Each remaining data point can be located within this space given knowledge of its distances to the landmark points. We give an elementary and explicit theoretical analysis of this procedure, and demonstrate with examples that LMDS is effective in practical use.

Apple Intelligence summary: Landmark MDS approximates classical multidimensional scaling for large datasets. It embeds a subset of data points, called "landmark points", in a low-dimensional space, then locates remaining points based on their distances to these landmarks.

4.2 Key Points

- **LMDS**: Landmark MDS.

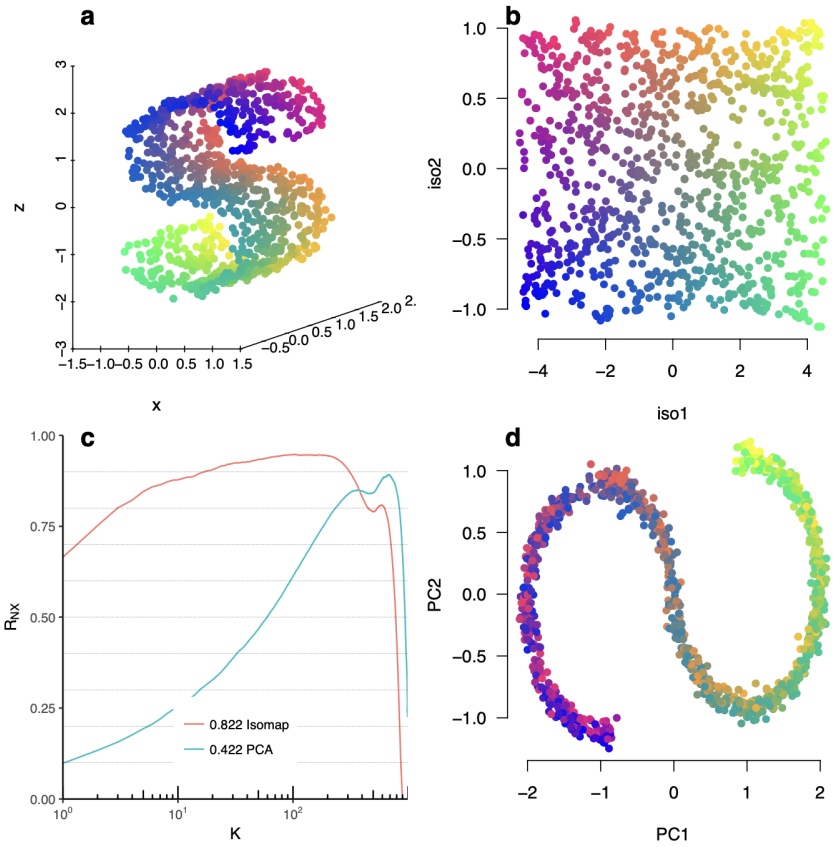

Figure 4: R_{NX} measures the quality of the embedding. [2].

- **Method**:
  1. Select n landmark points from N data points.
  2. Apply MDS to the $n \times n$ distance matrix to obrain L.
  3. Embed remaining points via distance-based triangulation.
- **Complexity**: Classical MDS: $\mathcal{O}(N^2)$ storage, $\mathcal{O}(N^3)$ time. LMDS: $\mathcal{O}(nN)$ storage, lower time complexity.
- It has links with Isomap (L-Isomap), the Nyström method (which finds approximate solutions to a positive semi-definite symmetric eigenvalue problem using just a few of the columns of the matrix) and FastMap.

5 Comparative study for dimensionality reduction techniques for big data [13]

5.1 Abstract

Original: Nowadays, big data represents the solution for different type of users especially enterprises due to its huge amount of information augmented in real time. All these generated data could be described in one of big data characteristics named variety. One of the most challenging issues for big data variety is high dimensionality because, it prevents the analysis process, demands heavy computations and adds noise to our data. The solution for this challenging issue is dimensionality reduction which minimize the dimensions and keeps only the essential information that give accurate results during analysis and decrease the computations complexity.

This paper aims to summarize and compare some of the recent methods that are used to solve the problem of high dimensionality in big data, Thereby, facilitating the process of research in this field.

Apple Intelligence summary: High dimensionality in big data poses challenges, requiring dimensionality reduction to minimize dimensions and retain essential information. This paper compares recent methods for solving this problem, aiding research in the field.

5.2 Key Points

- **Techniques Compared:**
  - Classical/Modified PCA (modified to handle memory limits via row-scanning and MapReduce).
  - Simplicial Nonnegative Matrix Tri-Factorization (SNMTF), an improved variant of NMF.
  - Stacked Autoencoders, which yield lower reconstruction error than PCA.
  - Linguistic Hedges Neuro-Fuzzy Classifiers with Feature Selection (LHNFCSF), a combination of neural networks and fuzzy inference systems (neuro-fuzzy) based on linguistic hedges with feature selection method.
  - Deep Belief Networks (DBNs), which consist of multiple hidden layers where each layer is an RBM (Restricted Boltzmann Machine), which is also a class of neural networks. Each RBM is connected with two layers, a hidden layer and a visible layer, and so on.

6 Out-of-Core Dimensionality Reduction for Large Data via Out-of-Sample Extensions [12]

6.1 Abstract

Original: Dimensionality reduction (DR) is a well-established approach for the visualization of high-dimensional data sets. While DR methods are often applied to typical DR benchmark data sets in the literature, they might suffer from high runtime complexity and memory requirements, making them unsuitable for large data visualization especially in environments outside of high-performance computing. To perform DR on large data sets, we propose the use of out-of-sample extensions. Such extensions allow inserting new data into existing projections, which we leverage to iteratively project data into a reference projection that consists only of a small manageable subset. This process makes it possible to perform DR out-of-core on large data, which would otherwise not be possible due to memory and runtime limitations. For metric multidimensional scaling (MDS), we contribute an implementation with out-of-sample projection capability since typical software libraries do not support it.

We provide an evaluation of the projection quality of five common DR algorithms (MDS, PCA, t-SNE, UMAP, and autoencoders) using quality metrics from the literature and analyze the trade-off between the size of the reference set and projection quality. The run-time behavior of the algorithms is also quantified with respect to reference set size, out-of-sample batch size, and dimensionality of the data sets. Furthermore, we compare the out-of-sample approach to other recently introduced DR methods, such as PaCMAP and TriMAP, which claim to handle larger data sets than traditional approaches. To showcase the usefulness of DR on this large scale, we contribute a use case where we analyze ensembles of streamlines amounting to one billion projected instances.

Apple Intelligence summary: The study compares out-of-sample DR methods with PaCMAP and TriMAP on large datasets. A use case demonstrates the usefulness of DR on a dataset of one billion projected streamline instances.

6.2 Key Points

- A framework for OOS extensions of DR methods similar to Interpolation MDS.
- Applied to PCA, MDS, t-SNE, UMAP and autoencoders.
- Compared to TriMap and PaCMAC, which are efficient DR methods suitable for big data.

Algorithm 1 Projection with random subset reference

var n_{ref} ▷ reference size
var n_{batch} ▷ batch size
procedure PROJECT(X, Φ) ▷ X: data set, Φ: DR method
    $X_a \leftarrow n_{\text{ref}}$ random points of X
    Y_a, $\beta \leftarrow \Phi(X_a)$ ▷ β: learned parameters
    $X_r \leftarrow X \setminus X_a$ ▷ X_r: remaining data
    for $i \in \{1 \ldots \lceil \text{len}(X_r)/n_{\text{batch}} \rceil\}$ **do** ▷ project batches
        $X_{b(i)} \leftarrow i^{\text{th}}$ subset of X_r
        $Y_{b(i)} \leftarrow \Phi_\beta(X_{b(i)})$ ▷ parameters β stay fixed
    end for
    return $Y_a \cup Y_{b(1)} \cup \ldots \cup Y_{b(n_{\text{batch}})}$
end procedure

Figure 5: OOS algorithm [12].

Table 1: DR methods used in the evaluation of the OOS framework.

Method	Optimizes for	Linear	β
PCA	reconstruction err.	yes	eigenvectors
MDS	global distances	no	$X_a \to Y_a$
t-SNE	local neighborhood	no	$X_a \to Y_a$, kNN
UMAP	local nb., global dst.	no	$X_a \to Y_a$, kNN
Autoencoder	reconstruction err.	no	weights, biases

Figure 6: OOS methods [12].

7 DR Python packages

7.1 Key Points

- To our knowledge, there is no comprehensive Python package that implements many DR techniques akin the R library `Rdimtools` (see section 1).
- To show the consequences of this in an example, classical MDS is only implemented in `pyseer` (through `pyseer.cmdscale`), a package for microbial pangenome-wide association studies [3].
- The package that contains the most DR methods is `scikit-learn`, specifically in its `manifold` module [9]. It implements: Isomap, LLE, Laplacian Eigenmaps, t-SNE and non-classical MDS (both `sklearn.manifold.MDS` and `sklearn.manifold.smacof` can perform metric and non-metric MDS). Moreover, `sklearn.manifold.trustworthiness` measures to what extent the local structure is retained when dimensionality is reduced. Hence, the total amount of methods present in `sklearn.manifold` cannot be compared to the 143 of `Rdimtools`.

- Another remarkable, although differently oriented, library is `direpack`, which implements a set of modern statistical dimension reduction techniques including projection pursuit, sufficient dimension reduction, and robust M estimators. It also includes regularized regression estimators, pre-processing utilities, plotting functionality, and cross-validation utilities, all consistent with the scikit-learn API [7]. Nonetheless, these methods are outside the scope of our thesis.

- Other, more specific, libraries that implement DR methods in Python are:

  - `umap-learn`: the library that introduced UMAP [6].
  - `MulticoreTSNE`: a multicore modification of Barnes-Hut t-SNE with Python CFFI-based wrappers [15]. Barnes-Hut is a tree-based algorithm that can be used to accelerate t-SNE up to $\mathcal{O}(N \log N)$ [5].
  - `fitsne`: Fast Fourier Transform-accelerated Interpolation-based t-SNE [4].
  - `OpenTSNE`: incorporates the latest improvements to the t-SNE algorithm, including the ability to add new data points to existing embeddings, massive speed improvements, enabling t-SNE to scale to millions of data points and various tricks to improve global alignment of the resulting visualizations [10]. The team behind `OpenTSNE` benchmarked the here described t-SNE Python implementations and showed that the fastest one in general is `fitsne`, although `OpenTSNE` is equally efficient on multicore systems (see fig. 7) [10].

- Moreover, other libraries focused on certain fields like genomics (`phate`) or NLP (`gensim`) implement DR methods for specific tasks and scenarios in their topics [8, 11].

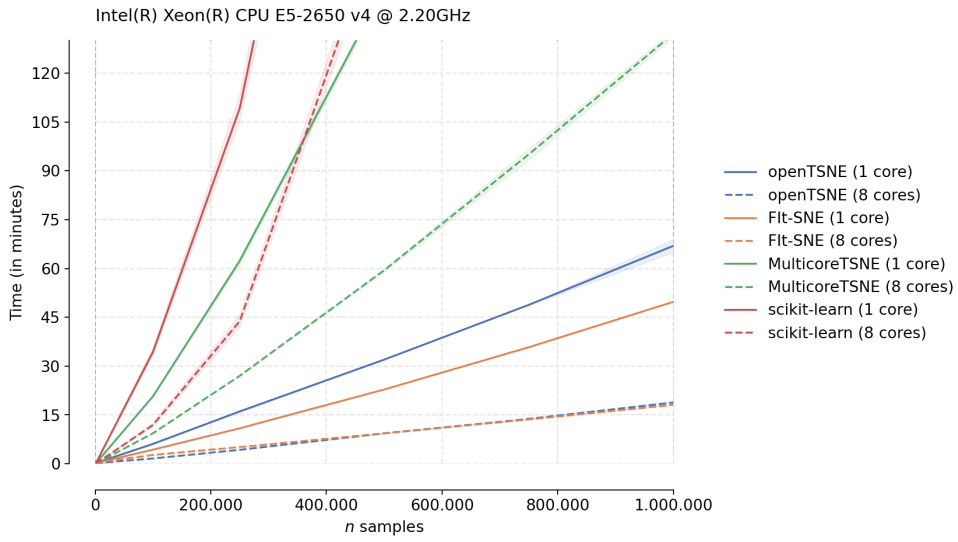

Figure 7: Benchmark of t-SNE Python implementations [10].

8 Title TEXstring

8.1 Abstract

Original:

Apple Intelligence summary:

8.2 Key Points

-

References

[1] Vin De Silva and Joshua B Tenenbaum. *Sparse multidimensional scaling using landmark points*. Tech. rep. Technical Report, Stanford University, 2004.

[2] Guido Kraemer, Markus Reichstein, and Miguel D. Mahecha. "dimRed and coRanking - Unifying Dimensionality Reduction in R". In: *R J.* 10 (2018), p. 342. URL: https://api.semanticscholar.org/CorpusID:62831555.

[3] John A Lees et al. "pyseer: a comprehensive tool for microbial pangenome-wide association studies". In: *Bioinformatics* 34.24 (July 2018), pp. 4310–4312. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty539. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/24/4310/48919461/bioinformatics_34_24_4310.pdf. URL: https://doi.org/10.1093/bioinformatics/bty539.

[4] George C. Linderman et al. "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data". In: *Nature Methods* 16.3 (Mar. 1, 2019), pp. 243–245. DOI: 10.1038/s41592-018-0308-4. URL: https://doi.org/10.1038/s41592-018-0308-4.

[5] Laurens van der Maaten. "Accelerating t-SNE using Tree-Based Algorithms". In: *Journal of Machine Learning Research* 15.93 (2014), pp. 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

[6] L. McInnes, J. Healy, and J. Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.03426 [stat.ML].

[7] Emmanuel Jordy Menvouta, Sven Serneels, and Tim Verdonck. "direpack: A Python 3 package for state-of-the-art statistical dimensionality reduction methods". In: *SoftwareX* 21 (2023), p. 101282. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2022.101282. URL: https://www.sciencedirect.com/science/article/pii/S235271102200200X.

[8] Kevin R. Moon et al. "Visualizing structure and transitions in high-dimensional biological data". In: *Nature Biotechnology* 37.12 (Dec. 1, 2019), pp. 1482–1492. DOI: 10.1038/s41587-019-0336-3. URL: https://doi.org/10.1038/s41587-019-0336-3.

[9] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[10] Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. "openTSNE: A Modular Python Library for t-SNE Dimensionality Reduction and Embedding". In: *Journal of Statistical Software* 109.3 (2024), pp. 1–30. DOI: 10.18637/jss.v109.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v109i03.

[11] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[12] Luca Reichmann, David Hägele, and Daniel Weiskopf. *Out-of-Core Dimensionality Reduction for Large Data via Out-of-Sample Extensions*. 2024. arXiv: 2408.04129 [cs.LG]. URL: https://arxiv.org/abs/2408.04129.

[13] Henouda Salah Eddine et al. "Comparative study for dimensionality reduction techniques for big data". In: Nov. 2020.

[14] Vin Silva and Joshua Tenenbaum. "Global Versus Local Methods in Nonlinear Dimensionality Reduction". In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press, 2002. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/5d6646aad9bcc0be55b2c82f69750387-Paper.pdf.

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

[16] Kisung You and Dennis Shung. "Rdimtools: An R Package for Dimension Reduction and Intrinsic Dimension Estimation". In: *Software Impacts* 14 (2022), p. 100414. ISSN: 26659638. DOI: 10.1016/j.simpa.2022.100414.