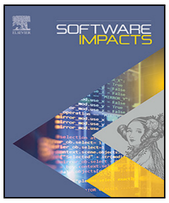
Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts



Original software publication

# Generative locally linear embedding: A module for manifold unfolding and visualization

Benyamin Ghojogh [a,*], Ali Ghodsi [b], Fakhri Karray [c], Mark Crowley [a]

[a] Machine Learning Laboratory, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada
[b] Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON, Canada
[c] Centre for Pattern Analysis and Machine Intelligence, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

ARTICLE INFO

*Keywords:*
Machine learning
Manifold learning
Dimensionality reduction
Unfolding and visualization

ABSTRACT

Data often have nonlinear patterns in machine learning. One can unfold the nonlinear manifold of a dataset for low-dimensional visualization and feature extraction. Locally Linear Embedding (LLE) is a nonlinear spectral method for dimensionality reduction and manifold unfolding. It embeds data using the same linear reconstruction weights as in the input space. In this paper, we propose an open source module which not only implements LLE, but also includes implementations of two generative LLE algorithms whose linear reconstruction phases are stochastic. Using this module, one can generate as many manifold unfoldings as desired for data visualization or feature extraction.

## Code metadata

| | |
|---|---|
| Current code version | v1 |
| Permanent link to code/repository used for this code version | https://github.com/SoftwareImpacts/SIMPAC-2021-63 |
| Permanent link to Reproducible Capsule | https://codeocean.com/capsule/0270963/tree/v1 |
| Legal Code License | MIT |
| Code versioning system used | git |
| Software code languages, tools, and services used | Python3 |
| Compilation requirements, operating environments & dependencies | Python3, numpy, matplotlib, scikit-learn |
| If available Link to developer documentation/manual | https://github.com/bghojogh/Generative-LLE/blob/main/README.md |
| Support email for questions | bghojogh@uwaterloo.ca |

## 1. Introduction

Dimensionality reduction can be used for manifold unfolding, data visualization, and feature extraction for better classification, prediction, or clustering of data. According to [1], it can be categorized into spectral [2], probabilistic [3], and deep methods [4]. Locally Linear Embedding (LLE), proposed in [5,6], is a nonlinear unsupervised spectral method for dimensionality reduction and manifold unfolding. It consists of three steps which are (1) constructing the $k$-nearest neighbors graph, (2) deterministic linear reconstruction of points by neighbors in the input space, and (3) deterministic linear embedding using the obtained weights [7]. LLE is a very important nonlinear manifold learning because it fits data locally in the embedding space to preserve the global structure of data [6]. This idea was later developed to the field of unsupervised manifold learning where the local distances or similarities of points are considered for embedding. There are two existing Generative LLE (GLLE) methods described in [8] whose linear reconstruction steps are stochastic. One GLLE uses Expectation Maximization (EM) and the other is based on direct sampling [8]. GLLE models are partly inspired by factor analysis [9] and probabilistic principal component analysis [10] where points are conditioned on a latent variable and noise (see [11] for details). Hence, they are combinations of spectral and probabilistic methods.

The GLLE algorithms can be considered as generative models [12], although they are not deep or autoencoder-based methods [13,14]. Deep generative models have been widely used for various applications

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.

*E-mail addresses:* bghojogh@uwaterloo.ca (B. Ghojogh), ali.ghodsi@uwaterloo.ca (A. Ghodsi), karray@uwaterloo.ca (F. Karray), mcrowley@uwaterloo.ca (M. Crowley).

https://doi.org/10.1016/j.simpa.2021.100105
Received 7 June 2021; Received in revised form 29 June 2021; Accepted 29 June 2021

2665-9638/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

**Table 1**
Settings in the json file of module.

| Setting | Choices | | |
|---|---|---|---|
| `method` | `LLE` | `GLLE` | `GLLE_DirectSampling` |
| | deterministic LLE | GLLE with EM | GLLE with direct sampling |
| `dataset` | `Swiss_roll`, `Swiss_roll_hole`, `S_curve`, `Sphere`, `Sphere_small` | | `User_data` |
| | ready datasets | | user's dataset |
| `make_dataset_again` | `True` | | `False` |
| | generate the ready datasets | | load data |
| `embed_again` | `True` | | `False` |
| | train embedding again | | useful for generations after training (not train again) |
| `generate_embedding_again` | `True` | | `False` |
| | generate [multiple] unfoldings | | no generations |
| `analyze_covariance_scales` | `True` | | `False` |
| | generate unfoldings for various scales of covariance | | no analysis of covariance scales |
| `n_generation_of_embedding` | A positive integer | | |
| | the number of unfoldings (embeddings) to generate | | |
| `max_iterations` | A positive integer | | |
| | maximum number of iterations for EM algorithm | | |
| `n_components` | A positive integer (between 1 and dimensionality of data) | | |
| | the dimensionality of unfolding (embedding) | | |
| `verbosity` | 0 | 1 | 2 |
| | not printing logging | print logging level 1 | print logging level 2 |

such as image synthesis and denoising [15]. However, GLLE methods are used for generating manifold unfoldings and data visualization in low-dimensional subspaces. In this paper, we propose a new open-source software module, in Python language [16], for LLE and GLLE algorithms. This module can be useful for nonlinear manifold unfolding and visualization.

## 2. Description and usage

This module includes a `main.py` file which instantiates the classes of algorithms, loads data, and feeds the datasets into the algorithms. The three classes of module implement LLE, GLLE with EM algorithm, and GLLE with direct sampling. There also exist some additional utility functions for saving, loading, and plotting results. As also stated in the read-me file of GitHub page for this module, there are several easy steps required for running the module. In the following, we explain these steps in detail.

**Preparation:** there are some essential dependencies which need to be installed before running the module. These requirements are *Python3, numpy, matplotlib, scikit-learn* and they can be installed by running the command "`pip install -r requirements.txt`".

**Settings:** The user should specify their desired settings in the provided user-friendly file named `settings.json`. These settings, which specify the algorithm, the number of iterations, the dataset, and similar choices, are summarized in Table 1 with their possible values.

**Dataset Loading:** User can choose among several readily provided datasets such as Swiss roll, Swiss roll with hole, S curve, severed sphere, and small severed sphere. For that, they should set `make_dataset_again=True` and `dataset` to one of the ready datasets. This will generate a random dataset in the path `./datasets/dataset_name/`.

The user is also free to provide the module with their own dataset. For that, they should set `make_dataset_again=False` and `dataset=User_data` and put their dataset in the path `./datasets/User_data/`. The format of data should be csv where `data.csv` should include data instances stacked row-wise where columns are features. The files `labels.csv` abd `color.csv` are optional where user can determine the class labels and colors of points for their relative distances on manifold, respectively.

**Module Running:** After setting the appropriate settings, the module can be run with the command "`python main.py`" in the root folder. According to the selected method in the settings file, the module unfolds manifold using LLE, GLLE with EM, or GLLE with direct sampling. If using LLE, only one embedding is output; however, by using GLLE, the module will learn unfolding once and then it can generate as many unfoldings as desired.

## 3. Impact overview

The proposed module can be used for generating multiple nonlinear manifold unfoldings for data visualization and feature extraction. Many of the dimensionality reduction methods, including LLE [5,6], generate merely one unfolding for the underlying manifold of data. That is while the GLLE algorithms in the module can generate as many manifold unfoldings as wanted where the generated visualizations all relate to the original LLE embedding because of their local fitting [6]. In addition to the benefit of having multiple embeddings for visulization and feature extraction, these unfoldings can bring more insights into the LLE embeddings by inspecting them. Researchers can benefit from this module by extracting meaningful and insightful embeddings and manifold unfoldings. This module can be used in various applications. It can be used for extracting features for better discrimination of classes or clusters. For instance, in medical image analysis, the tumorous images can be better separated from the normal images in the low-dimensional embedding space. Moreover, visualizing high dimensional data in low-dimensional figures can be used in statistics and genome analysis.

## 4. Conclusion and future work

We proposed a module for manifold unfolding and visualization. This module includes implementation of LLE and GLLE algorithms with EM and direct sampling. We introduced the module and explained how to run and use this module. The use cases and impact of this module were also discussed. For improvement of the module, we are working on better efficiency of algorithms when dealing with very large datasets. Out-of-sample extension of GLLE algorithms are also possible as future work for this module.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work has been partially supported by a Discovery Grant by the Natural Sciences and Engineering Research Council of Canada (RGPIN-2018-04381).

## References

[1] B. Ghojogh, Data Reduction Algorithms in Machine Learning and Data Science (Ph.D. thesis), University of Waterloo, 2021.
[2] L.K. Saul, K.Q. Weinberger, F. Sha, J. Ham, D.D. Lee, Spectral methods for dimensionality reduction, Semi-Supervised Learn. 3 (2006).
[3] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[4] Y. Wang, H. Yao, S. Zhao, Auto-encoder based dimensionality reduction, Neurocomputing 184 (2016) 232–242.
[5] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[6] L.K. Saul, S.T. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (Jun) (2003) 119–155.
[7] B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley, Locally linear embedding and its variants: Tutorial and survey, 2020, ArXiv Preprint arXiv:2011.10925.
[8] B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley, Generative locally linear embedding, 2021, ArXiv Preprint arXiv:2104.01525.
[9] B. Fruchter, Introduction To Factor Analysis, Van Nostrand, 1954.
[10] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, J. R. Stat. Soc. Ser. B Stat. Methodol. 61 (3) (1999) 611–622.
[11] B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley, Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey, 2021, ArXiv Preprint arXiv:2101.00734.
[12] G. Harshvardhan, M.K. Gourisaria, M. Pandey, S.S. Rautaray, A comprehensive survey and analysis of generative models in machine learning, Comp. Sci. Rev. 38 (2020) 100285.
[13] A. Oussidi, A. Elhassouny, Deep generative models: Survey, in: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), IEEE, 2018, pp. 1–8.
[14] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, Y. Zheng, Recent progress on generative adversarial networks (GANs): A survey, IEEE Access 7 (2019) 36322–36333.
[15] E. Luhman, T. Luhman, Denoising synthesis: A module for fast image synthesis using denoising-based models, Softw. Impacts (2021) 100076.
[16] M.F. Sanner, Python: a programming language for software integration and development, J. Mol. Graph. Model. 17 (1) (1999) 57–61.