



# Benchmarking Manifold Learning Methods on a Large Collection of Datasets

Patryk Orzechowski<sup>1,2(✉)</sup>, Franciszek Magiera<sup>2</sup>, and Jason H. Moore<sup>1</sup>

<sup>1</sup> Institute for Biomedical Informatics, University of Pennsylvania,  
Philadelphia, PA 19104, USA  
[patryk.orzechowski@gmail.com](mailto:patryk.orzechowski@gmail.com)

<sup>2</sup> Department of Automatics, AGH University of Science and Technology,  
al. Mickiewicza 30, 30-059 Krakow, Poland

**Abstract.** Manifold learning, a non-linear approach of dimensionality reduction, assumes that the dimensionality of multiple datasets is artificially high and a reduced number of dimensions is sufficient to maintain the information about the data. In this paper, a large scale comparison of manifold learning techniques is performed for the task of classification. We show the current standing of genetic programming (GP) for the task of classification by comparing the classification results of two GP-based manifold learning methods: GP-Mal and ManiGP - an experimental manifold learning technique proposed in this paper. We show that GP-based methods can more effectively learn a manifold across a set of 155 different problems and deliver more separable embeddings than many established methods.

**Keywords:** Manifold learning · Genetic programming · Machine learning · Dimensionality reduction · Benchmarking

## 1 Introduction

Dimensionality reduction has been a very important area of research over the past few years because of its pivotal role in machine learning (ML) and related fields. Feature extraction, which is determining the most informative and non-redundant features derived from the original features, reduces the feature space and allows multiple ML methods to be applied to increasingly large datasets. It also allows to create human interpretable visualization of the data in two or three dimensional space and to better understand underlying associations between the features. An extensive review of different dimensionality reduction approaches can be found in the literature [8, 9, 14, 16, 18, 22, 36, 48].

Over the years, multiple methods of dimensionality reduction have been developed. The most popular linear dimensionality reduction methods include principal component analysis (PCA) [30, 39], linear discriminant analysis (LDA) [12, 33], canonical correlations analysis [17] and factor analysis (FA) [37].

Manifold learning is one of the approaches for non-linear dimensionality reduction. The most popular approaches include multidimensional scaling (MDS) [4, 40], locally linear embeddings (LLE) [35], Laplacian eigenmaps [2], isomaps [38], local tangent space alignment (LTSA) [47], maximum variance unfolding [44], diffusion maps [7], and t-distributed stochastic neighbor embedding (t-SNE) [25]. Apart from visualization purposes, manifold learning has been also used as a preprocessing step before classification [41, 42].

Among multiple dimensionality reduction techniques, a couple of notable methods of feature extraction exist that use genetic algorithms (GA) [34, 45] or genetic programming (GP) [6, 15]. Over the recent years multiple attempts were also taken to use genetic programming for classification [3, 11, 20]. The main advantage of GP-based approaches is delivering a fully interpretable model that could be used for describing the data structure. In a recent paper Lensen et al. proposed a manifold learning method based on GP called GP-MaL. The method was shown to outperform other popular manifold learning techniques in terms of accuracy in at least half of 10 datasets considered [23]. This approach inspired us to design a study on separability of data using manifold learning techniques. We designed a GP-based method focusing in optimizing the same goal used in benchmarking the methods, which is providing observable separation of classes in the embedded space.

The major contribution of this paper is performing a large-scale comparison of different manifold learning methods with their sets of parameters on the large collection of 155 datasets from Penn Machine Learning Benchmark (PMLB) [27]. To our knowledge, this is the largest and the most comprehensive comparison of manifold learning techniques on the collection of real world problems. As manifold learning methods are usually applied to create convincing visualizations, we check if manifold learning methods can deliver easily separable embeddings in two dimensional space. To measure the performance of the methods, we used unsupervised clustering algorithm and verified the performance of the methods for the task of classification, for which the ground truth is already known. As some of the considered datasets in PMLB are multi-class problems with different numbers of instances per class, a balanced accuracy score [5] is used in order to account for the class imbalance. Type of the data (e.g. categorical or ordinal) was not taken into account.

The second contribution of the paper is providing a convenient open-source framework for testing new manifold learning methods. All source code for our analysis is available at <https://github.com/athril/manigp>. An important input here is providing a scikit-learn friendly wrapper for GP-MaL, a GP-based manifold learning method proposed at EuroGP track of EvoStar 2019, which investigated application of genetic programming to manifold learning.

Thirdly, we propose a novel manifold learning technique based on genetic programming called ManiGP which uses a multi-tree representation, a popular k-means clustering [24] and balanced accuracy to verify their integrity. The proposed method could be considered a thought experiment that would answer a question if a method intentionally designed to exploit the benchmark has unfair

advantage over its competitors. We show over a large collection of problems that, albeit its extensive running time, a method based on genetic programming can learn a manifold significantly better than multiple other techniques. Therefore, we touch on a broader subject of the fairness of comparisons of the methods which feature GP.

## 2 Methods

In our study we covered eight well established manifold learning methods and two methods that use GP: GP-MaL, a winner of a best paper award at EvoStar 2019, and ManiGP, which is introduced in this paper. All the methods were benchmarked against the collection of 155 datasets from Penn Machine Learning Benchmark (PMLB). The input data was standardized using RobustScaler and split into training and test set with stratification maintained. Manifold learning method were launched with different combinations of input parameters (apart from GP-MaL, which was launched with default parameters). The embeddings provided by the methods were later clustered using K-Means, a popular clustering algorithm. Hungarian (Kuhn-Munkres) algorithm was used for obtaining optimal assignment of clusters to the actual classes [26]. The distance to the nearest centroid served as a basis for predicting a class label for unseen data points. Balanced accuracy was used as a metric for measuring performance (i.e separability) of the embeddings. As the major application of manifold learning is visualization, we have focused on distinction between the classes in two dimensional space. The detailed information on the design of the experiment is provided in the methodology section.

### 2.1 Manifold Learning Methods

We have included the methods that are part of scikit-learn [31], a popular machine learning library. We have also adapted the source code GP-MaL and created a Python wrapper.

*Isomap* is a manifold learning method partly based on MDS [38]. Isomap improves on standard MDS by aiming to preserve geodesic distances between points instead of straight-line distances. This change leads to the creation of more adequate embeddings compared to MDS for manifolds with characteristics akin to the Swiss Roll for which the small euclidean distance between two points does not imply that they are similar. Isomap is computationally efficient and scales well to high-dimensional datasets.

*Locally Linear Embedding (LLE)* is based on a premise that finding linear transformations preserving local structures and applying them to overlaying neighborhoods can retain a non-linear global structure of a dataset [35]. Local structures are maintained by expressing each data instance as a linear combination of its neighbors using the same weights in high and low-dimensional space. LLE can perform poorly when data is separated and has tendencies to collapse the low-dimensional mapping to a single point.

*Hessian Locally Linear Embedding (HLLE)* is a variant of LLE that uses quadratic form based on Hessian matrix to preserve the local structure of the data [10]. HLLE was shown to perform better on non-convex manifolds than standard LLE, but it suffers from high computational complexity and does not scale well to large datasets, partly due to the necessity of estimating second order derivatives.

*Modified Locally Linear Embedding (MLLE)* is a variant of LLE that looks for more than one linear combinations of every point's neighbors to embed the data in a low-dimensional space while maintaining the local structure of a manifold [46]. This change results in a more robust embedding compared to standard LLE.

*Local Tangent Space Alignment (LTSA)* is a modification of LLE that represents local structures using tangent spaces and aligns them in a global structure to derive a coordinate system to describe manifold [47]. Similarly to LLE, the method is susceptible to the noise.

*Multidimensional Scaling (MDS)* is one of the oldest methods which can be applied to non-linear dimensionality reduction [4]. It focuses on maintaining the euclidean distances between the points in the low-dimensional embedding. This approach works well if distance is a good measure of similarity between points, because then related points are grouped together.

*Laplacian Eigenmaps - Spectral embedding (SE)* is another manifold learning method which focuses on preserving the local structures of the data [2]. This is achieved by representing the dataset as a graph and using the eigenvectors of the Laplacian matrix of that graph for dimensionality reduction. Laplacian Eigenmaps have also been used for data clustering.

*t-Distributed Stochastic Neighbor Embedding (t-SNE)* is a popular method for visualizing high-dimensional datasets [25]. Similarity of points in the dataset is evaluated using conditional probabilities which measure how likely a certain point would be to choose another point as its neighbor. The method looks for a low-dimensional embedding of the dataset that minimizes the difference between the distributions of high and low-dimensional sets which is expressed by Kullback-Leibler divergence [19]. t-SNE was shown to create high quality visualizations of high-dimensional datasets [25]. In addition, it is a powerful tool for data exploration and can be used after feature extraction to assess what characteristics of the data the extraction really captured. Nonetheless, t-SNE has some disadvantages. Firstly, it is not easily interpretable [43]. Secondly, new points cannot be embedded in low-dimensional space<sup>1</sup>.

---

<sup>1</sup> <https://lvdmaaten.github.io/tsne/>.

*GP-MaL* is a recently proposed manifold learning method based on genetic programming [23]. The method uses interpretable trees to evolve mappings from high to low dimensional space. The mappings are measured on how well they maintain dataset’s structure with a similarity metric based on neighbor ordering. GP-MaL was shown to be a competitive dimensionality reduction technique for the tasks of classification, data visualization and establishing feature importance. The mappings produced by the method while interpretable, suffered from excessive complexity. GP-MaL demonstrated the potential of GP in dimensionality reduction and manifold learning applications.

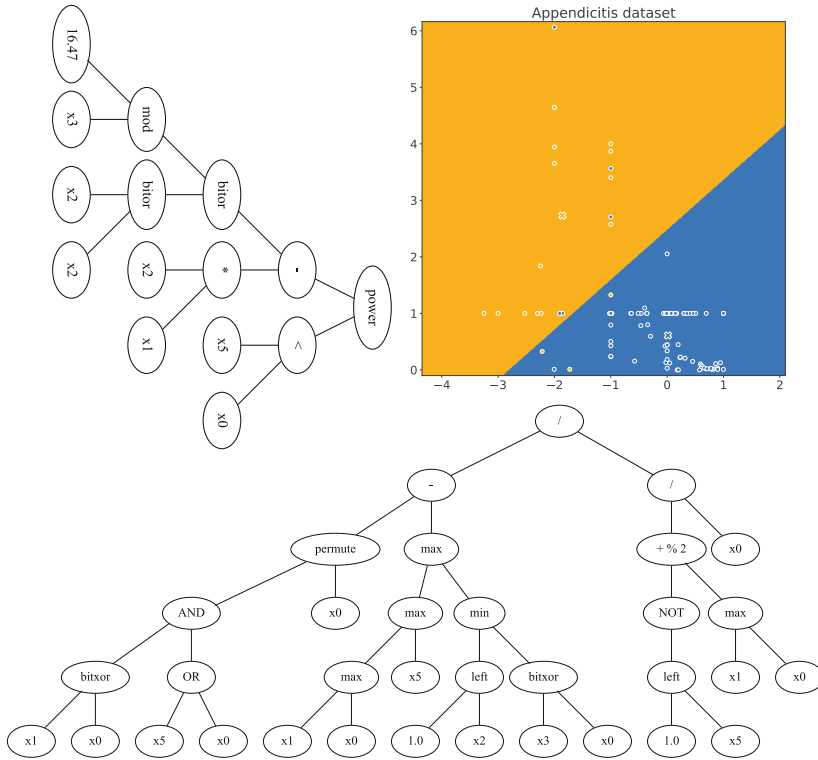
## 2.2 ManiGP - A New Manifold Learning Method Based on Genetic Programming

ManiGP is a manifold learning method which, similar to GP-MaL, uses a multi-tree representation of an individual and focuses on delivering highly separable mappings of distinct classes.

*Motivation.* The intuition behind the method is as follows. As manifold learning techniques are primarily used for creating convincing visualizations, our aim was to provide a highly interpretable method, which would clearly distinguish classes for the task of classification and also have potential for being adapted to unsupervised data analysis. Thus, despite its susceptibility to outliers and noise, a k-means clustering [24] was chosen as a base for finding groupings of the instances in the embedded space. The number of clusters is always set to the number of classes in the data. Each of the clusters was later assigned to the closest classes using Hungarian algorithm. Finally, a balanced accuracy metric is used as a fitness score for each of the individual in order to become independent of the size of each class in the original data. The goal of the method is maximizing the fitness score, which should lead to higher separability of the classes and thus to more convincing visualizations. The predictions for the unseen instances were made based on the distance to the closest centroid.

*Allowable Operators.* The primitive set of a GP syntax tree for ManiGP comprises various mathematical operations, random ephemeral constants and terminals equal to 0 or 1. The allowable operations are addition, subtraction, multiplication, safe division (to prevent division by 0), modulo, modulo-2 summation; equal, not equal, less than, greater than comparisons; logical and, or, xor and not; bitwise and, or and xor; abs, factorial, power, logarithm a of b, permutation, choose; left, right, min and max. A maximal permissible height is used as a bloat control to prevent the excessive growth of trees.

*The Method.* The concept of the method is visualized on the example of the appendicitis dataset in Fig. 1. Each point represents an instance of the original dataset, which was mapped into a new space. The color of the point corresponds to its original class. The coordinates of the new space are determined by syntax



**Fig. 1.** A concept of ManiGP. A population of the algorithm contains evolving tuples of trees that create a manifold to separate classes.

trees which were evolved using GP. Each of the trees is responsible for constructing a single embedding in a low-dimensional space. This means that the number of trees in the individual is equal to the dimensionality of the reduced dataset. In this example, the balanced accuracy on the training dataset is equal to 93.7%, whereas on the testing data 80.9% (both were plotted in the same chart).

The evolution process in ManiGP is set up as follows.

1. A number of best individuals from the previous population is selected to the next generation. The selection operator sorts the individuals by their fitness function values in descending order and then chooses the ones from the beginning of the list. The number of individuals chosen is equal to the size of the initial population.
2. The population is randomly shuffled and individuals are assigned to pairs based on their position in the population. Specifically, the first individual on the list is paired with the second one, the third one with the fourth one and so on.
3. A crossover operator is applied on each pair of individuals with a certain probability. Corresponding coordinates of the trees are crossed over using a

one point crossover operator which chooses random nodes in each of the two trees and swaps the subtrees that have a root in that randomly chosen node creating two new trees. In this way, two new individuals are created. If all of the trees comprising the new individual are of admissible height, then the individual is added to the population. The resulting population contains both parents and children.

4. Each individual in the population - including the children created with a crossover operator - is mutated with a certain probability. Each tree of an individual is mutated using the mutation operator. Within each of the trees, the operator chooses randomly a subtree and swaps it with a randomly generated one.
5. A bloat control check (i.e. verification of the height of the tree) is performed before admitting it to the population.
6. Each of the instances is mapped into the new space using trees. On the reduced space, a k-means clustering is run to find groupings of instances. The number of clusters used in k-means is equal to the number of classes. As clustering does not use information on the class labels, an additional assignment needs to be made in order to match clusters to classes. This is performed using Hungarian algorithm [26].
7. The fitness function of each individual is defined as a balanced accuracy across all classes.
8. The evolution stops after a predefined number of iterations. If the predefined number of iterations is not met, the next population is created using selection of best individuals<sup>2</sup>. Otherwise, the instances of the data are transformed using the individual with the highest fitness.

*Implementation.* ManiGP was implemented in Python using Distributed Evolutionary Algorithms in Python (DEAP) framework [13,32] and other popular Python libraries such as NumPy, pandas, and sklearn.

*Related Work.* GP and nearest centroid classifiers were shown to be powerful tools for improving classification accuracy and feature selection [1]. Although in some ways similar, the approach taken in this study applies GP and K-means in a different context than in the research conducted by Al-Madi et al., which focused more on improving classification results derived from GP with a k-means based algorithm. The design of the method used in this paper is very closely related to M4GP, the method of dimensionality reduction by La Cava et al. [20] with four major differences. Firstly, the representation of the individual is different. ManiGP represents individual as multi-tree structure, whereas M4GP use a stack-based representation. Secondly, ManiGP uses the simplest possible strategies (e.g. selection of best individuals), whereas M4GP uses advanced selection operators, such as epsilon lexicase selection and Age-fitness Pareto survival.

---

<sup>2</sup> Empirical tests have surprisingly shown the superior performance of this technique in comparison to tournament selection.

Thirdly, the fitness objective is different [20]. Finally, ManiGP presets the resulting dimensionality, whereas M4GP dynamically determines it in the course of evolution.

It needs to be emphasized, that ManiGP by no means was designed to be a fully fledged classifier. It emerged as an experiment during designing a fair methodology for the comparison of the methods. Our hypothesis was that it is possible to design a method that would work at least as well as the state-of-the-art methods for a large set of problems, if it is given a leverage and optimizes the same metric used later in benchmarking. This, in our opinion, not only puts the method in favourable position over the others, but also adds an interesting layer in the discussion on existence of the objective comparison methodology.

### 2.3 Datasets

Penn Machine Learning Benchmark (PMLB) is one of the largest available collections of publicly available dataset for classification and regression. Multiple problems included in the benchmark suite were pulled from popular repositories, such as UCI, OpenML or Kaggle and transformed into a machine learning friendly format [27]. In this paper our focus was classification task, for which we pulled 166 classification datasets. The following 11 datasets were excluded from the comparison because of issues with convergence or running times exceeding 168 h for at least 2 methods: ‘adult’, ‘connect-4’, ‘fars’, ‘kddcup’, ‘krkopt’, ‘letter’, ‘magic’, ‘mnist’, ‘poker’, ‘shuttle’, and ‘sleep’. As the result, the collection included in this analysis contained 155 different datasets.

### 2.4 Methodology of Comparison

One of the greatest challenges in benchmarking manifold learning methods is the fact that not every method supports mapping of the previously unseen data. For example a very popular t-SNE learns a non-parametric mapping, which means there is no function learned that would map the point from the input space to the embedded one. A similar issue involves also MDS and Spectral Embeddings, two other popular manifold learning techniques. This has far-reaching consequences, as the performance of the methods on test data can’t be reported for the method. It also means that cross-validation can’t be used for finding the optimal setting of the input parameters.

There are two potential ways of getting around this fact, but none of them could be considered a good strategy. First, a regressor could be proposed which for each of the samples in the test data finds the closest sample (or samples) and assigns its score (or performs a form of majority voting). Unfortunately, this may be erroneous for some of the datasets, which contain categorical, or ordinal values. The second approach is learning manifold on full data and reporting the score only for the test points. In this scenario, however, testing data is used in training, which creates additional bias. After consideration we have decided not to report the performance of aforementioned methods on test data. This resulted in abandoning k-fold cross validation in favor of running method with 5 different



random seeds with a grid of the parameters and using the best performance within training data for testing.

The second challenge was proposing an objective metric for assessing performance. After learning the manifold, multiple existing classifiers could be adapted and trained on the data with reduced dimensionality. Our choice was using k-means, a popular clustering technique. This unsupervised technique could be considered as unbiased measure of separability of the classes as it does not use real class labels for the analysis. The correct assignments to the classes are handled using Hungarian algorithm in similar way as in Orzechowski and Boryczko [28]. Although we are aware that k-means isn't perfect, as doesn't handle well clusters of different densities, irregular shapes as well as outliers, the simplicity and linear division of the classes in the embedded space were the reasons why k-means was chosen as the base for the analysis.

The workflow for benchmarking manifold learning method was constructed as follows:

1. For each of the datasets 5 different randomly initialized seeds were selected. The seeds were used for splitting each of the dataset to train and testing sets with proportion 75%–25%. Stratified split was used to maintain the proportion of the representatives of each class in both training and testing set and RobustScaler was used for preprocessing data.
2. Each manifold learning method was initialized with the same random seed. For each of the methods we have used a grid of input parameters. The methods with their parameters are presented in Table 1.
3. In order to better understand separability of the classes, the resulting two dimensional embedding from each manifold learning methods served as a base for performing k-means clustering with the number of clusters equal to the number of classes.
4. Hungarian algorithm [26] was used to assign class label to the clusters.
5. For the methods that allow projection of a test set, the distance to the center of the nearest cluster was used to assign label to the point.
6. As class imbalance is an issue with different datasets, a popular metric called balanced accuracy was used as a measure of performance.
7. For each of the methods and a given seed the highest balanced accuracy score was considered across a grid of the parameters.

### 3 Results

For benchmarking we have included all of the manifold learning methods described in the previous chapter. For the sake of clarity, we have narrowed down our analysis to manifold learning methods only and decided not to include linear dimensionality reduction techniques, as they remain out of scope for this paper. The detailed results of the analysis have been added to our project repository<sup>3</sup>.

---

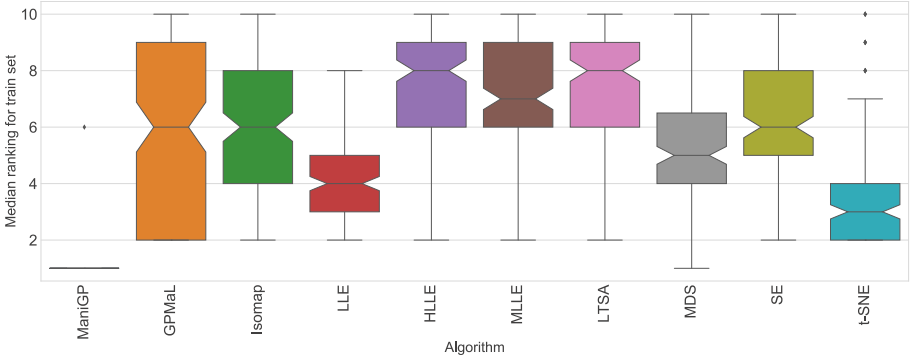
<sup>3</sup> <https://github.com/athril/manigp/>.

**Table 1.** Parameters settings of the analyzed methods. The names of the parameters refer to scikit-learn implementation.

Algorithm	Parameter	Values
ManiGP	'(xover rate, mut rate)'	{(0.9,0.1), (0.5,0.5), (0.1,0.9)}
	'generations'	500
	'pop_size'	100
GP-MaL	'generations'	1000
	pop_size	1024
Isomap	'n_neighbors'	[5,6,7,8,9,10,15,20],
	'eigen_solver'	['arpack','dense'],
LLE	'reg'	[1e-4, 1e-3, 0.001, 0.1, 1, 10]
	'n_neighbors'	[5,6,7,8,9,10,15,20]
	'eigen_solver'	['dense']
Hessian	'reg'	[1e-4, 1e-3, 0.001, 0.1, 1, 10]
	'n_neighbors'	[5,6,7,8,9,10,15,20]
	'eigen_solver'	['dense']
Modified LLE	'reg'	[1e-4, 1e-3, 0.001, 0.1, 1, 10]
	'n_neighbors'	[5,6,7,8,9,10,15,20]
	'eigen_solver'	['dense']
LTSA	'reg'	[1e-4, 1e-3, 0.001, 0.1, 1, 10]
	'n_neighbors'	[5,6,7,8,9,10,15,20]
	'eigen_solver'	['dense']
MDS	'max_iter'	[300,500]
	'metric'	[True,False]
	'dissimilarity'	['euclidean']
Spectral	'affinity'	['nearest_neighbors']
	'n_neighbors'	[5,6,7,8,9,10]
	'eigen_solver'	[None, 'arpack', 'lobpcg', 'amg']
		or
	'affinity'	['rbf']
	'eigen_solver'	[None, 'arpack', 'lobpcg', 'amg']
t-SNE	'perplexity'	[5,10,15,20,25,30,35,40,45,50]
	'n_iter'	[1000,5000]

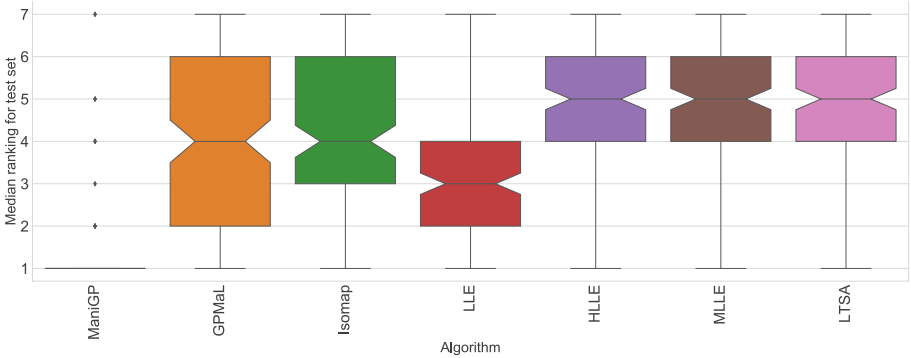
All of the methods were run starting from 5 different seeds on a reduced PMLB benchmark suite with grid of the parameters presented in Table 1. For each of the seeds, the setting with the highest balanced accuracy score on the training data was chosen for the subsequent analysis. The median score across the seeds served for ranking the methods.

The performance for the task of classification on the training dataset is presented in Fig. 2. This analysis compares the methods on separating labeled data, what makes it a good benchmark for providing clarity of visualization.



**Fig. 2.** The ranking of the manifold learning on the training dataset. The lower, the better.

Three of the methods, namely MDS and SE and t-SNE are not suitable for making predictions for the unseen data. Thus, they were removed from the assessment in the test data, which is presented in Fig. 3. This analysis shows the potential of the methods to be used as dimensionality reduction techniques.



**Fig. 3.** The ranking of the manifold learning on the testing dataset. The lower, the better. Some of the methods (PCA, t-SNE and SE) had to be excluded as they don't provide the mapping that would allow to make predictions for unseen data.

To inspect the significant differences, we ran a Friedman test using balanced accuracy scores for the test data. As the number of datasets is large, the analysis has higher statistical power. P-values less than 0.005 suggest significant differences between the methods. Post-hoc pairwise tests are presented in Table 2.

*Summary.* Ranking the methods on the training data shows ManiGP as the leader, followed by t-SNE and LLE. The other methods performed visibly worse, with a few exceptions of MDS and GP-MaL.

**Table 2.** Friedman’s asymptotic general symmetry test. P-values lower than 0.005 were boldfaced.

	GPMaL	Isomap	LLE	HLLE	MLE	LTSA
ManiGP	<b>2.2e−16</b>	<b>2.2e−16</b>	<b>3.3e−14</b>	<b>2.2e−16</b>	<b>2.2e−16</b>	<b>2.2e−16</b>
GPMaL	–	0.99	2.0e−02	6.6e−03	4.1e−02	2.9e−02
Isomap	–	–	<b>4.8e−03</b>	0.02	0.12	0.09
LLE	–	–	–	<b>4.6e−10</b>	<b>3.5e−09</b>	<b>2.0e−09</b>
HLLE	–	–	–	–	0.99	0.99
MLE	–	–	–	–	–	0.99

Considering the results on testing data, the clear leader is again ManiGP, which statistically outperformed each of the considered methods. The runner up is LLE, which significantly outperformed all methods, but ManiGP and GPMaL. The remaining comparisons showed no significant differences.

Considering running times, both GP methods were a couple of orders of magnitude slower than the rest of approaches. Additionally, ManiGP was far slower than GP-MaL. GP-MaL was run for 1000 iterations with population of 1024 and was faster than ManiGP, which was run for 500 iterations with population of 100. The other methods had comparable running times, counted in seconds/minutes instead of hours/dozens of hours, as was the case with GP approaches. Notice however, that for the fairness all the methods were run with a single thread and GP methods could be run in parallel. Among the non-GP methods, MDS and t-SNE were an order of magnitude slower than the remaining approaches.

## 4 Conclusions

In this paper a comparison of 10 different manifold learning methods was performed on a large collection of real world datasets. The comparison was performed using rigorous machine learning standards [21, 29] over a large collection of 155 datasets from PMLB [27]. The study aimed at empirical verification of how well the manifold learning approaches separate instances for the task of classification. All the source code of our analysis is open source and publicly available. Two of the methods included in the study were based on genetic programming: GP-MaL and ManiGP. To perform this study, we have created an open source framework for benchmarking of manifold learning techniques, as well as created a wrapper for GP-MaL.

Due to the nature of some of the manifold learning methods, such as MDS, SE and t-SNE, which are unable to transform an unseen instance, our benchmark is split into two parts: the training part, which could be considered benchmarking of potential of manifold learning techniques for visualization, and the testing part, in which the potential of the methods to serve as dimensionality reduction techniques could be more thoroughly assessed.

As for the visualization purposes, we have confirmed that ManiGP – an experimental technique proposed in this paper – delivers the most separable charts in comparison to any other manifold learning technique for the vast majority of datasets. Due to its excessive time however, its practical use remains highly limited. Among the methods with reasonable execution time, t-SNE can be considered the first choice and LLE the second.

Taking into account potential of using the methods toward further data analysis, ManiGP provides by far more separable results than any other manifold learning technique included in the comparison. The obtained results of ManiGP in terms of balanced accuracy were also significantly better compared to any other method included in the study. Another advantage of the method is interpretability, as it uses two (or more) syntax trees with arithmetic-logical operations. This makes the method easily adaptable to an unseen data. The greatest downside of the method is excessive running time, which we consider not feasible for larger datasets. On the other hand, the method wasn't optimized for the purpose of this study and a couple possible improvements might be taken out of the box. The further experiments suggest that the results are also comparable with some of the leading machine learning classifiers in the field. Once again, we would emphasize that the development of a method wasn't the major goal of this study and this method shouldn't be considered a valid classifier, which could be used in production.

Our analysis showed that GP-based approaches are capable of delivering fully interpretable, better separable and even significantly better results than multiple well established manifold learning approaches. Despite its excessive run times for larger datasets, ManiGP proposed in this paper outperformed other methods, although it used only very basic evolutionary techniques and evaluated over 20 times less individuals than GP-MaL. Among non-GP methods, considering potential of using a method to predict an unseen data, we believe that LLE remains a good trade off between the speed and performance, as it offers superior results to multiple other manifold learning methods within reasonable time frame. For the pure aspect of visualization, t-SNE remains a convenient approach, as it provides better separation.

Finally, we would like to elaborate more on the fairness of the presented comparison. One of the aspects of our study was proposing a method that deliberately optimizes the same score used later for evaluation. We have discovered that this approach performed statistically better than the approaches based on other merits. The question that should be asked is how objective could any comparison be considered, assuming that one method intentionally used the evaluation metric. The answer is not straightforward; it is necessary however that the designers of the study put as much effort as possible into designing as objective a comparison as possible with support for their results from multiple tests.

In summary, we believe that this paper set new standards in benchmarking manifold learning techniques and addresses not only their visualization potential, but far beyond. An important findings in this study are as follows: (1) not every manifold learning method provides the possibility to analyze unseen data, (2) the

advantage of GP in manifold learning lies in providing interpretable results, (3) because of excessive running time of GP-based methods, their potential of creating convincing visualizations is highly limited, (4) application of GP in manifold learning is justified if the focus of the study is interpretability of the model and excessive running time will later be rewarded by instantaneous testing, (5) the fairness of benchmarking requires further research. Even those benchmarks that seem to be objective may be exploited by the methods that purposefully exploit the design of the study.

**Acknowledgements.** This research was supported in part by PL-Grid Infrastructure and by National Institutes of Health (NIH) grant LM012601. The authors would like to thank Dr. Andrew Lensen from Victoria University of Wellington for his help in running GP-MaL.

## References

1. Al-Madi, N., Ludwig, S.A.: Improving genetic programming classification for binary and multiclass datasets. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 166–173. IEEE (2013)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–591 (2002)
3. Bhowan, U., Zhang, M., Johnston, M.: Genetic programming for classification with unbalanced data. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 1–13. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-12148-7\\_1](https://doi.org/10.1007/978-3-642-12148-7_1)
4. Borg, I., Groenen, P.: Modern multidimensional scaling: theory and applications. *J. Educ. Meas.* **40**(3), 277–280 (2003)
5. Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: 2010 20th International Conference on Pattern Recognition, pp. 3121–3124. IEEE (2010)
6. Cano, A., Ventura, S., Cios, K.J.: Multi-objective genetic programming for feature extraction and data visualization. *Soft. Comput.* **21**(8), 2069–2089 (2015). <https://doi.org/10.1007/s00500-015-1907-y>
7. Coifman, R.R., Lafon, S.: Diffusion maps. *Appl. Comput. Harmonic Anal.* **21**(1), 5–30 (2006)
8. Cunningham, J.P., Ghahramani, Z.: Linear dimensionality reduction: survey, insights, and generalizations. *J. Mach. Learn. Res.* **16**(1), 2859–2900 (2015)
9. De Backer, S., Naud, A., Scheunders, P.: Non-linear dimensionality reduction techniques for unsupervised feature extraction. *Pattern Recogn. Lett.* **19**(8), 711–720 (1998)
10. Donoho, D.L., Grimes, C.: Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proc. Nat. Acad. Sci.* **100**(10), 5591–5596 (2003)
11. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **40**(2), 121–144 (2009)
12. Fisher, R.A.: The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **7**(2), 179–188 (1936)

13. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. *J. Mach. Learn. Res.* **13**, 2171–2175 (2012)
14. Gisbrecht, A., Hammer, B.: Data visualization by nonlinear dimensionality reduction. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* **5**(2), 51–73 (2015)
15. Guo, H., Zhang, Q., Nandi, A.K.: Feature extraction and dimensionality reduction by genetic programming based on the fisher criterion. *Expert Syst.* **25**(5), 444–459 (2008)
16. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature Extraction: Foundations and Applications. STUDDFUZZ, vol. 207. Springer, Heidelberg (2006). <https://doi.org/10.1007/978-3-540-35488-8>
17. Hotelling, H.: Relations between two sets of variates. In: Kotz, S., Johnson, N.L. (eds.) *Breakthroughs in Statistics. SSS*, pp. 162–190. Springer, New York (1992). [https://doi.org/10.1007/978-1-4612-4380-9\\_14](https://doi.org/10.1007/978-1-4612-4380-9_14)
18. Khalid, S., Khalil, T., Nasreen, S.: A survey of feature selection and feature extraction techniques in machine learning. In: 2014 Science and Information Conference, pp. 372–378. IEEE (2014)
19. Kullback, S., Leibler, R.A.: On information and sufficiency. *Ann. Math. Stat.* **22**(1), 79–86 (1951)
20. La Cava, W., Silva, S., Danai, K., Spector, L., Vanneschi, L., Moore, J.H.: Multidimensional genetic programming for multiclass classification. *Swarm Evol. Comput.* **44**, 260–272 (2019)
21. La Cava, W., Williams, H., Fu, W., Moore, J.H.: Evaluating recommender systems for AI-driven data science. arXiv preprint [arXiv:1905.09205](https://arxiv.org/abs/1905.09205) (2019)
22. Lee, J.A., Verleysen, M.: *Nonlinear Dimensionality Reduction*. Springer, New York (2007). <https://doi.org/10.1007/978-0-387-39351-3>
23. Lensen, A., Xue, B., Zhang, M.: Can genetic programming do manifold learning too? In: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (eds.) *EuroGP 2019. LNCS*, vol. 11451, pp. 114–130. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-16670-0\\_8](https://doi.org/10.1007/978-3-030-16670-0_8)
24. Lloyd, S.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982)
25. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**(Nov), 2579–2605 (2008)
26. Munkres, J.: Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* **5**(1), 32–38 (1957)
27. Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R.J., Moore, J.H.: PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.* **10**(1), 36 (2017)
28. Orzechowski, P., Boryczko, K.: Parallel approach for visual clustering of protein databases. *Comput. Inform.* **29**(6+), 1221–1231 (2012)
29. Orzechowski, P., La Cava, W., Moore, J.H.: Where are we now?: a large benchmark study of recent symbolic regression methods. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1183–1190. ACM (2018)
30. Pearson, K.: LIII. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **2**(11), 559–572 (1901)
31. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
32. Rainville, D., Fortin, F.A., Gardner, M.A., Parizeau, M., Gagné, C., et al.: DEAP: a python framework for evolutionary algorithms. In: *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, pp. 85–92. ACM (2012)

33. Rao, C.R.: The utilization of multiple measurements in problems of biological classification. *J. Roy. Stat. Soc.: Ser. B (Methodol.)* **10**(2), 159–203 (1948)
34. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., Jain, A.K.: Dimensionality reduction using genetic algorithms. *IEEE Trans. Evol. Comput.* **4**(2), 164–171 (2000)
35. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**(5500), 2323–2326 (2000)
36. Sorzano, C.O.S., Vargas, J., Montano, A.P.: A survey of dimensionality reduction techniques. arXiv preprint [arXiv:1403.2877](https://arxiv.org/abs/1403.2877) (2014)
37. Spearman, C.: General intelligence objectively determined and measured. *Am. J. Psychol.* **15**, 107–197 (1904)
38. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science* **290**(5500), 2319–2323 (2000)
39. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. *J. Roy. Stat. Soc.: Ser. B (Stat. Methodol.)* **61**(3), 611–622 (1999)
40. Torgerson, W.S.: Multidimensional scaling: I. Theory and method. *Psychometrika* **17**(4), 401–419 (1952). <https://doi.org/10.1007/BF02288916>
41. Vural, E., Guillemot, C.: Out-of-sample generalizations for supervised manifold learning for classification. *IEEE Trans. Image Process.* **25**(3), 1410–1424 (2016)
42. Vural, E., Guillemot, C.: A study of the classification of low-dimensional data with supervised manifold learning. *J. Mach. Learn. Res.* **18**(1), 5741–5795 (2017)
43. Wattenberg, M., Viégas, F., Johnson, I.: How to use t-SNE effectively. *Distill* (2016). <https://doi.org/10.23915/distill.00002>. <http://distill.pub/2016/misread-tsne>
44. Weinberger, K.Q., Saul, L.K.: An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In: *AAAI*, vol. 6, pp. 1683–1686 (2006)
45. Yao, H., Tian, L.: A genetic-algorithm-based selective principal component analysis (GA-SPCA) method for high-dimensional data feature extraction. *IEEE Trans. Geosci. Remote Sens.* **41**(6), 1469–1478 (2003)
46. Zhang, Z., Wang, J.: MLLLE: modified locally linear embedding using multiple weights. In: *Advances in Neural Information Processing Systems*, pp. 1593–1600 (2007)
47. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comput.* **26**(1), 313–338 (2004)
48. Zhao, D., Lin, Z., Tang, X.: Laplacian PCA and its applications. In: *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. IEEE (2007)