Proximity Forests on Manifolds

Editors: List of editors' names

Abstract

Recent work has focused on machine learning methods for manifold-valued data. Certain manifolds admit a notion of pairwise distance, allowing the k-nearest neighbors (KNN) classifier to be used with manifold-valued data. However, the computation of pairwise manifold distances is often computationally expensive, even with modern geometric machine learning software. In this work, we generalize the Proximity Forest (PF) model, originally designed as a time series distance-based classifier, to accommodate more general distance measures. We show that the PF model scales more favorably than KNN for large sample sizes in high dimensions. Given recent applications of the PF model, this also introduces supervised outlier detection and imputation for manifold-valued data. Additionally, we integrate the differential geometry software package for Maple to obtain a reduction in computational costs for certain pairwise manifold distances. We also introduce a method for estimating manifold distances on level sets.

Keywords: Machine Learning on Manifolds, Geometric AI, Learning on Graphs, Scalable Learning

1. Introduction

Machine learning on manifolds is a topic of growing interest, and many machine learning methods have recently been developed to analyze manifold-valued data (Miolane et al., 2020). While most machine learning techniques are built using operations in a vector space, manifolds generally do not have the structure of a vector space and require a different set of mathematical tools. Some manifolds, such as Riemannian manifolds, have the structure of a metric space. Therefore, conventional distance-based machine learning algorithms, such as the k-nearest neighbors (KNN) algorithm, can be used for manifold-valued data, provided that a suitable distance measure can be computed.

The KNN classifier is not without its drawbacks, however. To obtain model predictions on a test dataset from a training dataset, a distance must be computed between each test sample and each training sample, which operation has complexity $\mathcal{O}(kn^2)$, where k is the complexity of the distance measure and n is the number of data instances. For manifold-valued data, the geodesic distance is often used, which is a computationally-expensive operation, making the KNN classifier infeasible for large datasets.

A distance-based time series classifier known as a Proximity Forest (PF) was proposed in Lucas et al. (2018). The PF model is designed to be similar to a Random Forest (RF) model, though "proximity trees" are used instead of decision trees. The primary difference between a proximity tree and a decision tree is that data instances travel down a proximity tree according to distances rather than feature values. The original PF model was designed to use time series distance(s). However, it can be readily adapted, as we have done, to work with any distance measure.

One benefit of the PF model is that it has a computational complexity of $\mathcal{O}(kn \log n)$ (Lucas et al., 2018), which is lower than that of a KNN classifier. In addition, the PF model can be used to compute RF-based proximity graphs in which the nodes correspond to individual data instances. Thus, the PF model can be used to learn a graph in a supervised manner. The learned proximity graph can be used in subsequent applications such as intra-class outlier detection, supervised dimensionality reduction, and supervised missing data imputation (Rhodes et al., 2023; Shaw et al., 2025). While KNN can also be used for all of these tasks, RF-based proximities are substantially different.

In addition to using manifold distances for classification, we also consider the problem of estimating manifold distances to begin with. For Riemannian manifolds, a Riemannian metric tensor induces a distance measure, endowing the manifold with the structure of a metric space. Meanwhile, there is an ever-growing list of methods by which a metric tensor can be estimated (Arvanitidis et al., 2017; Beik-Mohammadi et al., 2023; Chen et al., 2020; Hauberg et al., 2012; Jørgensen and Hauberg, 2021; Rozo et al., 2025; Lebanon, 2002; Lee et al., 2022; Sun et al., 2025). However, the distances induced by these metric tensors are difficult to compute. As we demonstrate experimentally, current geometric machine learning software (Miolane et al., 2020) is unable to compute pairwise distances within a practical timeframe for implementing the KNN classifier. To address this limitation, our generalization of the PF model is able to integrate Maple, thereby leveraging the extensive capabilities of the *differential geometry software package* (Anderson and Torre, 2022), which we show experimentally to achieve gains in computational efficiency over existing software.[1]

2. Manifold Distance Estimation for Level Sets via Gradient Descent

Consider an embedded submanifold of $\mathbb{R}^n$ of dimension m characterized as a level set of a smooth function $F : \mathbb{R}^n \to \mathbb{R}^q$, where $q = n - m$. We first calculate the Jacobian matrix $J_{q\times n}$. Then, we identify m vector fields $\{X_i\}_{i=1}^m$ which each annihilate the components of F (that is, satisfy $X_i(f_j) = 0$ for each component f_j of F–a more detailed treatment of vector fields and flows is given in Appendix B) by calculating a basis, with smooth functions as components, for the nullspace of the Jacobian matrix. The next step is to obtain the flow parameters of each vector field, defined as functions $\theta_i(\mathbf{x})$ for which $X_i(\theta_i) = 1$.

Next, given the points p_0 and p_1 for which the distance is to be estimated (expressed in the local coordinates, assuming they can be expressed in the same coordinate chart), we define the function $\mathcal{L}$ to be the square error of θ and p_1:

$$\mathcal{L}(\theta) = (p_1 - \theta)^2.$$

This function has a single global/local minimizer of p_1. The function can also be expressed in the ambient coordinates,

$$\mathcal{L}(\mathbf{x}) = (p_1(\mathbf{x}) - \theta(\mathbf{x}))^2,$$

so that the Riemannian gradient of $\mathcal{L}$ in the local coordinates coincides with the ordinary gradient of $\mathcal{L}(\mathbf{x})$. Thus, optimization of $\mathcal{L}(\mathbf{x})$ can be done using gradient descent, initializing at $p_0(\mathbf{x})$.

1. We should note that the purpose of using Maple is not necessarily to advocate for the widespread use of Maple in machine learning tasks: the primary disadvantage of Maple is that it is commercial, not open source software. However, the performance gained by using the *differential geometry software package* at the very least highlights the need for improved computational tools in geometric machine learning.

However, after a single iteration of gradient descent, the updated point $\tilde{p}_{0_1}$ will not generally lie on the level set manifold. Therefore, after each iteration of gradient descent, we obtain a "fixed" update p_{0_i} by finding the minimizer of the following function using a (nested) gradient descent algorithm initializing at $\tilde{p}_{0_i}$:

$$\tilde{\mathcal{L}} = (F - \mathbf{0})^2,$$

assuming the level set can be given as $F = \mathbf{0}$.

The combined gradient descent algorithms provides a series of points which begin at p_0 and continue approximately on the level set until a final point p_{0_b} is reached, where b is the number of iterations of the outer gradient descent algorithm. When $\mathcal{L}$ is sufficiently small, p_{0_b} is near the point p_1 in terms of the manifold distance. At each iteration, the (fixed) updates $\{p_{0_i}\}_{i=0}^{b}$ are stored, and the straight-line (Euclidean) distance between subsequent points are computed and added to approximate the manifold distance.

3. PF vs KNN at Scale

We design a three-class classification experiment using multivariate Gaussians, one for each class, and where the dimension of the data is 100. The sample size takes values in the following set (with a 50/50 test/train split): $\{10^3, 10^4, 10^5, 5 \cdot 10^5, 10^6, 2 \cdot 10^6\}$. For each sample size, we record the time in seconds needed to train and obtain predictions for both a KNN model with $k = 5$ as well as a PF model with 11 trees and $r = 5$. The Euclidean distance is used in both cases.

The result of this experiment suggests that the computational benefit of the PF model is not realized unless the sample size exceeds 10^5.[2] This suggests that the PF model is a more efficient choice than KNN for large datasets in high dimensions. With smaller datasets, particularly in low dimensions, KNN may be a more suitable choice in terms of efficiency.

4. Comparing Manifold Distance Approximation Methods on a Paraboloid

In general, though *geomstats* has efficient methods for computing distances for several specific manifolds, it is unlikely that a learned metric tensor would correspond with a metric for which the geodesic distances have been examined in advance. Therefore, we find it necessary to compare Maple to the feature of *geomstats* which allows one to "implement your own geometry" to design a custom metric tensor. We generate 1024 data points, half of which are for training and half of which are for testing/evaluation, on a subset of a paraboloid described by $x^2 + y^2 - z = 0$ which, in addition to having a level set description, can be described as the image of $\Phi(u, v) = (u, v, u^2 + v^2)$ to give a pullback metric in the coordinates (u, v).

In *geomstats*, the manifold distance is calculated using the *distance* method of the pullback metric. In Maple, this distance is estimated using the authors' code, leveraging the *differential geometry software package* and Maple's numeric *dsolve* command. We call

2. A graphic depicting these results is given in Appendix A.

this distance d for brevity. We also compare to the method proposed in Section 2 using $F = z - x^2 - y^2$ with 20 outer epochs and 10 inner epochs, which distance we refer to as $\tilde{d}$ for brevity.

We first compute the distances between the first test point and all 512 training points, which is needed to classify the test point according to KNN. Using *geomstats*, this task takes approximately 3128 seconds, which is approximately 52 minutes. The same number of d-distances takes approximately 18.1 seconds, which is roughly a 100-fold reduction in computation time compared with *geomstats*. Again in Maple, we estimate $\tilde{d}$-distances, obtaining an approximate time of 10.2 seconds, though we note that this timing depends heavily on the choices of the number of inner or outer epochs.

Having established that the two Maple-based distances are faster, we wish to compare computation times for the KNN and PF models using them. For KNN, we find that computing all 512 d-distances for each of the 512 test instances takes approximately 8579 seconds, or roughly 2.4 hours. On the other hand, a PF model is trained in approximately $11,402$ seconds, which is approximately 3.2 hours. For $\tilde{d}$-distances, the KNN distances and PF model require approximately 4797 and 8578 seconds, respectively. Using a 50/50 test/training split, we note that the test accuracy of the PF model using d-distances is 98.8%, while the test accuracy using $\tilde{d}$-distances is 99.4%. This suggests that $\tilde{d}$-distances may be a well-suited manifold distance estimation technique for distance-based classifiers, notwithstanding any gaps between these distances and “true” manifold distances.

5. Conclusion

We have generalized the previously-given classifier known as Proximity Forests (PF) to obtain a distance-based classifier for manifold-valued data. Our work suggests that the PF model scales more favorably for large samples ($> 10^5$) in high dimensions. Our work also suggests that current methods for computing manifold-valued distances can be improved, with code in Maple vastly outperforming state-of-the-art software in geometric machine learning in terms of computation time. However, as the KNN and PF model merely *compare* distances, exact manifold distances are not strictly needed, allowing one to use a computationally inexpensive (by comparison) substitute distance, where one can be identified, such as our proposed method for manifolds characterized by level sets. Future work includes further refinement of our modified implementation of the PF model to explicitly accommodate various data types. We also seek to apply the generalized PF model to additional data types, such as text data, in a future endeavor.

References

I. Anderson and C. Torre. The differential geometry software project, 2022. URL https://digitalcommons.usu.edu/dg/.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv: Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:51780574.

Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo. Reactive motion generation on learned riemannian manifolds. *The International Journal of Robotics Research*, 42(10):729–754, 2023. doi: 10.1177/02783649231193046. URL https://doi.org/10.1177/02783649231193046.

Leo Breiman and Adele Cutler. Random forests – classification description, 2001. URL https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. Accessed: Jul. 2, 2025.

Nutan Chen, Alexej Klushyn, Francesco Ferroni, Justin Bayer, and Patrick Van Der Smagt. Learning flat latent manifolds with vaes. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

A. Cutler, D. R. Cutler, and J. R. Stevens. *Random Forests*, pages 157–175. Springer US, Boston, MA, 2012. ISBN 978-1-4419-9326-7. doi: 10.1007/978-1-4419-9326-7_5.

Søren Hauberg, Oren Freifeld, and Michael J. Black. A geometric take on metric learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2024–2032, Red Hook, NY, USA, 2012. Curran Associates Inc.

Allison Marie Horst, Alison Presmanes Hill, and Kristen B Gorman. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020. URL https://allisonhorst.github.io/palmerpenguins/. R package version 0.1.0.

Martin Jørgensen and Soren Hauberg. Isometric gaussian process latent variable model for dissimilarity data. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5127–5136. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jorgensen21a.html.

Guy Lebanon. Learning riemannian metrics. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, page 362–369, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 0127056645.

John M. Lee. *Introduction to Smooth Manifolds*. Springer New York, NY, 2012.

Yonghyeon Lee, Sangwoong Yoon, Minjun Son, and {Frank C.} Park. Regularized autoencoders for isometric representation learning. In *10th International Conference on Learning Representations*, 2022.

Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Abbas Zaidi, Bart Goethals, François Petitjean, and Geoffrey I. Webb. Proximity forest: An effective and scalable distance-based classifier for time series. *CoRR*, abs/1808.10594, 2018. URL http://arxiv.org/abs/1808.10594.

Nina Miolane, Alice Le Brigant, Johan Mathe, Benjamin Hou, Nicolas Guigui, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Bernhard Kainz, Claire

Donnat, Susan Holmes, and Xavier Pennec. Geomstats: A python package for riemannian geometry in machine learning, 2020. URL https://arxiv.org/abs/2004.04667.

Kevin Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel Burkhardt, William Chen, Kristina Yim, Antonia Elzen, Matthew Hirn, Ronald Coifman, Natalia Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37:1482 – 1492, 2019. URL https://doi.org/10.1038/s41587-019-0336-3.

John Paparrizos, Haojun Li, Fan Yang, Kaize Wu, Jens d'Hondt, and Odysseas Papapetrou. A survey on time-series distance measures, 12 2024.

Jake S. Rhodes. Supervised manifold learning via random forest geometry-preserving proximities*. In *2023 International Conference on Sampling Theory and Applications (SampTA)*, pages 1–5, 2023. doi: 10.1109/SampTA59647.2023.10301399.

Jake S. Rhodes, Adele Cutler, Guy Wolf, and Kevin R. Moon. Random forest-based diffusion information geometry for supervised visualization and data exploration. In *2021 IEEE Statistical Signal Processing Workshop (SSP)*, pages 331–335, 2021. doi: 10.1109/SSP49050.2021.9513749.

Jake S Rhodes, Adele Cutler, and Kevin R Moon. Geometry-and accuracy-preserving random forest proximities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Leonel Rozo, Miguel González-Duque, Noémie Jaquier, and Søren Hauberg. Riemann^2: Learning riemannian submanifolds from riemannian data. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 3097–3105. PMLR, 03–05 May 2025. URL https://proceedings.mlr.press/v258/rozo25a.html.

Ben Shaw, Jake Rhodes, Soukaina Filali Boubrahimi, and Kevin R. Moon. Forest proximities for time series, 2025. URL https://arxiv.org/abs/2410.03098.

Xingzhi Sun, Danqi Liao, Kincaid MacDonald, Yanlei Zhang, Chen Liu, Guillaume Huguet, Guy Wolf, Ian Adelstein, Tim G. J. Rudner, and Smita Krishnaswamy. Geometry-aware generative autoencoders for warped riemannian metric learning and generative modeling on data manifolds, 2025. URL https://arxiv.org/abs/2410.12779.

Appendix A. Graphical Computational Comparison of PF and KNN

With regard to the experiment in Section 3, Figure 1 illustrates the stated conclusion, which is that the computational advantage of the PF model seems to be realized primarily as the number of samples exceeds 10^5.

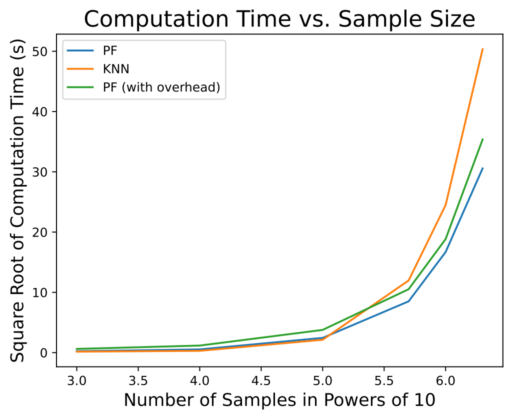

Figure 1: Square root of running times of KNN and PF approaches on the 100-dimensional three-class classification task, as well as the running times of PF including all computational overhead (beginning with the data being read into Java and ending after the final auxiliary file is created). The complexity reduction of the PF model is realized as a benefit for samples greater than 10^5.

Appendix B. Vector Fields and Flows

We now provide some background on vector fields and their associated flows (1-parameter transformations). We refer the reader to literature on the subject for additional information (Lee, 2012). Suppose that X is a smooth (tangent) vector field on $\mathbb{R}^n$:

$$X = \alpha^i \partial_{x^i} := \sum_{i=1}^{n} \alpha^i \partial_{x^i}, \tag{1}$$

where $\alpha^i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [1, n]$, and where $\{x^i\}_{i=1}^n$ are coordinates on $\mathbb{R}^n$. X assigns a tangent vector at each point and can also be viewed as a function on the set of smooth, real-valued functions. E.g. if $f : \mathbb{R}^n \to \mathbb{R}$ is smooth,

$$X(f) = \sum_{i=1}^{n} \alpha^i \frac{\partial f}{\partial x^i}. \tag{2}$$

For example, for $n = 2$, if $f(x, y) = xy$ and $X = y\partial_x$, then $X(f) = y^2$. X is also a *derivation* on the set of smooth functions on $\mathbb{R}^n$: that is, for smooth functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ and $a_1, a_2 \in \mathbb{R}$,

$$X(a_1 f_1 + a_2 f_2) = a_1 X(f_1) + a_2 X(f_2), \qquad X(f_1 f_2) = X(f_1) f_2 + f_1 X(f_2). \tag{3}$$

These properties are satisfied by derivatives. A flow on $\mathbb{R}^n$ is a smooth function $\Psi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ which satisfies

$$\Psi(0, p) = p, \qquad \Psi(s, \Psi(t, p)) = \Psi(s + t, p) \tag{4}$$

for all $s, t \in \mathbb{R}$ and for all $p \in \mathbb{R}^n$. A flow is a 1-parameter group of transformations. An example of a flow $\Psi : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$ is

$$\Psi(t, (x, y)) = (x \cos(t) - y \sin(t), x \sin(t) + y \cos(t)), \tag{5}$$

with t being the continuous parameter known as the flow parameter. This flow rotates a point (x, y) about the origin by t radians.

For a given flow Ψ, one may define a (unique) vector field X as given in Equation (2), where each function α^i is defined as

$$\alpha^i = \left(\frac{\partial \Psi}{\partial t}\right)\bigg|_{t=0}. \tag{6}$$

Such a vector field is called the infinitesimal generator of the flow Ψ. For example, the infinitesimal generator of the flow given in Equation (5) is $-y\partial_x + x\partial_y$.

Conversely, given a vector field X as in Equation (2), one may define a corresponding flow as follows. Consider the following system of differential equations:

$$\frac{dx^i}{dt} = \alpha^i, \qquad x^i(0) = x^i_0. \tag{7}$$

Suppose that a solution $\mathbf{x}(t)$ to Equation (7) exists for all $t \in \mathbb{R}$ and for all initial conditions $\mathbf{x}_0 \in \mathbb{R}^n$. Then the function $\Psi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ given by

$$\Psi(t, \mathbf{x}_0) = \mathbf{x}(t) \tag{8}$$

is a flow. The infinitesimal generator corresponding to Ψ is X. For example, to calculate the flow of $-y\partial_x + x\partial_y$, we solve

$$\dot{x} = -y, \quad \dot{y} = x, \qquad x(0) = x_0, \quad y(0) = y_0 \tag{9}$$

and obtain the flow $\Psi(t, (x_0, y_0))$ defined by Equation (5). It is generally easier to obtain the infinitesimal generator of a flow than to obtain the flow of an infinitesimal generator.

A smooth function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be X-invariant if $X(f) = 0$ identically for a smooth vector field X. The function f is Ψ-invariant if, for all $t \in \mathbb{R}$, $f = f(\Psi(t, \cdot))$ for a flow Ψ. If X is the infinitesimal generator of Ψ, f is Ψ-invariant if and only if f is X-invariant.

Appendix C. Manifold Distances Induced by Riemannian Metric Tensors

A Riemannian metric tensor g on a differentiable manifold M defines, at each point $p \in M$, an inner product on the tangent (vector) space T_pM. It can also be used to define geodesic curves (via the metric connection), which can be used to define distances between points on a manifold as follows.

Let $\gamma(t)$ be a geodesic curve derived from the metric connection of g such that $\gamma(0) = p$ and $\gamma(1) = q$, and assume that $\gamma(t)$ is unique. The distance induced by the metric g is defined as the norm (induced by g) of $\dot{\gamma}(0)$ at p:

$$d(p, q) = \sqrt{g_{ij}(p)\dot{\gamma}^i(0)\dot{\gamma}^j(0)}. \tag{10}$$

Such a distance endows M with the structure of a metric space. Henceforth, to avoid a conflict of names, we use the term "metric" to refer to a Riemannian metric tensor, and we use the term "distance measure" to refer to what is considered a metric in the sense of defining manifold distances.

The primary difficulty in Equation (10) lies in solving the geodesic equation. The geodesic equation is a second-order system of ordinary differential equations and is written using the components of a connection (such as the metric connection): therefore, exact solutions are difficult to obtain. Numeric solutions can be obtained, at least in a neighborhood of $\gamma(0)$, though the process of obtaining these numeric solutions is computationally expensive when compared to simple Euclidean distances.

We also note that, when applied to time series, the distance defined by Equation (10) is, in the language of a recent taxonomy of time series distances (Paparrizos et al., 2024), a lock-step distance. This is due primarily to the would-be model assumption that two time series are modeled as (stationary) points on the manifold M, with the number of time points corresponding to the dimension of the manifold.

Appendix D. Forest Proximities and their Applications

D.1. Geometry- and Accuracy-Preserving (GAP) Proximities

The development of a notion of tree-based similarity through random forest models was original work from Leo Breiman in the early 2000's (Breiman and Cutler, 2001). The general idea was to define similarity between two points according to colocation in terminal nodes, which define the decision space of random forests. To explicitly define similarity, the number of terminal nodes in which a pair of points, x_i and x_j, colocate was totaled, and this sum was normalized by the total number of trees trained. This notion was further extended in Rhodes et al. (2023), where it was determined that the aggregation of the total number of training points in a terminal node shared with an out-of-sample point, x_i, could define the similarity in such a way that the weighted sum of these similarities could perfectly reproduce the out-of-sample prediction of the random forest. Thus, these proximities were called random forest geometry- and accuracy-preserving proximities (RF-GAP). It has been demonstrated that the use of these RF-GAP proximities generally improved the performance of proximity-based applications such as data visualization, outlier detection, or missing value imputation, over other tree-based proximity measures (Rhodes et al., 2023; Rhodes, 2023).

The formal definition of RF-GAP proximities is as follows: given points x_i and x_j, the GAP similarity between them is

$$p(x_i, x_j) \;=\; \frac{1}{|S_i|} \sum_{t \in S_i} \frac{I(x_j \in J_i(t))\; c_j(t)}{|M_i(t)|}.$$

Here, S_i denotes the set of trees in which x_i is out-of-sample (e.g., not used to train the trees). For each tree $t \in S_i$, the multiset $J_i(t)$ contains the indices of in-bag observations whose points fall into the same terminal node as x_i. The quantity $c_j(t)$ records the number of times x_j appears in the in-bag training sample for tree t, and $M_i(t)$ denotes the multiset of all in-bag indices associated with points that share the same terminal node as x_i, including bootstrap multiplicities. The function $I(\cdot)$ is the 0-1 indicator function.

The definition was adapted for time-series data by slightly altering the formulation of proximity forests (Lucas et al., 2018) in Shaw et al. (2025). In this work, the authors introduced bootstrap resampling into the proximity forest framework. This modification was required to define out-of-sample sets, to enable the construction of GAP-style proximities. With this adjustment, the PF-GAP proximities are computed analogously to RF-GAP, using the ensemble of proximity trees for time series.

The resulting proximities retain the key theoretical properties of RF-GAP. In particular, they satisfy proximity-weighted classification, ensuring that the out-of-sample prediction of the proximity forest can be reconstructed by a weighted-majority vote where the weights are given by the PF-GAP proximities.

Depending on the application, the GAP proximities are often symmetrized. The symmetrized proximities can also be converted into pairwise dissimilarities via $d_{ij} = (1 - p(x_i, x_j))^2$. This squared transformation accentuates variation between data points, since raw GAP values are often small in magnitude. The proximities themselves define a graph in which the individual instances of data are nodes. Applications of the GAP proximities draw additional insight from the learned graph. For example, the dissimilarities d_{ij} defined above can then be employed for supervised embedding via multidimensional scaling (MDS) or PHATE (Cutler et al., 2012; Rhodes et al., 2021; Moon et al., 2019).

From a geometric perspective, each terminal node of a random forest can be interpreted as a data-adaptive partition of the underlying data structure. The co-occurrence probability $p(x_i, x_j)$ therefore quantifies how frequently two points are assigned to the same local neighborhood of the underlying data structure. In this way, RF-GAP proximity provides a probabilistic measure of adjacency that adapts to both the data distribution and the random forest structure.

D.2. Proximity-based Outliers

For forest-derived outliers, the general idea is to compare the proximities between a given point and other points belonging to the same class. If, on average, a point's same-class similarity is relatively low compared to average similarities among other same-class points, it will produce a higher outlier score. The formulation for random forest outlier scores is described below (Breiman and Cutler, 2001).

Given an observation x_i belonging to class c, we define the average proximity of x_i to the other training samples in class c as

$$\bar{P}(x_i) = \sum_{x_j \in c} \text{prox}^2(x_i, x_j).$$

The raw outlier measure for observation x_i is then defined as

$$\text{outlier\_score}(x_i) = \frac{n}{\bar{P}(x_i)}.$$

We emphasize that this measure is relative to points belonging to the same (known) class labels and does not necessarily reflect extreme variable values, as opposed to many traditional outlier scores.

We compare the use of nearest-neighbor distances versus GAP proximities derived from PF in identifying outlying points. For KNN-derived outliers, the idea is to quantify how far a point lies from its neighbors in the feature space. To formalize this, let $x_i \in \mathbb{R}^d$ be an observation, and let $N_k(x_i)$ denote the set of its k nearest neighbors under a chosen metric. We define the average neighbor distance as $\bar{d}_k(x_i)$. We consider two metrics: standard Euclidean distance and a manifold-based distance induced by the learned representation. For each case, we compute the median of $\bar{d}_k(x_i)$ across all training points, $\mathrm{med}(\bar{d}_k)$, and the median absolute deviation $\mathrm{MAD}(\bar{d}_k)$. The normalized outlier score for observation x_i is then defined as

$$\text{outlier\_score}(x_i) = \frac{|\bar{d}_k(x_i) - \mathrm{med}(\bar{d}_k)|}{\mathrm{MAD}(\bar{d}_k)}.$$

Larger values indicate greater deviation from the local neighborhood structure. We compare results obtained under the Euclidean and to assess the extent to which geometry-aware distances improve sensitivity to outliers. Unlike the forest-based score, this measure is unsupervised and does not rely on class label information.

We now illustrate both visualization and outlier scores of GAP proximities using the Palmer Penguin dataset (Horst et al., 2020), where the Euclidean distance is used for the PF model.

In Figure 2, we obtain an MDS embedding of the training points for a separate iteration which uses a train/test split of 50/50. The distances used to obtain the MDS embeddings are obtained using the PF proximities, as described in Section D.1. This embedding largely separates the classes, and we highlight the points with high outlier scores in red.

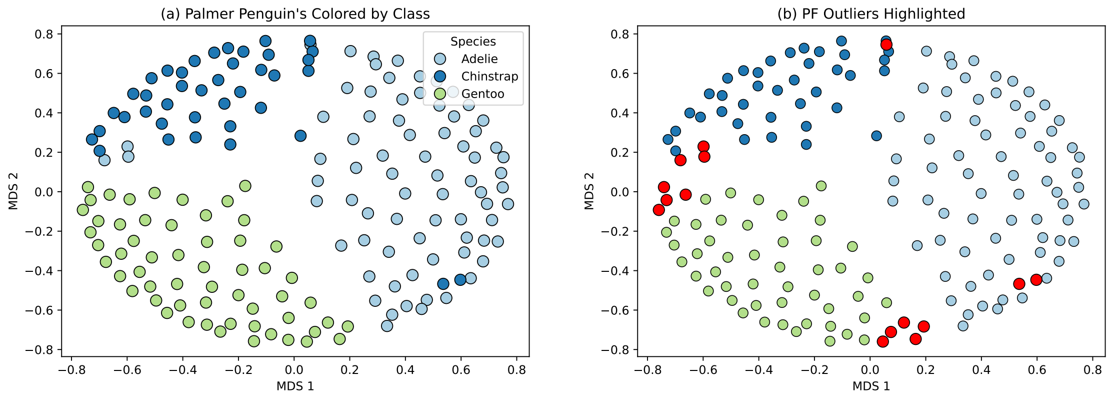

Figure 2: Left: MDS embedding of the Palmer penguin dataset using PFGAP proximities. Right: the same embedding with points having the highest outlier scores for each class highlighted in red.