MANIFOLD-AWARE FORESTS: CLOSING THE GAP TO CONVOLUTIONAL NEURAL NETWORKS

by

Ronan Perry

A thesis submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Master of Science and Engineering

Baltimore, Maryland

May, 2020

Abstract

Decision forests (DF), in particular random forests and gradient boosting trees, have demonstrated state-of-the-art accuracy compared to other methods in many supervised learning scenarios. In particular, DFs dominate other methods in tabular data, that is, when the feature space is unstructured, so that the signal is invariant to permuting feature indices. However, in structured data lying on a manifold—such as images, text, and speech—neural nets (NN), specifically convolutional neural nets (CNN), tend to outperform DFs. We conjecture that at least part of the reason for this is that the input to NN is not simply the feature magnitudes, but also their indices (for example, the convolution operation uses “feature locality”). In contrast, naïve DF implementations fail to explicitly consider feature indices. A recently proposed DF approach demonstrates that DFs, for each node, implicitly sample a random matrix from some specific distribution. Here, we build on that to show that one can choose distributions in a manifold aware fashion. For example, for image classification, rather than randomly selecting pixels, one can randomly select contiguous patches. We demonstrate the empirical performance of data living on three different manifolds: images, time-series, and a torus. In all three cases, our Manifold-aware Forest (*MF*) algorithm empirically dominates other state-of-the-art approaches that ignore feature space structure, achieving a lower classification error on all sample sizes. This dominance extends to the MNIST data set as well. Moreover, training time is significantly faster for *MF* as compared to deep nets. This approach, therefore, has promise to enable DFs and other machine learning methods to close the gap with deep nets on manifold-valued data.

Advisor: Professor Joshua Vogelstein

Secondary Readers: Professor Carey Priebe and Professor Mauro Maggioni

Acknowledgements

First and foremost, many thanks to my research advisor, Professor Joshua Vogelstein, for all his support and mentorship during my time in his lab, working on this project and others. This thesis would not have been possible without his guidance and enthusiasm. I would also like to thank the entire Neurodata lab for being an engaging and supportive environment, frisbees and all. One could not hope for a better academic environment. Among them, credit to Tyler Tomita, Jesse Patsolic, and Benjamin Falk for their initial ideas on this topic and work developing the software infrastructure that enabled this research as well as Ronak Mehta, Ali Geisa, and Jesús Arroyo for their valuable feedback.

I would like to acknowledge all the amazing faculty at Johns Hopkins I've had the fortunate of learning and working under. I cannot list them all but would like to thank to Professor Donniell Fishkind for his unwavering humor, guidance, and Matrix Analysis class, Professor Justin Bledin for his rewarding philosophy lessons, and Professor Avanti Athreya who I had the pleasure of learning under and serving as a TA for.

I also want to thank Professors Mauro Maggioni and Carey Priebe as readers on my thesis committee as well as engaging lecturers from whom I've learned much.

Lastly, special thanks to all the wonderful friends I've made at Hopkins. The people I have met during my time here have been truly inspiring and I could not have accomplished as much as I did without their support.

Ronan Perry

Table of Contents

List of Tables

List of Figures

1 Introduction

Decision forests, including random forests and gradient boosting trees, have solidified themselves in the past couple decades as a powerful ensemble learning method in supervised settings [1, 2], including both classification and regression [3]. In classification, each forest is a collection of decision trees whose individual classifications of a data point are aggregated together using majority vote. One of the strengths of this approach is that each decision tree need only perform better than chance for the forest to be a strong learner, given a few assumptions [4, 5]. Additionally, decision trees are relatively interpretable because they can provide an understanding of which features are most important for correct classification [6]. Breiman originally proposed decision trees that partition the data set using hyperplanes aligned to feature axes [6]. Yet, this limits the flexibility of the forest and requires deep trees to classify some data sets, leading to overfitting. He also suggested that algorithms which partition based on sparse linear combinations of the coordinate axes can improve performance [6]. More recently, Sparse Projection Oblique Randomer Forest (*SPORF*), partitions a random projection of the data and has shown impressive improvement over other methods [7].

Yet random forests and other machine learning algorithms frequently operate in a tabular setting, viewing an observation $\vec{x} = (x_1, \ldots, x_p)^\mathsf{T} \in \mathbb{R}^p$ as an unstructured feature vector. In doing so, they neglect the indices in settings where the indices encode additional information. For structured data, e.g. images or time series, traditional decision forests are not able to incorporate known continuity between features to learn new features. For decision forests to utilize known local structure in data, new features encoding this information must be manually constructed. Prior research has extended random forests to a variety of computer vision tasks [8, 9, 10, 11] and augmented random forests with structured pixel label information [12]. Yet these methods either generate features a priori from individual pixels, and thus do not take advantage of the local topology, or lack the flexibility to learn relevant patches. Decision forests have been used to learn distance metrics on unknown manifolds [13], but such manifold forest algorithms are unsupervised and aim to learn a low dimensional representation of the data.

Inspired by *SPORF*, we propose a projection distribution that takes into account continuity between neighboring features while incorporating enough randomness to learn relevant projections. At each node in the decision tree, sets of random spatially contiguous features are randomly selected using knowledge of the underlying manifold, a supervised manifold-aware forest for data with known structural relations. Summing the intensities of the sampled features yields a set of projections which can then be evaluated to partition the observations. We describe this proposed classification algorithm, Manifold-aware Forests (*MF*) in detail and show its effectiveness in three simulation settings as compared to common classification algorithms. Furthermore, the optimized and parallelizable open source implementation of *MF* in Python is available [1]. This addition makes for an effective and flexible learner across a wide range of manifold structures.

2 Background and Related Work

2.1 Classification

Let $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ and $D_n := \{(X_i, Y_i)\}_{i=1}^n$ be our data where $(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}$ for all i. $\mathcal{X} \subseteq \mathbb{R}^p$ (the space of feature vectors), and $\mathcal{Y} = \{1, \ldots, K\}$ (the space of class labels). A classifier is a function that assigns to $X \in \mathcal{X}$ a class label $y \in \mathcal{Y}$. Our goal is to learn a classifier $g_n(X; D_n) : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{Y}$ from our data that minimizes the expected $0 - 1$ loss, the probability of incorrect classification,

$$L(g) := P(g(X) \neq Y),$$

[1] https://neurodata.io/sporf/

with respect to the distribution of (X, Y). The optimal such classifier is the Bayes classifier,

$$g^*(X) := \underset{y \in \{1,\ldots,K\}}{\operatorname{argmax}}\, P(Y = y \mid X),$$

which has the lowest attainable error $L^* := L(g^*(X))$. For some finite number of samples n, our goals is to learn a classifier g_n from the data D_n that performs well with error denoted by $L_n = L(g_n(X)) := P(g_n(X) \neq Y)$.

2.2 Random Forests

Originally popularized by Breiman, the random forest (RF) classifier is empirically very effective [1] while maintaining strong theoretical guarantees [6]. A random forest is an ensemble of decision trees whose individual classifications of a data point are aggregated together using majority vote. Each decision tree consists of split nodes and leaf nodes. A split node is associated with a subset of the data $S = \{(X_1, Y_i)\} \subseteq D$ and splits into two child nodes, each associated with a binary partition of S. Let $e_j \in \mathbb{R}^p$ denote a unit vector in the standard basis (that is, a vector with a single one and the rest of the entries are zero) and τ a threshold value. Then S is partitioned into two subsets given the pair $\theta_j = \{e_j, \tau\}$.

$$S_\theta^L = \{(X_1, Y_i) \mid e_j^\mathsf{T} X_1 < \tau\}$$

$$S_\theta^R = \{(X_1, Y_i) \mid e_j^\mathsf{T} X_1 \geq \tau\}$$

To choose the partition, the optimal $\theta^* = (e_j^*, \tau^*)$ pair is selected via a greedy search from among a set of d randomly selected standard basis vectors e_j. The selected partition is that which maximizes some measure of information gain. A typical measure is a decrease in impurity, calculated by the Gini impurity score $I(S)$, of the resulting partitions [3]. Let $\hat{p}_k = \frac{1}{|S|} \sum_{Y_i \in S} \mathbb{I}[Y_i = k]$ be the fraction of elements of class k in partition S, then the optimal split is found as

$$I(S) = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}} |S|I(S) - |S_\theta^L|I(S_\theta^L) - |S_\theta^R|I(S_\theta^R).$$

A leaf node is created once the partition reaches a stopping criterion, typically either falling below an impurity score threshold or a minimum number of observations [3].

To classify a test sample x, it is evaluated at root node of the tree and split into one of the two partitions. This process is repeated recrusively at subsequent split nodes until x "falls into" a leaf, upon which posterior probability estimates of the class labels can be assigned. Let $l_b(x)$ be the set of training examples at the leaf node in tree b into which x falls. The empirical posterior probability of label y in b is thus $p_{nb}(y \mid x) = \frac{1}{|l_b(x)|} \sum_{i=1}^{n} \mathbb{I}[Y_i = y]\mathbb{I}[X \in l_b(x)]$. The forest composed of B trees computes the empirical posterior probability for x by averaging over the trees $p_n(y \mid x) = \frac{1}{B} \sum_{b=1}^{B} p_{nb}(y \mid x)$ and classifies x per the label with the greatest empirical posterior probability [3]

$$g_n(x) = \underset{y \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p_n(y|x)$$

For good performance of the ensemble and strong theoretical guarantees, the individual decision trees must be relatively uncorrelated from one another. Breiman's random forest algorithm does this in two ways:

1. At every node in the decision tree, the optimal split is determined over a random subset d of the total collection of features p.

2. Each tree is trained on a randomly bootstrapped sample of data points $D' \subset D$ from the full training data set.

Applying these techniques reduces the capability of random forests to overfit and lowers the upper bound of the generalization error [6].

2.3 Sparse Projection Oblique Randomer Forests

SPORF is a recent modification to random forest that has shown improvement over other versions [7, 14]. Recall that RF split nodes partition data along the coordinate axes by comparing the projection $e_j^\top x$ of observation x on standard basis e_j to a threshold value τ. *SPORF* generalizes the set of possible projections, allowing for the data to be partitioned along axes specified by any sparse vector a_j. The partition

$$S_\theta^L = \{(X_i, Y_i) \mid a_j^\top X_i < \tau\}$$

$$S_\theta^R = \{(X_i, Y_i) \mid a_j^\top X_i \geq \tau\}$$

follows from our choice of $\theta = \{a_j, \tau\}$.Rather than partitioning the data solely along the coordinate axes (i.e. the standard basis), *SPORF* creates partitions along axes specified by sparse vectors. In other words, let the dictionary $\mathcal{A}$ be the set of atoms $\{a_j\}$, each atom a p-dimensional vector defining a possible projection $a_j^\top x$. In axis-aligned forests, $\mathcal{A}$ is the set of standard basis vectors $\{e_j\}$. In *SPORF*, the dictionary $\mathcal{D}$ can be much larger, because it includes, for example, all 2-sparse vectors. At each split node, *SPORF* samples d atoms from $\mathcal{D}$ according to a specified distribution. By default, each of the d atoms are randomly generated with a sparsity level drawn from a Poisson distribution with a specified rate λ. Then, each of the non-zero elements are uniformly randomly assigned either $+1$ or -1. Note that the size of the dictionary for *SPORF* is 3^p (because each of the p elements could be -1, 0, or $+1$), although the atoms are sampled from a distribution heavily skewed towards sparsity.

3 Methods

3.1 Random Projection Forests on Manifolds

In the structured setting, the dictionary of projection vectors $\mathcal{A} = \{a_j\}$ is modified to take advantage of *a priori* knowledge of the underlying manifold on which the data lie. We term this method the Manifold-aware Forest (*MF*). This modification constrains the space of random projection forest classifiers which can be learned to better suit certain classification tasks.

The features of an observation $x \in \mathbb{R}^p$ may be viewed in some settings to lie on a manifold, which induces a notion of feature locality. Such relations between nearby features may be indicative information in certain classification tasks. For instance, edges in images may be identified by a neighborhood of pixels. The distribution of $\mathcal{A}$ is designed with respect to prior knowledge of relevant structure in the features. Each atom $a_j \in \mathcal{A}$ projects an observation to a real number, nonzero elements effectively weight and sum features. Thus, neighborhoods of features on the manifold define the atoms of $\mathcal{A}$; the distribution of those patterns yields a distribution over the atoms. At each node in the decision tree, *MF* samples d atoms, yielding d new features per observation. *MF* proceeds just like *SPORF* by optimizing the best split according to the Gini index. Algorithm pseudocode can be found in Appendix B.

In our experiments, the data was such that an observation $x \in \mathbb{R}^p$ was generalized to a vectorization of a matrix lying in $\mathbb{R}^{W \times H}$, where W and H are the width and height of that matrix, respectively. In these settings, the atoms of interest captured locally connected sets of features, specifically rectangular patches reminiscent of convolution filters. A rectangular patch is fully parameterized by the location of its upper-left corner (u, v), its height h, and width w. The location is sampled given the other two with appropriate padding so that the probability of a given feature being used is uniform. Hyperparameters determine the minimum and maximum heights $\{h_{min}, h_{max}\}$ and widths $\{w_{min}, w_{max}\}$, respectively. The atom a_j yields a projection of the data $a_j^\top x$, effectively a filter summing feature values in the sampled rectangular patch. Nonzero elements of atoms in our experiments were all set to 1 to limit combinatorial complexity but domain-specific atom design may be desired in some settings.

The structure of these atoms is flexible and task dependent. For graph-valued data, one may consider sampling a collection of neighboring edges or nodes [15]. In the case of data lying on a cyclic manifold, as in the first experiment we conduct, the atoms are patches that "wrap-around" borders of the matrix to capture added continuity. In the case of multi-channel time-series data, neighboring points in the time domain are selected but sampling of multiple channels faces no structural restrictions. We pose a single channel experiment later as well. By constructing features in this way, *MF* learns low-level features in the structured data, such as corners in images or spikes in time-series. The forest can therefore learn the features that best distinguish a class.

An important aspect of this method is that *MF* represents a shift to more "locally connected" tools whereby local points provide relevant information. However, unlike CNNs, this method is not translation equivariant. Thus, discriminative features must be constant in their indices or the training data must be rich enough to fully encapsulate possible observations [11].

3.2 Feature Importance

One of the benefits to decision trees is that their results are fairly interpretable in that they allow for estimation of the relative importance of each feature. Many approaches have been suggested [6, 16] and here a projection forest specific metric is used in which the number of times a given feature was used in projections across the ensemble of decision trees is counted. A decision tree T is composed of many nodes j, each one associated with an atom a_j^* and threshold that partition the feature space according to the projection $a_j^{*T} x$. Thus, the indices corresponding to nonzero elements of a_j^* indicate important features used in the projection. For each feature k, the number of times π_k it is used in a projection, across all split nodes and decision trees, is counted.

$$\pi_k = \sum_{T} \sum_{j \in T} \mathbb{I}(a^*_{jk} \neq 0)$$

These normalized counts represent the relative importance of each feature in making a correct classification. Such a method applies to both *SPORF* and *MF*, although different results between them would be expected due to different projection distributions yielding different hyperplanes.

4 Theoretical Results

4.1 Partitioning of the Feature Space

We note, that *MF* and all other other random projection forests have certain nice properties, such as the following result. Further details can be found in the Appendix.

Theorem 1. *A random projection tree partitions the feature space into a finite number of (possibly unbounded) convex polytopes.*

Note that by the invariance of a forest to monotone transformations of the features, our feature space $\mathcal{X}$ is equivalent to $[0, 1]^p$ for which all resulting polytopes are bounded.

4.2 Classifier Consistency

The least we can ask of a classification rule $\{g_n\}_{n=1}^{\infty}$ is for it to be consistent. That is, the probability of misclassification converges to the minimum (Bayes) error of misclassification. Random projection forests are well-behaved in that they maintain and hold certain desired statistical properties from the literature of axis-aligned splits. *Honesty* is a mild constraint that requires the set of training examples used to learn the structure of the tree to be independent of the set of examples used at the leaf nodes to estimate the posterior probabilities and has been used to prove asymptotic results about forests [5] [17] [18] [19].

Following from the results of Athey, Tibshirani, and Wager [19], we adopt this constraint and show that appropriately constructed honest random projection forests produce consistent classification rules given a few general assumptions, see Appendix for details. We denote as before $x \in \mathcal{X} \subseteq \mathbb{R}^p$ and $y \in [1, \ldots, K]$. Consider the following assumptions:

Assumption 1. *The dictionary $\mathcal{A}$ contains the set of standard basis vectors $\{e_i\}_{i=1}^{p}$, all with fixed nonzero probability of being selected at each split node.*

Assumption 2. *For all $y \in \mathcal{Y}$, $P(Y = y \mid X = x)$ is Lipschitz continuous in $x \in \mathcal{X}$.*

Assumption 3. *The samples used to populate the leaves of the trees are mutually exclusive from the set used to learn the structure of the trees (Honesty).*

Assumption 4. *There exists a density f over $\mathcal{X}$ and for all $x \in \mathcal{X}$ there exists a $\varepsilon > 0$ such that $\varepsilon < f(x) < \frac{1}{\varepsilon}$.*

Theorem 2. *Under Assumptions 1-4, the MF classification rule is consistent, i.e.*

$$L_n \xrightarrow{P} L^* \quad \text{as } n \to \infty$$

.
The Lipschitz assumption is a valid one taken in the literature on random forests [19]. It intuitively makes sense *a priori* that small deviations in x should lead to small deviations in the class probability. Note, nothing specific to the projection distribution of *MF* is used here. This implies the following result.

Corollary 1. *Under Assumptions 1-4, a random projection forest estimating posteriors as in MF yields a consistent classification rule.*

This extends Theorem 2 to hold for *SPORF* as well. Like axis-aligned splits in this setting, random projection forests partition the feature space into convex polytopes whose radii go to zero, but slow enough to populate the leaves with sets of size going to infinity.

5 Simulation Results

We evaluate *MF* in three simulation settings to show its ability to take advantage of the structure in the data. It was compared to a set of traditional classifiers and *SPORF* that all learn from the raw features. For each experiment, we used our open source implementation of *MF* and that of *SPORF* while the other classifiers were run from the Scikit-learn Python package [20] with default parameters. Additionally, we tested against a Convolutional Neural Network (CNN) built using PyTorch [21] with two convolution layers, ReLU activations, and maxpooling, followed by dropout and a densely connected hidden layer. See Appendix C for details on the hyperparameters and network architectures across experiments.

5.1 Simulation Settings

Experiment (A) is a non-Euclidean cyclic manifold example inspired by Younes [22]. Each observation is a discretization of a circle into 100 features with two non-adjacent segments of 1's in two differing patterns: class 1 features two segments of length five while class 2 features one segment of length four and one of length six. Figure 1(A) shows examples from the two classes and classification results across various sample sizes.

In experiment (B) consists of a simple 28×28 binary image classification problem. Images in class 0 contain randomly sized and spaced *horizontal* bars while those in class 1 contain randomly sized and spaced *vertical* bars. For each sampled image, $k \sim Poisson(\lambda = 10)$ bars were distributed among the rows or columns, depending on the class. The distributions of the two classes are identical if a 90 degree rotation is applied to one of the classes. Figure 1(B) shows examples from the two classes and classification results across various sample sizes.

Experiment (C) is a signal classification problem. One class consists of 100 values of Gaussian noise while the second class has an added exponentially decaying unit step beginning at time 20.

$$X_t^{(0)} = \epsilon_t$$

$$X_t^{(1)} = u(t-20)e^{(t-20)} + \epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

Figure 1(C) shows examples from the two classes and classification results across various sample sizes.

5.2 Classification Accuracy

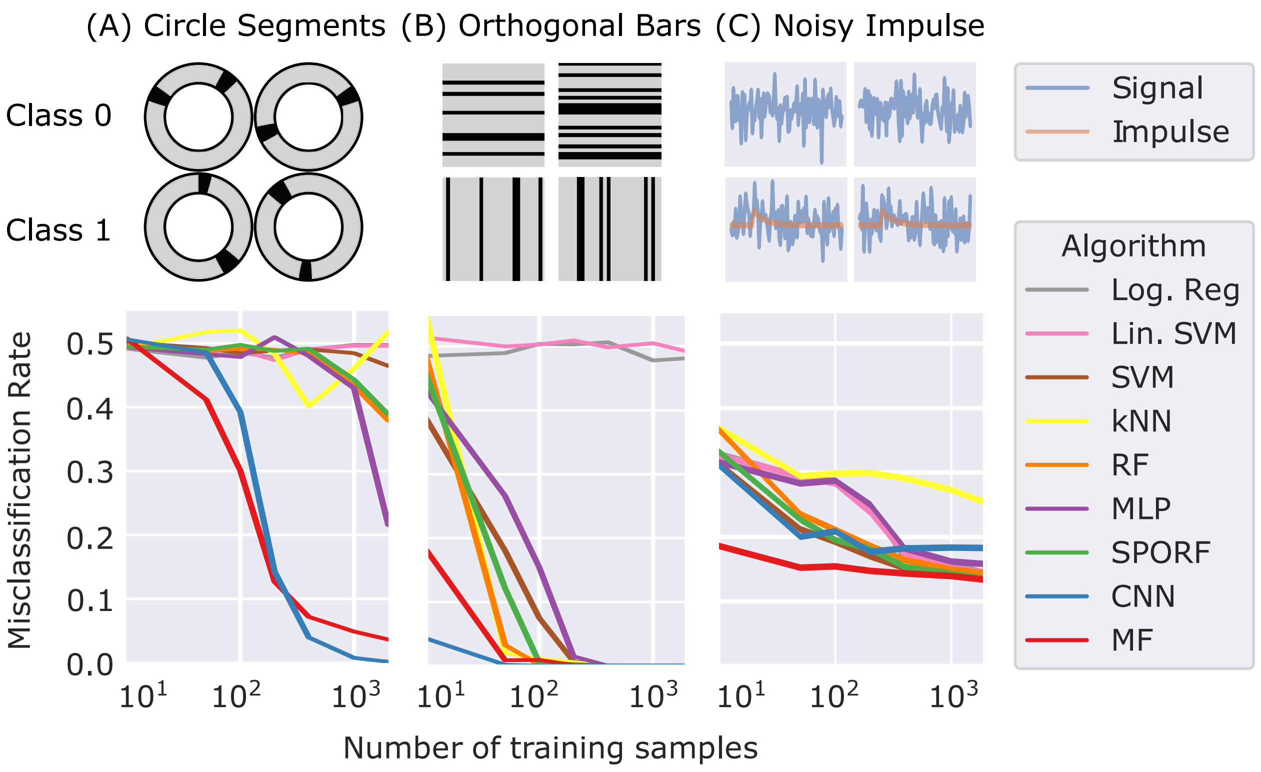

Figure 1: *MF* outperforms other algorithms in three two-class classification settings. Upper row shows examples of simulated data from each setting and class. Lower row shows misclassification rate in each setting, tested on 10,000 test samples. **(A)** Two segments in a discretized circle. Segment lengths vary by class. **(B)** Image setting with uniformly distributed horizontal or vertical bars. **(C)** White noise (class 0) vs. exponentially decaying unit impulse plus white noise (class 1).

In all three simulation settings, *MF* outperforms all other classifiers, doing especially better at low sample sizes, except the CNN for which there is no clear winner. The performance of *MF* is particularly good in the discretized circle simulation for which most other classifiers perform at chance levels. *MF* also performs quite well in the signal classification problem.

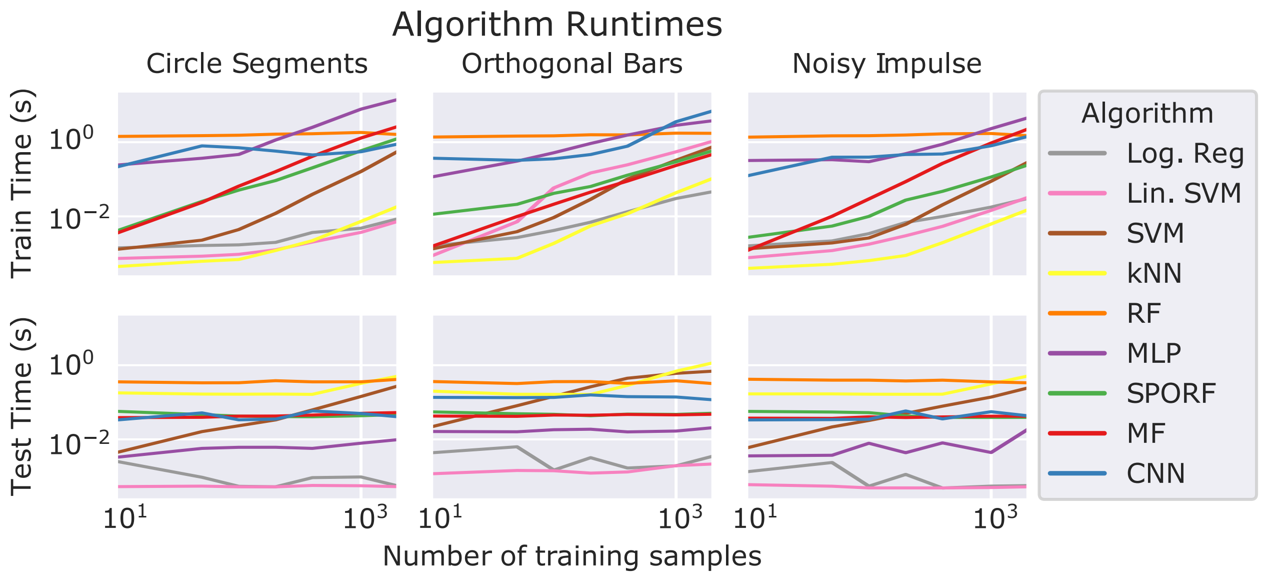

Figure 2: Algorithm train times (above) and test times (below). *MF* runtime is not particularly costly and well below CNN runtime in most examples.

5.3 Run Time

All experiments were run on CPUs in parallel and allocated the same computational resources. *MF* has train and test times on par with those of *SPORF* and *RF* and so is not particularly more computationally intensive to run. The CNN, however, took noticeably longer to run across simulations for the majority of sample sizes. Thus its strong performance in those settings comes at an added computational cost, a typical issue for deep learning methods [23].

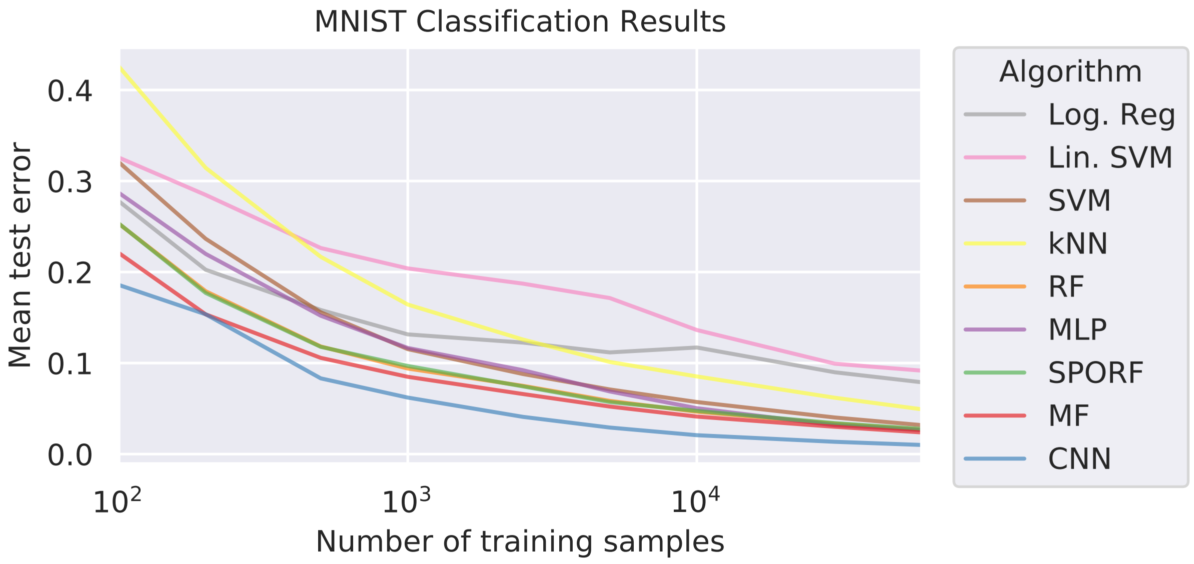

Figure 3: *MF* improves classification accuracy over all other non-CNN algorithms for all sample sizes, especially in small sample sizes.

6 Real Data Results

6.1 Classification Accuracy

MF's performance was evaluated on the MNIST dataset, a collection of handwritten digits stored in 28 by 28 square images [24], and compared to the algorithms used in the simulations. 10,000 images were held out for testing and the remaining images were used for training. The results are displayed in Figure 3. Default hyperparameters were used and *MF* was optimized to find a suitable dictionary of projections, see Appendix C for hyperparameter details. *MF* showed an improvement over the other algorithms, especially for smaller sample sizes. Thus, even this trivial modification can improve performance by several percentage points. Specifically, *MF* achieved a lower classification error than all other algorithms besides CNNs for all sample sizes on this real data problem.

6.2 Feature Importance

To evaluate the capability of *MF* to identify importance features in manifold-valued data as compared to *SPORF* and *RF*. All methods were run on a subset of the MNIST dataset: we only used threes and fives, 100 images from each class.

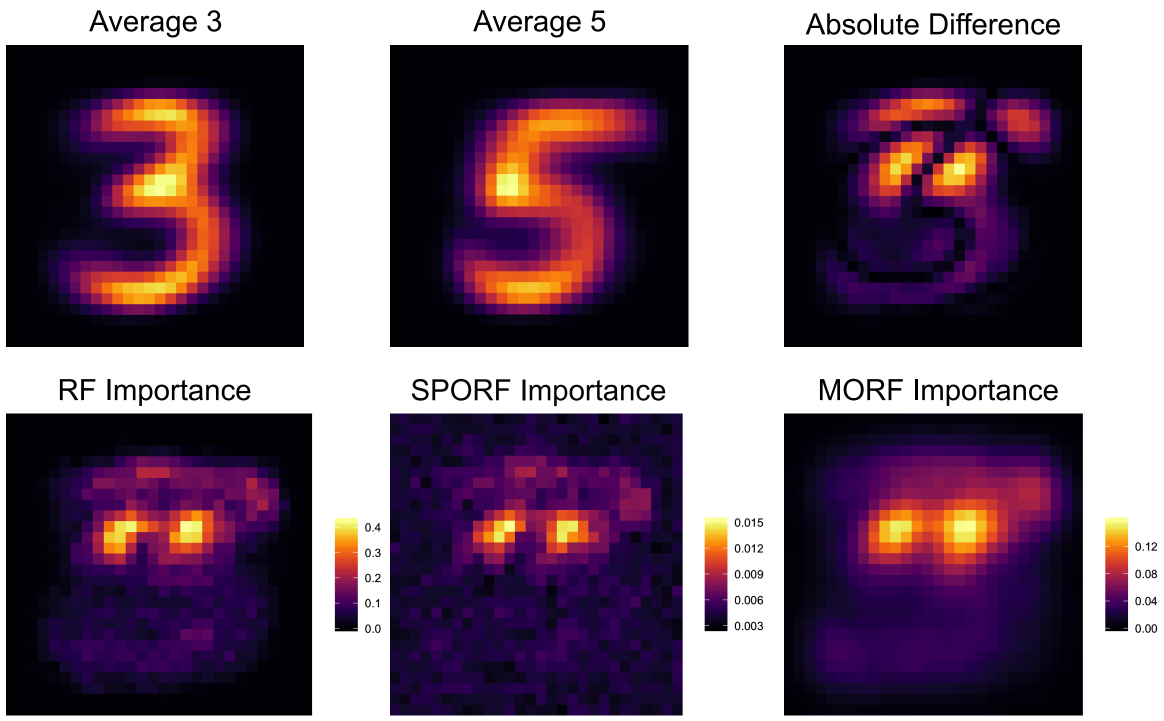

Figure 4: Averages of images in the two classes and their difference (above). Feature importance from *MF* (bottom right) shows less noise than *SPORF* (bottom middle) and is smoother than RF (bottom left).

The feature importance of each pixel is shown in Figure 4. *MF* visibly resultss in a smoother pixel importance, a result most likely from the continuity of neighboring pixels in selected projections. Although Tomita et al. [7] demonstrated empirical improvement of *SPORF* over *RF* on the MNIST data, its projection distribution yields scattered importance of unimportant background pixels as compared to *RF*. Since projections in *SPORF* have no continuity constraint, those that select high importance pixels will also select pixels of low importance by chance. This may be a nonissue asymptotically, but is a relevant problem in low sample size settings. *MF* , however, shows little or no importance of these background pixels by virtue of the modified projection distribution.

7 Discussion

The success of sparse oblique projections in decision forests has opened up many possible ways to improve axis-aligned decision forests (including random forests and gradient boosting trees) by way of specialized projection distributions. Traditional decision forests have already been applied to some manifold-valued data, using predefined features to classify images or pixels. Decision forest algorithms like that implemented in the Microsoft Kinect showed great success but ignore pixel continuity and specialize for a specific data modality, namely images [11].

We expand upon sparse oblique projections and introduced manifold-aware projection distributions that uses prior knowledge of the topology of a feature space. The open source implementation of *SPORF* has allowed for a relatively easy implementation of *MF*, creating a flexible classification method for a variety of data modalities and tailored projection dictionaries. We showed in various simulated settings that appropriate domain knowledge can improve the projection distribution to yield impressive results that challenge the strength of deep learning techniques on manifold-valued data. On the MNIST data set, *MF* closed the gap to CNNs while maintaining interpretability, robustness to hyperparameter tuning, quick run time, and theoretical justification.

Research into other, task-specific projection dictionaries may lead to improved results in real-world computer vision tasks and or classification tasks in other manifold-valued settings such as graphs. These projection distributions, while incorporated into *SPORF* here, may also be incorporated into other state of the art algorithms such as XgBoost [25].

Data and Code Availability Statement The analysis of the data was performed using an open-source software package *SPORF* (https://sporf.neurodata.io).

References

[1] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?” In: *Journal of Machine Learning Research* 15 (2014), pp. 3133–3181.

[2] Rich Caruana and Alexandru Niculescu-Mizil. “An Empirical Comparison of Supervised Learning Algorithms”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 161–168. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143865.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[4] Robert E. Schapire. “The Strength of Weak Learnability”. In: *Mach. Learn.* 5.2 (July 1990), pp. 197–227. ISSN: 0885-6125. DOI: 10.1023/A:1022648800760.

[5] Gérard Biau, Luc Devroye, and Gábor Lugosi. “Consistency of Random Forests and Other Averaging Classifiers”. In: *J. Mach. Learn. Res.* 9 (June 2008), pp. 2015–2033. ISSN: 1532-4435.

[6] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[7] Tyler M. Tomita, James Browne, Cencheng Shen, Jesse L. Patsolic, Jason Yim, Carey E. Priebe, Randal Burns, Mauro Maggioni, and Joshua T. Vogelstein. “Random Projection Forests”. In: *arXiv e-prints*, arXiv:1506.03410 (June 2015), arXiv:1506.03410. arXiv: 1506.03410 [stat.ML].

[8] V. Lepetit, P. Lagger, and P. Fua. “Randomized trees for real-time keypoint recognition”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. June 2005, 775–781 vol. 2. DOI: 10.1109/CVPR.2005.288.

[9] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. “Hough Forests for Object Detection, Tracking, and Action Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.11 (Nov. 2011), pp. 2188–2202. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.70.

[10] A. Bosch, A. Zisserman, and X. Munoz. “Image Classification using Random Forests and Ferns”. In: *2007 IEEE 11th International Conference on Computer Vision*. Oct. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409066.

[11] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. “Real-time human pose recognition in parts from single depth images”. In: *CVPR 2011*. June 2011, pp. 1297–1304. DOI: 10.1109/CVPR.2011.5995316.

[12] P. Kontschieder, S. R. Bulò, H. Bischof, and M. Pelillo. “Structured class-labels in random forests for semantic image labelling”. In: *2011 International Conference on Computer Vision*. Nov. 2011, pp. 2190–2197. DOI: 10.1109/ICCV.2011.6126496.

[13] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. “Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning”. In: *Found. Trends. Comput. Graph. Vis.* 7.2&#8211;3 (Feb. 2012), pp. 81–227. ISSN: 1572-2740. DOI: 10.1561/0600000035.

[14] Tyler M. Tomita, Mauro Maggioni, and Joshua T. Vogelstein. “ROFLMAO: Robust Oblique Forests with Linear MAtrix Operations”. In: *Proceedings of the 2017 SIAM International Conference on Data Mining*. 2017, pp. 498–506. DOI: 10.1137/1.9781611974973.56.

[15] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–21. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978386.

[16] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4765–4774.

[17] Misha Denil, David Matheson, and Nando De Freitas. “Narrowing the Gap: Random Forests In Theory and In Practice”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research. Bejing, China: PMLR, 22–24 Jun 2014, pp. 665–673.

[18] Stefan Wager and Guenther Walther. “Adaptive Concentration of Regression Trees, with Application to Random Forests”. In: *arXiv:1503.06388 [math, stat]* (Apr. 2016). arXiv: 1503.06388.

[19] Susan Athey, Julie Tibshirani, and Stefan Wager. “Generalized Random Forests”. In: *arXiv:1610.01271 [econ, stat]* (Apr. 2018). arXiv: 1610.01271.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[22] Laurent Younes. *Diffeomorphic Learning*. 2018. arXiv: 1806.01240 [stat.ML].

[23] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. “On the Computational Efficiency of Training Neural Networks”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’14. Montreal, Canada: MIT Press, 2014, pp. 855–863.

[24] Yann Lecun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. 1999.

[25] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785.

[26] Richard Guo, Ronak Mehta, Jesus Arroyo, Hayden Helm, Cencheng Shen, and Joshua T. Vogelstein. “Estimating Information-Theoretic Quantities with Uncertainty Forests”. In: *arXiv:1907.00325 [cs, stat]* (Nov. 2019). arXiv: 1907.00325.

[27] Ryan Rifkin and Aldebaro Klautau. “In Defense of One-Vs-All Classification”. In: *J. Mach. Learn. Res.* 5 (Dec. 2004), pp. 101–141. ISSN: 1532-4435.

Appendices

A Proofs

A.1 Convex Polytope Partition Results

Theorem 1. *A random projection tree partitions the feature space into a finite number of (possibly unbounded) convex polytopes.*

Proof. A convex polytope in d dimensions can be defined as the union of a finite number of halfspaces, where a halfspace is a $d-1$ dimensional surface defined by the linear inequality

$$a^T x \leq b$$

for fixed $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$. In a random projection tree, each split node i partitions the set of points at that node according to such an inequality $a_i^T x \leq b_i$. Consider the path of k split nodes, including the root, to a leaf l and the set of corresponding halfspace defining $\{(a_i, b_i\}_{i=1}^k$ terms for each split node. We see that in the feature space S, the subset that "falls into" leaf l is the solution set to

$$A_l x \leq b_l$$

where $A_l = [a_1, \ldots, a_k]^T$ and $b_l = [b_1, \ldots, b_k]^T$.

Thus each leaf node forms a convex polytope. Additionally, note that any $x \in S$ will deterministically end up in a leaf node (by classification of x) as the tree is of finite depth and that all leaf node convex polytopes are mutually exclusive as the lowest common ancestor of any two leaves forms mutually exclusive sets. If the feature space is unbounded, then at least one partition must be unbounded too. Thus, a tree partitions the feature space into a finite number of possibly infinite convex polytopes. $\square$

A.2 Consistency Results

The least we can ask of our classification rule $\{g_n\}_{n=1}^{\infty}$ is for it to be consistent,

$$L_n \xrightarrow{P} L^* \quad \text{as } n \to \infty$$

where L_n and L^* are the expected 0-1 losses of g_n and the Bayes decision g^*, respectively.

A.2.1 Posterior Consistency

Following from the results of Athey, Tibshirani, and Wager [19] and in the spirit of [26], we show that random projection forests such as *MF* can produce a consistent empirical estimate $p_n(y \mid x)$ of the posterior $P(Y = y \mid X = x)$ which we denote as $p(y \mid x)$.

Lemma 2. *Under Assumptions 1-4 of Section 4.2, for all $y \in \{1, \dots, K\}$ and $x \in \mathcal{X}$, the honest MF estimate $p_n(y \mid x) \xrightarrow{P} p(y \mid x)$ as $n \to \infty$*

Proof. To estimate the posterior probability in a multiclass setting, we adopt the one-vs-all approach [27]. Let y be the fixed arbitrary class label of interest. Given our data, we seek to estimate the posterior probability $p(y \mid x)$, equivalent to estimating the conditional mean $\mu(x) := \mathbb{E}[\mathbb{I}[Y = y] \mid X = x]$ for a fixed y. To follow the notation of Athey, Tibshirani, and Wager [19], we frame $\mu(x)$ as the solution to the estimation equation

$$M_\mu(x) := \mathbb{E}[\psi_{\mu(x)}(Y) \mid X = x] = 0$$

where the score function $\psi_{\mu(x)}(Y)$ is defined as

$$\psi_{\mu(x)}(Y) := \mathbb{I}[Y = y] - \mu(x).$$

The solution can be estimated as the solution, $\hat{\mu}(x)$, to the empirical estimation equation

$$\sum_{i=1}^{n} \alpha_i(x)\psi_{\hat{\mu}(x)}(Y_i) = 0.$$

It follows that

$$\hat{\mu}(x) = \sum_{i=1}^{n} \alpha_i(x)\mathbb{I}[Y_i = y]$$

per the expansion

$$\sum_{i=1}^{n} \alpha_i(x)\psi_{\hat{\mu}(x)}(Y_i) = \sum_{i=1}^{n} \alpha_i(x)(\mathbb{I}[Y_i = y] - \hat{\mu}(x)) = \sum_{i=1}^{n} \alpha_i(x)\mathbb{I}[Y_i = y] - \hat{\mu}(x) = 0.$$

These weights we derive from a learned random forest. Let a forest be composed of B trees. In a single tree b, let $l_b(x)$ denote the set of training examples at the leaf node for which X is placed. Define the weights $\alpha_{ib}(x)$ for that tree as

$$\alpha_{ib}(x) := \frac{1}{|l_b(x)|}\mathbb{I}[x_i \in l_b(x)],$$

the normalized indicator of whether or not x and x_i exist in the same leaf. Thus the forest weights $\alpha_i(x) = \frac{1}{B}\sum_{b=1}^{B} \alpha_{ib}(x)$ are simply the normalized weights across all trees.

By Theorem 3 of Generalized Random Forests [19], a random forest built according to the following Specification 1A and solving an estimation problem satisfying the following assumptions yields a consistent estimator $p_n(y \mid x)$ for $p(y \mid x)$. We list these requirements and verify that they hold for honest random projection forests.

Importantly, the use of splits which may be a linear combination of features does not violate these properties given Assumption 1 from Section 4.2 guarantees that the individual features are split on with positive probability and so maintains the Bayes error. The same heterogeneity maximization step at each split node in [19] occurs, albeit with more ways to form the partition into two child nodes.

Specification 1A: Each tree in the forest is built with the following requirements:

i. Is symmetric (their outputs are invariant to the permuting of the indices of the training data).

ii. Has balanced splits (each child node receives a nonzero fraction of the observations at the parent node).

iii. Is randomized (each feature has a nonzero probability of being split on).

iv. The forest is honest (the posterior probabilities for a test point at a tree come from training examples not used in construction of that tree).

v. $s \to \infty$ and $s/n \to 0$ where s is the number of subsampled training examples used to construct each tree and n is the total number of examples.

Specification 1A is clearly met. Subsampling of the training examples is invariant on the example indices (i) and we can asymptotically take subsamples of size n^α for $0 < \alpha < 1$ to satisfy (v). An unbalanced split is no better than any other split per the Gini split criterion and so each child node will contain a nonzero fraction of the samples at the parent (ii). Lastly, Assumptions 1 and 3 from Section 4.2 give us (iii) and (iv) respectively.

Assumption 1A: For fixed values $\mu(x)$, we assume that $M_\mu(x)$ is Lipschitz continuous in x.

This holds per Assumption 2 from Section 4.2.

Assumption 2A: When x is fixed, we assume that $M_\mu(x)$ is twice continuously differentiable in μ with a uniformly bounded second derivative, and that $\frac{\partial}{\partial(\mu)} M_\mu(x) \mid_{\mu(x)} \neq 0$ for all $x \in \mathcal{X}$.

This is true, as evident in the derivatives

$$\frac{\partial}{\partial \mu} M_\mu(x) = -1 \quad \text{and} \quad \frac{\partial^2}{\partial^2 \mu} M_\mu(x) = 0.$$

Assumption 3A: The worst-case variogram of $\psi_\mu(Y)$ is Lipschitz-continuous in $\mu(x)$.

This is evident in the worse-case variogram for two solutions μ and μ'

$$\begin{aligned}\gamma(\mu(x), \mu'(x)) &:= \sup_{x\in\mathcal{X}}\{Var(\psi_{\mu(x)}(Y) - \psi_{\mu'(x)}(Y_i) \mid X = x)\} \\ &= \sup_{x\in\mathcal{X}}\{Var(\mathbb{I}[Y = y] - \mu(x) - \mathbb{I}[Y = y] - \mu'(x) \mid X = x)\} \\ &= \sup_{x\in\mathcal{X}}\{Var(\mu'(x) - \mu(x)|X = x)\} = 0\end{aligned}$$

which is trivially Lipschitz-continuous.

Assumption 4A: The ψ-functions can be written as $\psi_{\mu(x)}(Y) = \lambda(\mu(x); Y) + \xi_{\mu(x)}(g(Y))$, such that λ is Lipschitz-continuous in μ, $g : \{Y\} \to R$ is a univariate summary of Y, and $\xi_{\mu(x)} : R \to R$ is any family of monotone and bounded functions.

Clearly $\psi_{\mu(x)}(Y)$ is linear in $\mu(x)$ and so is a Lipschitz-continuous function in $\mu(x)$. The other term is 0 in this case.

Assumption 5A: For any weights $\alpha_i(x)$ such that $\sum_i \alpha_i(x) = 1$, the estimation equation returned a minimizer $\mu\hat{(x)}$ that at least approximately solves the estimating equation

$$||\sum_{i=1}^{n} \alpha_i(x)\psi_{\mu\hat{(x)}}(Y_i)||_2 \leq Cmax\{\alpha_i(x)\}$$

for some constant $C \geq 0$.

As shown previously shown, the estimation equation is solved to equal 0.

Assumption 6A: The score function $\psi_{\mu(x)}(Y)$ is a negative sub gradient of a convex function, and the expected score $M_\mu(x)$ is the negative gradient of a strongly convex function.

This holds true by construction of the following convex function

$$\Psi_{\mu(x)}(Y) := \frac{1}{2}(\mathbb{I}[Y = y \mid X = x] - \mu(x))^2 \quad \text{such that} \quad \psi_\mu(Y) = -\frac{d}{d\mu}\Psi_{\mu(x)}(Y)$$

and the following strongly convex function

$$\mathbb{M}_\mu(x) := \frac{1}{2}(P(Y = y \mid x) - \mu(x))^2 \quad \text{such that} \quad M_\mu(x) = -\frac{d}{d\mu}\mathbb{M}_\mu(x)$$

□

A.2.2 Classifier Consistency

The prior proof established consistency for each posterior probability estimate $p_n(y \mid x)$. We now proceed to show consistency for the classification rule $g_n(x) = \operatorname{argmax}_y p_n(y \mid x)$.

Lemma 3. *Let $x \in \mathcal{X}$ with unique maximum $y^* := \operatorname{argmax}_y p(y \mid x)$, and define the finite sample estimate $\hat{y} := \operatorname{argmax}_y p_n(y \mid x)$. Given that $p_n(y \mid x)$ is a consistent estimator for $p(y \mid x)$, then*

$$P[\hat{y} \neq y^* \mid x] \to 0 \quad \textit{as} \quad n \to \infty$$

Proof. We omit the conditional for notational brevity by substituting $p(y) := p(y \mid x)$ and $p_n(y) := p_n(y \mid x)$,

$$\begin{aligned}
P[\hat{y} \neq y^* \mid x] &= P[\max_y p_n(y) > p_n(y^*)] \\
&= P\left[\bigcup_{y \neq y^*} p_n(y) > p_n(y^*)\right] \\
&\leq \sum_{y \neq y^*} P[p_n(y) > p_n(y^*)] \\
&= \sum_{y \neq y^*} P[p_n(y) - p_n(y^*) > 0] \\
&= \sum_{y \neq y^*} P[(p_n(y) - p_n(y^*)) - (p(y) - p(y^*)) > p(y^*) - p(y)]
\end{aligned}$$

Let $\varepsilon_y := p(y^*) - p(y)$ and note that $\varepsilon_y > 0$ for all $y \in \mathcal{Y} \setminus \{y^*\}$ since y^* is a unique maximum.

Observe that

$$\sum_{y \neq y^*} P[(p_n(y) - p_n(y^*)) - (p(y) - p(y^*)) > \varepsilon_y]$$

$$\leq \sum_{y \neq y^*} P\big[\big|(p_n(y) - p_n(y^*)) - (p(y) - p(y^*))\big| > \varepsilon_y\big]$$

By the consistency of the individual posteriors, the difference of two is consistent and so since $\mathcal{Y}$ is a finite set,

$$P[\hat{y} \neq y^* \mid x] \leq \sum_{y \neq y^*} P[|(p_n(y) - p_n(y^*)) - (p(y) - p(y^*))| > \varepsilon_y] \to 0 \quad \text{as} \quad n \to \infty$$

$\square$

Theorem 2. *Under assumptions 1-4 of Section 4.2, the MF classification rule is consistent, i.e.*

$$L_n \xrightarrow{P} L^* \quad \text{as } n \to \infty$$

.

Proof. Denote the finite samples classification rule $\hat{y} := \operatorname{argmax}_y p_n(y \mid x)$ as before and let $y^* := \operatorname{argmax}_y p(y \mid x)$ be a unique maximum. If y^* were not unique, we would instead consider the aggregate of all such maximum classes as a pseudo class, apply the following analyses, and be confident in both L_n and L^* up to a factor equal to the reciprocal of the number aggregated classes.

To begin, for any $\varepsilon > 0$,

$$P\left[|L_n - L^*| > \varepsilon\right] = P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon\right]$$

$$= P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} = y^*\right] \times P\left[\hat{y} = y^*\right] +$$

$$P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} \neq y^*\right] \times P\left[\hat{y} \neq y^*\right].$$

In the case that $\hat{y} = y^*$, by Lemma 2 we have convergence of the posteriors and so

$$P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} = y^*\right] \to 0 \quad \text{as } n \to \infty.$$

In the case that $\hat{y} \neq y^*$, by Lemma 3 we have that

$$P\left[\hat{y} \neq y^*\right] \to 0 \quad \text{as } n \to \infty.$$

Since the probabilities are bounded above by one, it follows that both

$$P\left[|p_n(\hat{y} \mid X) - p(y^* \mid x)| > \varepsilon \mid \hat{y} = y^*\right] \times P\left[\hat{y} = y^*\right] \to 0 \quad \text{as } n \to \infty$$

and

$$P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} \neq y^*\right] \times P\left[\hat{y} \neq y^*\right] \to 0 \quad \text{as } n \to \infty.$$

Thus,

$$P\left[|L_n - L^*| > \varepsilon\right] = P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon\right] \to 0 \quad \text{as } n \to \infty.$$

□

B Pseudocode

Algorithm 1 Learning a Manifold-aware Forest decision tree.

Input: (1) $\mathcal{D}_n$: training data (2) d: dimensionality of the projected space, (3) $f_{\mathbf{A}}$: distribution of the atoms, (4) Θ: set of split eligibility criteria

Output: A *MF* decision tree T

```
 1: function $T$ = GROWTREE($\mathbf{X}, \mathbf{y}, f_{\mathbf{A}}, \Theta$)
 2:     $c = 1$                                   ▷ $c$ is the current node index
 3:     $M = 1$                                   ▷ $M$ is the number of nodes currently existing
 4:     $S^{(c)}$ = bootstrap($\{1, ..., n\}$)     ▷ $S^{(c)}$ is the indices of the observations at node $c$
 5:     while $c < M + 1$ do                      ▷ visit each of the existing nodes
 6:         $(\mathbf{X}', \mathbf{y}') = (\mathbf{x}_i, y_i)_{i \in S^{(c)}}$     ▷ data at the current node
 7:         for $k = 1, \ldots, K$ do $n_k^{(c)} = \sum_{i \in S^{(c)}} I[y_i = k]$ end for ▷ class counts (for classification)
 8:         if $\Theta$ satisfied then            ▷ do we split this node?
 9:             $\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_d] \sim f_{\mathbf{A}}$     ▷ sample random $p \times d$ matrix of atoms
10:             $\widetilde{\mathbf{X}} = \mathbf{A}^T \mathbf{X}' = (\widetilde{\mathbf{x}}_i)_{i \in S^{(c)}}$     ▷ random projection into new feature space
11:             $(j^*, t^*)$ = findbestsplit($\widetilde{\mathbf{X}}, \mathbf{y}'$)     ▷ Algorithm 2
12:             $S^{(M+1)} = \{i : \mathbf{a}_{j^*} \cdot \widetilde{\mathbf{x}}_i \leq t^* \quad \forall i \in S^{(c)}\}$     ▷ assign to left child node
13:             $S^{(M+2)} = \{i : \mathbf{a}_{j^*} \cdot \widetilde{\mathbf{x}}_i > t^* \quad \forall i \in S^{(c)}\}$     ▷ assign to right child node
14:             $\mathbf{a}^{*(c)} = \mathbf{a}_{j^*}$     ▷ store best projection for current node
15:             $\tau^{*(c)} = t^*$                  ▷ store best split threshold for current node
16:             $\kappa^{(c)} = \{M + 1, M + 2\}$     ▷ node indices of children of current node
17:             $M = M + 2$                          ▷ update the number of nodes that exist
18:         else
19:             $(\mathbf{a}^{*(c)}, \tau^{*(c)}, \kappa^{*(c)})$ = NULL
20:         end if
21:         $c = c + 1$                              ▷ move to next node
22:     end while
23:     return $(S^{(1)}, \{\mathbf{a}^{*(c)}, \tau^{*(c)}, \kappa^{(c)}, \{n_k^{(c)}\}_{k \in \mathcal{Y}}\}_{c=1}^{m-1})$
24: end function
```

Algorithm 2 Finding the best node split. This function is called by growtree (Alg 1) at every split node. For each of the p dimensions in $\mathbf{X} \in \mathbb{R}^{p \times n}$, a binary split is assessed at each location between adjacent observations. The dimension j^* and split value τ^* in j^* that best split the data are selected. The notion of "best" means maximizing some choice in scoring function. In classification, the scoring function is typically the reduction in Gini impurity or entropy. The increment function called within this function updates the counts in the left and right partitions as the split is incrementally moved to the right.

```
Input: (1) $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{p \times n} \times \mathcal{Y}^n$, where $\mathcal{Y} = \{1, \dots, K\}$
Output: (1) dimension $j^*$, (2) split value $\tau^*$
1: function $(j^*, \tau^*) =$ FINDBESTSPLIT($\mathbf{X}, \mathbf{y}$)
2:     for $j = 1, \dots, p$ do
3:         Let $\mathbf{x}^{(j)} = (x_1^{(j)}, \dots, x_n^{(j)})$ be the $jth$ row of $\mathbf{X}$.
4:         $\{m_i^j\}_{i \in [n]} =$ sort($\mathbf{x}^{(j)}$)        ▷ $m_i^j$ is the index of the $i^{th}$ smallest value in $\mathbf{x}^{(j)}$
5:         $t = 0$        ▷ initialize split to the left of all observations
6:         $n' = 0$        ▷ number of observations left of the current split
7:         $n'' = n$        ▷ number of observations right of the current split
8:         if (task is classification) then
9:             for $k = 1, \dots, K$ do
10:                $n_k = \sum_{i=1}^n I[y_i = k]$        ▷ total number of observations in class $k$
11:                $n'_k = 0$        ▷ number of observations in class $k$ left of the current split
12:                $n''_k = n_k$        ▷ number of observations in class $k$ right of the current split
13:            end for
14:        end if
15:        for $t = 1, \dots, n-1$ do        ▷ assess split location, moving right one at a time
16:            $(\{(n'_k, n''_k)\}, n', n'', y_{m_t^j}) =$ increment($\{(n'_k, n''_k)\}, n', n'', y_{m_t^j}$)
17:            $Q^{(j,t)} =$ score($\{(n'_k, n''_k)\}, n', n''$)        ▷ measure of split quality
18:        end for
19:    end for
20:    $(j^*, t^*) = \operatorname{argmax}_{j,t} Q^{(j,t)}$
21:    for $i = 0, 1$ do $c_i = m_{t^*+i}^{j^*}$ end for
22:    $\tau^* = \frac{1}{2}(x_{c_0}^{(j^*)} + x_{c_1}^{(j^*)})$        ▷ compute the actual split location from the index $j^*$
23:    return $(j^*, \tau^*)$
24: end function
```

C Hyperparameters

Table 1: CNN hyperparameters for each experiment.

Experiment	Classifier	Architecture Sequence
Circle	CNN	Conv1d(32, window=6, stride=1)
		MaxPool1d(window=2, stride=2)
		Conv1d(64, window=10, stride=1)
		MaxPool1d(window=2, stride=2)
		Dropout(p=0.5)
		Linear(500)
		Linear(2)
H/V Bars	CNN	Conv2d(32, window=5, stride=1)
		MaxPool1d(window=2, stride=2)
		Conv2d(64, window=5, stride=1)
		MaxPool1d(window=2, stride=2)
		Dropout(p=0.5)
		Linear(200)
		Linear(2)
Impulse	CNN	Conv1d(32, window=10, stride=1)
		MaxPool2d(window=2, stride=2)
		Conv2d(64, window=5, stride=1)
		MaxPool2d(window=2, stride=2)
		Dropout(p=0.5)
		Linear(200)
		Linear(2)
MNIST	CNN	Conv2d(32, window=5, stride=1)
		MaxPool2d(window=2, stride=2)
		Conv2d(64, window=5, stride=1)
		MaxPool2d(window=2, stride=2)
		Dropout(p=0.5)
		Linear(200)
		Linear(10)

Table 2: *scikit-learn*, *SPORF*, and *MF* hyperparameters for each experiment.

Experiment	Classifier	Hyperparameters
Circle	Lin. SVM	C=1, penalty="l2";kernel="linear";loss="squared_hinge"
Circle	Log. Reg	C=1, penalty="l2"
Circle	MF	max_features=0.5; patch_height_max=1; patch_height_min=1; patch_width_max=12; patch_width_min=3
Circle	MLP	activation="relu"; alpha=0.0001; hidden_layer_sizes=(100,); solver="adam"
Circle	RF	max_features=sqrt(n)
Circle	SPORF	max_features=sqrt(n)
Circle	SVM	C=1; gamma=1/(n_features*Var(X)); kernel="rbf"
Circle	kNN	n_neighbors=5; p=2
H/V Bars	Lin. SVM	C=1, penalty="l2";kernel="linear";loss="squared_hinge"
H/V Bars	Log. Reg	C=1, penalty="l2"
H/V Bars	MF	max_features=auto; patch_height_max=2; patch_height_min=2; patch_width_max=9; patch_width_min=2
H/V Bars	MLP	activation="relu"; alpha=0.0001; hidden_layer_sizes=(100,); solver="adam"
H/V Bars	RF	max_features=sqrt(n)
H/V Bars	SPORF	max_features=sqrt(n)
H/V Bars	SVM	C=1; gamma=1/(n_features*Var(X)); kernel="rbf"
H/V Bars	kNN	n_neighbors=5; p=2
Impulse	Lin. SVM	C=1, penalty="l2";kernel="linear";loss="squared_hinge"
Impulse	Log. Reg	C=1, penalty="l2"
Impulse	MF	max_features=0.3; patch_height_max=1; patch_height_min=1; patch_width_max=12; patch_width_min=2
Impulse	MLP	activation="relu"; alpha=0.0001; hidden_layer_sizes=(100,); solver="adam"
Impulse	RF	max_features=sqrt(n)
Impulse	SPORF	max_features=sqrt(n)
Impulse	SVM	C=1; gamma=1/(n_features*Var(X)); kernel="rbf"
Impulse	kNN	n_neighbors=5; p=2
MNIST	Lin. SVM	C=1, penalty="l2";kernel="linear";loss="squared_hinge"
MNIST	Log. Reg	C=1, penalty="l2"
MNIST	MF	max_features=auto; patch_height_max=2; patch_height_min=2; patch_width_max=5; patch_width_min=2;
MNIST	MLP	activation="relu"; alpha=0.0001; hidden_layer_sizes=(100,); solver="adam"
MNIST	RF	max_features=sqrt(n)
MNIST	SPORF	max_features=sqrt(n)
MNIST	SVM	C=1; gamma=1/(n_features*Var(X)); kernel="rbf"
MNIST	kNN	n_neighbors=5; p=2

Biographical Note

Ronan Perry grew up in Ithaca, NY, and attended the Johns Hopkins University where he earned a Bachelor’s degree in Applied Mathematics & Statistics in 2019 and a Master’s degree in Biomedical Engineering in 2020, with a concentration in biomedical data science. During this period, he worked at the biotechnology company Rheonix, spent a summer as a researcher in the Medical Image Processing Lab in Geneva, Switzerland, and studied abroad at the Danish Technical Institute where he competed in national and international ultimate frisbee tournaments.