Manifold Oblique Random Forests: Towards Closing the Gap on Convolutional Deep Networks

Ronan Perry[+,1], Adam Li[+,1,3], Chester Huynh[+,1,3], Tyler M. Tomita[1], Ronak Mehta[2], Jesus Arroyo[2], Jesse Patsolic[2], Benjamin Falk[2], Sridevi V. Sarma[1,3], Joshua T. Vogelstein[1,2,3*]

Abstract. Decision forests (Forests), 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, Forests dominate other methods in tabular data, that is, when the feature space is unstructured, so that the signal is invariant to a permutation of the feature indices. However, in structured data lying on a manifold—such as images, text, and speech—deep networks (Networks), specifically convolutional deep networks (ConvNets), tend to outperform Forests. We conjecture that at least part of the reason for this is that the input to Networks is not simply the feature magnitudes, but also their indices (for example, the convolution operation uses "feature locality"). In contrast, naïve Forest implementations fail to explicitly consider feature indices. A recently proposed Forest approach demonstrates that Forests, for each node, implicitly sample a random matrix from some specific distribution. These Forests, like some classes of Networks, learn by partitioning the feature space into convex polytopes corresponding to linear functions. We build on that approach and show that one can choose distributions in a *manifold-aware fashion* to incorporate feature locality. We demonstrate the empirical performance on data whose features live on three different manifolds: a torus, images, and time-series. Moreover, we demonstrate its strength in multivariate simulated settings and also show superiority in predicting surgical outcome in epilepsy patients and predicting movement direction from raw stereotactic EEG data from non-motor brain regions. In all simulations and real data, Manifold Oblique Random Forest (MORF) algorithm outperforms approaches that ignore feature space structure and challenges the performance of ConvNets. Moreover, MORF runs fast and maintains interpretability and theoretical justification. Our code is open source and available as part of the package at https:///neurodata.io/code/.

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]. In 2001, 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 trees of large depth 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], which was corroborated in subsequent work [7]. More recently, Sparse Projection Oblique Randomer Forest (SPORF)—which leverages sparse random projection of the data—has shown impressive improvement over other methods [8]. Other extensions have led to neural decision forests [9, 10] which attempt to combine the strengths of neural networks and random forests by using differentiable functions at split nodes and leaves, leading to trees which can be learned via backpropagation. Under certain function choices, once learned, these forests turn out to be equivalent to neural networks with many zeroed weights [10].

Random forests and other machine learning algorithms typically 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 feature indices in settings where the indices encode additional information. For structured data, e.g. images or time series, traditional decision forests do not incorporate the known local structure. For decision forests to utilize known local structure in data, new features encoding this information must be manually constructed or new splitting criterion must be implemented. Prior research has extended random forests to a variety of computer vision tasks [11–14] and augmented random forests with struc-

* [1] Department of Biomedical Engineering Johns Hopkins University, [2] Center for Imaging Science, [3] Institute for Computational Medicine, Kavli Neuroscience Discovery Institute, Johns Hopkins University, [+] Indicates co-first authorship

tured pixel label information [15]. The decision tree at the heart of the Microsoft Kinect showed great success by specializing for image data with depth information [14]. Yet these methods either generate features *a priori* from individual pixels (and thus do not take full advantage of the local topology) or lack the flexibility to learn relevant structure. Other approaches have circumvented the problem of learning from raw structured data through tabular feature engineering, notably employed by the aforementioned deep neural decision forest [9] using a convolutional deep network. Decision forests have also been used to learn distance metrics on unknown manifold structures [16], but such manifold forest algorithms are unsupervised.

Inspired by SPORF, we propose a projection distribution that takes into account neighboring features on a manifold, while incorporating enough randomness to learn the relevant projections. At each node in the decision tree, a set of neighboring features are randomly selected using knowledge of the underlying manifold. Weighting and summing the values of the selected features yields a set of oblique projections of the data which can then be evaluated to partition the observations. We describe this proposed classification algorithm, Manifold Oblique Random Forest (MORF), in detail and show its effectiveness across simulated and real-data settings as compared to common classification algorithms. In each case, MORF performs better than algorithms that lack local feature information, while approaching, or even improving upon ConvNet performance in real data applications. Furthermore, the optimized and parallelizable open source implementation of MORF in Python is available at at https:///neurodata.io/code/.

2 Background and Related Work

2.1 Classification Let $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ be a random sample from the joint distribution F_{XY} and $D_n := \{(x_i, y_i)\}_{i=1}^n$ be our n observed data points where $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is drawn from F_{XY} 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 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 risk corresponding to $0-1$ loss, the probability of incorrect classification,

$$L(g) := \mathbb{E}[\mathbb{I}[g(X) \neq Y]] = P(g(X) \neq Y),$$

with respect to the distribution of F_{XY}. 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 risk $L^* := L(g^*(X))$. For some finite number of samples n, our goal 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 Amit and Geman [17] and subsequently codified 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 and leaf nodes. A split node is associated with a subset of the data $S = \{(x_i, y_i)\} \subseteq D_n$ which is split into two child nodes, each associated with a binary partition of S based on the value of a selected feature j. Let $e_j \in \mathbb{R}^p$ denote a unit vector in the standard basis (that is, a vector with a single one in the jth entry and the rest of the entries are zero) and $\tau \in \mathbb{R}$ a threshold value. Then, S is partitioned into the two subsets

$$S_\theta^L = \{(x_i, y_i) \mid e_j^\mathsf{T} x_i < \tau, (x_i, y_i) \in S\},$$
$$S_\theta^R = \{(x_i, y_i) \mid e_j^\mathsf{T} x_i \geq \tau, (x_i, y_i) \in S\}.$$

given the pair $\theta = \{e_j, \tau\}$. To choose the partition, the locally optimal $\theta^* = (e_j^*, \tau^*)$ pair is greedily selected from among a set of d randomly selected standard basis vectors as 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 split is chosen as

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

where $I(S) = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$ and $|\cdot|$ is the number of elements in the enclosed set. A leaf node in the decision tree is created once a partition reaches a stopping criterion, typically either falling below an impurity score threshold or a minimum number of samples [3].

To classify a feature vector x, it is evaluated at root node of the tree and split into one of the two partitions. This process is repeated recursively 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 $\hat{p}_{n,b}(y \mid x) = \frac{1}{|l_b(x)|}\sum_{i=1}^{n} \mathbb{I}[y_i = y]\mathbb{I}[x_i \in l_b(x)]$. The forest composed of B trees computes the empirical posterior probability for x by averaging over the trees $\hat{p}_n(y \mid x) = \frac{1}{B}\sum_{b=1}^{B} p_{n,b}(y \mid x)$ and classifies x per the label with the greatest empirical posterior probability [3]

$$g_n(x) = \underset{y \in \{1,\ldots,K\}}{\operatorname{argmax}} \hat{p}_n(y|x).$$

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 $D' \subset D_n$ from the full training data set.

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

2.3 Oblique Forests Sparse Projection Oblique Randomer Forests (SPORF) is a recent modification to random forest that has shown improvement over other versions [7, 8]. Recall that RF split nodes partition data along the coordinate axes by comparing the projection $e_j^\mathsf{T} x$ 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 any linear combination of axes specified by the sparse vector $a_j \in \mathbb{R}^p$. The partition

$$S_\theta^L = \{(x_i, y_i) \mid a_j^\mathsf{T} x_i < \tau, (x_i, y_i) \in S\},$$
$$S_\theta^R = \{(x_i, y_i) \mid a_j^\mathsf{T} x_i \geq \tau, (x_i, y_i) \in S\}$$

follows from our choice of $\theta = \{a_j, \tau\}$. In other words, let the dictionary $\mathcal{A}$ be the set of vectors (atoms) $\{a_j\}$, each atom a p-dimensional vector defining a possible projection $a_j^\mathsf{T} x$. In axis-aligned forests, $\mathcal{A}$ is the set of standard basis vectors $\{e_j\}_{j=1}^p$. In SPORF, the dictionary $\mathcal{A}$ can be much larger, because it includes, for example, all 2-sparse vectors. At each split node, SPORF samples d atoms from $\mathcal{A}$ according to a specified distribution. By default, each of the d atoms are randomly generated with the number of non-zero elements 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 although the size of the dictionary for SPORF is 3^p (because each of the p elements could be -1, 0, or $+1$), the atoms are sampled from a distribution heavily skewed towards sparsity controlled by the λ term. All other aspects of SPORF are the same as RF.

3 Methods

3.1 Random Projection Forests on Manifolds In the structured data setting, the dictionary of atoms $\mathcal{A} = \{a_j\}$ is modified to take advantage of *a priori* knowledge of feature locality on the underlying manifold on which the data lie. We call this 'Manifold Oblique Random Forest' (MORF). This modification constrains the space of random projection decision trees which can be learned in order to better suit certain classification tasks where relations between features may add information. By constructing features in this way, MORF learns low-level features in the structured data, such as corners in images or spikes in time-series.

As in SPORF, let $\mathcal{A}$ be a dictionary of m, p-dimensional atoms with probability density or mass function $f_{\mathcal{A}}$ over the m atoms. Each atom $a_j \in \mathcal{A}$ projects an observation x_i to a real number $a_j^T x_i$, where nonzero elements of a_j effectively weight and sum features. At each node in the decision tree, MORF selects the best split according to the Gini index over each candidate atom and threshold pair. While the nonzero indices of each atom in SPORF were mutually independent of one another, the key aspect of MORF lies in how the atoms are constructed to take advantage of feature locality.

Relations between features, or feature locality, may be abstractly encoded as a network, or graph, in which each feature represents a node and an edge between two features (an adjacency) permits an atom in $\mathcal{A}$ with nonzero weights on both of those features. If no two features are adjacent, we recover RF. If all features are pairwise adjacent, we recover SPORF. For our applications, we provide a concrete implementation targeted at translation equivariant feature locality in 1D and 2D, such as time series and images respectively. Specifically, the features of a single observation $x_i \in \mathbb{R}^p$ are viewed as organized into a 2D matrix in $\mathbb{R}^{W \times H}$ such that $W \times H = p$ such that each feature is indexed by a row and column. We term the collection of nonzero features a "patch", defined by a subset of rows and columns. If 1D locality is specified, then the patch columns are consecutive while if 2D locality is specified, then both the patch columns and rows are each consecutive. All feature values in a patch are weighted and summed to produce a candidate univariate feature to split on. Thus, a patch corresponds to a unique atom $a \in \mathbb{R}^p$ where $a_i \neq 0$ if i is in the patch and $a_i = 0$ otherwise; the candidate feature is calculated through sparse matrix multiplication of the observation feature vector and atom.

At each split node, a set of atoms in the form of patches are randomly sampled to produce candidate features across observations. MORF accepts hyperparameters defining the minimum and maximum number of patch rows $\{h_{min}, h_{max}\}$ and columns $\{w_{min}, w_{max}\}$, respectively. To sample a patch, first the number of rows h and columns w are independently and uniformly sampled,

$$h \sim \mathcal{U}\{h_{min}, h_{max}\}$$
$$w \sim \mathcal{U}\{w_{min}, w_{max}\},$$

between respective minima and maxima inclusively. As columns are consecutive, a reference leftmost column in the unraveled matrix is sampled as $u \sim \mathcal{U}\{-h+1, H\}$. If 2D locality is specified, then a reference upper row is sampled as $v \sim \mathcal{U}\{-w+1, W\}$. In both cases, the reference column and row may be outside of the matrix so that each feature has an equally likely chance of being included in a patch. The region outside of the matrix is ignored, effectively a zero-padded boundary. The algorithm pseudocode is equivalent to that of SPORF except for the distribution $f_{\mathcal{A}}$ described above which can be seen in detail in Appendix B.

In our experiments, the atom weights were limited to values of 1 and 0 to limit combinatorial complexity but domain-specific atom design may be desired in some settings along with further task-specific atoms. For graph-valued data, one may consider sampling a collection of neighboring edges or nodes [18].

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, 19], and here we introduce a projection forest specific metric which counts the the number of times a given feature was used in projections across the ensemble. Formally, a decision tree T in the trained forest F contains a set of split nodes, where each node $s \in T$ is associated with an atom a_s^* from the dictionary of atoms in the forest $\mathcal{A}_{\mathrm{F}}$ and a threshold that partition the feature space

according to the projection $a_s^{*T}x$. Thus, the indices corresponding to nonzero elements of a_s^* indicate important features used in the projection. The importance of feature k, denoted π_k, is calculated as

$$\pi_k = \frac{1}{|\mathcal{A}_F|} \sum_T \sum_{s \in T_S} \mathbb{I}(a_{sk}^* \neq 0),$$

the number of times it is used in a projection, across all decision trees and split. These counts represent the relative importance of each feature in making a correct classification. Such a method applies to RF,SPORF, and MORF although different results between them would be expected due to different dictionary distributions.

4 Theoretical Results

4.1 Classifier Consistency According to Bickel and Doksum [20], the least we can ask of our classification rule is for it to be consistent. A classification rule is consistent whenever its probability of misclassification converges to the minimum (Bayes) error as our sample size increases. In the simple k-nearest neighbor setting, consistency was proven using the histogram rule whereby the feature space is partitioned into regions whose radii go to zero, but slow enough for the regions to be populated with sets of size going to infinity [21]. Random forests are effectively adaptive nearest neighbor classifiers [22], partitioning the feature space into hyper-rectangles whose contents are a variable number of nearest neighbors.

Recent work has shown that certain axis-aligned random forests are consistent [23, 24]. Typical decision trees are biased because they learn partitions using the same samples that vote in each partition (leaf node). *Honesty* is a mild condition that removes this bias by requiring 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 [5, 24–26]. In practice this is done using holdout set per tree.

Although oblique decision trees do not create hyper-rectangular partitions, they do partition the feature space into a finite number of (possibly unbounded) convex polytopes (see Appendix C for a proof). Each polytope region corresponds to a leaf node with a constant classification label per leaf. That oblique trees yield convex polytopes that are not necessarily hyper-rectangles is the only difference compared to axis-aligned trees, and the basis as to why the consistency results of Athey et al. [24] can be extended to the oblique setting.

We show that given a minor restriction on the dictionary of a oblique random regression forest, Theorem 5 of Athey et al. [24] holds, and thus its point estimates are asymptotically normal for appropriately specified quantities. As a corollary, the classification rule for an oblique random forest under appropriate conditions is consistent and thus its error converges to the Bayes error (see Appendix A for full proofs). We repeat the specifications and the set of assumptions made by Athey et al. [24] below for reference. The second Specification is new and restricts the set of possible dictionaries.

Specifications

1. All trees are symmetric, in that their output is invariant to permuting the indices of training examples; make balanced splits, in the sense that every split puts at least a fraction *w* of the observations in the parent node into each child, for some $w > 0$; and are randomized in such a way that, at every split, the probability that the tree splits on the j-th feature is bounded from below by some $\alpha > 0$. The forest is honest and built via subsampling with subsample size s satisfying $s/n \to 0$ and $s \to \infty$ [24].
2. The oblique dictionary $\mathcal{A}$ is finite and contains the set of standard basis vectors $\{e_i\}_{i=1}^p$, each with a fixed nonzero probability of being selected at each split node.

Assumptions

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

3. There exists a density f over $\mathcal{X}$ that is bounded away from zero and infinity. That is, for all $x \in \mathcal{X}$ there exists a $\varepsilon > 0$ such that $\varepsilon < f(x) < \frac{1}{\varepsilon}$.

Theorem 1. *Under Assumptions 1-3, the posterior probability estimate from a oblique random forest built to Specifications 1-2 is consistent.*

Corollary 1. *Under Assumptions 1-3, the classification rule from a oblique random forest built to Specifications 1-2 is consistent, i.e.*

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

Note that Corollary 1 applies to both oblique forests with unstructured atoms such as SPORF, as well as structured atoms such as MORF.

The Lipschitz assumption is a frequent one taken in the literature on random forests [24]. It intuitively makes sense *a priori* that small deviations in x should lead to small deviations in the class probability. Honesty restricts the amount of data available for each of learning the partitions and voting, but is well suited to the subsampling of data for each tree in the forest and in some cases is actually beneficial [23]. As in Athey et al. [24], however, this theorem is limited to continuous-valued features which rules out certain classes of data.

4.2 Time and Space Complexity Theoretical analysis of the time and space complexity of decision trees during training is difficult without making assumptions on the data as a tree's possible structure occupies a combinatorially large space. While the worst case may not be expected, the bound it presents is too large to be helpful in typical scenarios. We examine a simplified average setting in which the possible sizes induced at each partition node are equally likely. It has been posited and supported empirically that this is a lower bound for the true average case in a RF [27]. One reason that worse-than average cases may occur is that when none of the candidate features are informative, edge splits are frequent and lead to deep trees [27]. The candidate features in SPORF and MORF are combinations of individual features and we expect this greater flexibility to reduce the chance that no candidate features are informative.

Let a forest have T trees, d candidate features at each split node, and n training samples. At each split node in RF, the complexity is $O(dn \log n)$ to sort observations along each feature using an optimal sorting algorithm [27]. In the average case, the time complexity for RF is then $\mathcal{O}(Tdn \log^2 n)$ [27]. MORF, like SPORF, utilizes sparse matrix multiplication to compute weighted sum while sorting observations at each split node. Thus, letting H and W denote the maximum height and width of a patch in MORF, the time complexity for MORF is $\mathcal{O}(TdHWn \log^2 n)$, because for each of the d features we must make HW multiplication operations. However, as we will show empirically, MORF can find better partitions and thus learn smaller trees. Note that random forests are embarrassingly parallelizable and the scaling in T can be scaled down linearly by the number of processors in parallel (with some overhead).

The space complexity during training defines how much physical memory is needed with respect to the data and choice of hyperparameters at any given point during training. Note, this is different from the space required to store the trained forest. First, the data matrix of size np must be kept in memory throughout training. At each split node, a set of d candidate feature patches are created at cost dHW. To evaluate a candidate feature for a classification problem with K classes, two arrays of length K are needed to store the class counts in order to calculate the score of the resulting partition. These terms are all additive, which for T trees being trained in parallel leads to a MORF space complexity during training of $O(np + TdHW)$ where we have dropped the K term as it is dominated by n.

5 Simulation Experiments We examine the performance of MORF in terms of predictive accuracy and runtime in three simulations highlighting 1D and 2D manifolds. In all cases. MORF outperforms methods that do not consider feature locality as well as ConvNets in some cases.

5.1 Three simulated manifolds We evaluated MORF in three simulation settings to show its ability to take advantage of structure in data. MORF was compared to a set of traditional classifiers (and SPORF) that learn from the raw features. For each experiment, we used our open source implementation of

MORF as well as SPORF and the RF implementation contained in the SPORF, each with 500 trees. Other classifiers were run from the Scikit-learn Python package [28] and the gradient boosted tree XGBoost (XGB) was run using its Python implementation [29]. Additionally, we tested against a Convolutional Deep Network (ConvNet) built using PyTorch [30] with two convolution layers, ReLU activations, and maxpooling, followed by dropout and a densely connected hidden layer.

Method hyper-parameters were left as defaults except for the ConvNets and MORF which are each specific to the structure of the data and so must be changed. Thus, a well-performing ConvNet architecture was selected and MORF minimum and maximum patch sizes were optimized using a grid search over a range of potential values as well as max_features (*mtry*) whose magnitude is dependent on the size of the combinatorial space of possible features determined by the patch size. See Appendix C for details on the hyperparameters and network architectures across experiments.

Experiment (A) is a non-Euclidean cyclic manifold example inspired by Younes [31] in which the discriminating information is solely contained in the structure of the data. Each observation is a discretization of a circle's perimeter into a one dimensional feature vector with 100 features and 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. Because features are arranged on a circle, segments can wrap around the cyclic feature vector. Figure 1(A) shows examples from the two classes and classification results across various sample sizes.

Experiment (B) is 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, $k \sim \text{Poisson}(\lambda = 10)$ 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 and so a classifier cannot be learned without learning the structure of the data. Figure 1(B) shows examples from the two classes and classification results across various sample sizes.

Experiment (C) is a signal classification problem highlighting the presence of structure in time series data. One class consists of 100 values of Gaussian noise independent and identically distributed (iid) while the second class has an added exponentially decaying unit step (u) beginning at time 20.

$$X_t^{(0)} = \epsilon_t$$
$$X_t^{(1)} = u(t-20)\exp^{(t-20)} + \epsilon_t, \qquad \epsilon \overset{iid}{\sim} \mathcal{N}(0,1)$$

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

In all three simulation settings, MORF outperforms all other classifiers that ignore the local structure, doing especially better at low sample sizes. As compared with ConvNets, MORF sometimes does better, and sometimes worse. The variance across five repeated runs was negligible across sample sizes. The performance of MORF and ConvNets are particularly good in the discretized circle simulation for which most other classifiers perform at chance levels. MORF dominates in the signal classification problem for all sample sizes, most likely because of the ability to learn wide patches which approaches the Bayes classifier. Results on these experiments using uniformly distributed atom weights in between 0 and 1 showed no improvement but increased training time and so were omitted.

We compare the empirical complexity of MORF, SPORF, and RF (from the SPORF package[1]) in Figure 2. Although projection forests required more computations at each partition node during training, as outlined in Section 4.2, MORF is able to learn less complex trees in all cases and sample sizes.

Each simulated experiment was run on CPUs and allocated 45 cores for parallel processing. The resulting train and test times as a function of the number of training samples are plotted in Figure 3. MORF has train and test times slightly longer than those of SPORF. This cost comes at the benefit of less complex trees, as show in in Figure 2. The other method to utilize feature locality, the ConvNet, took

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

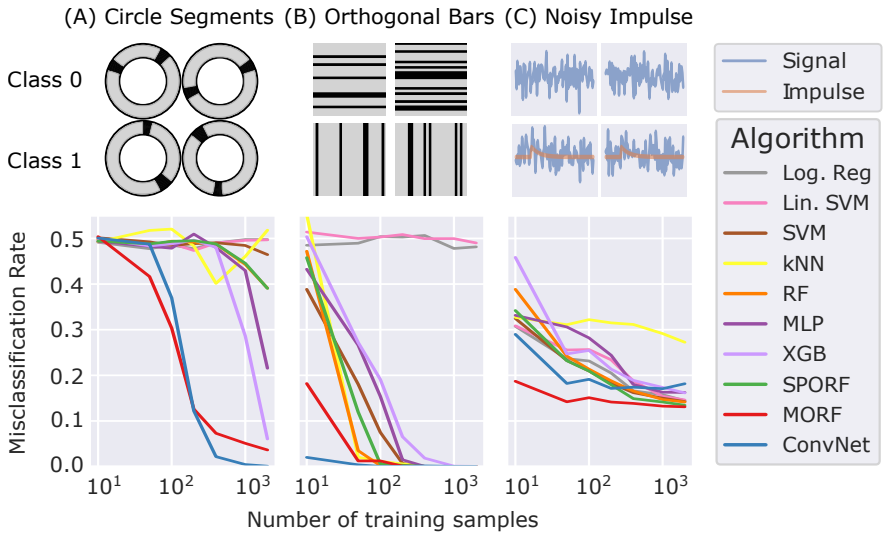

Figure 1: MORF outperforms other algorithms in three two-class classification settings. Upper row shows examples of simulated data from each setting and class. Lower row shows misclassification rate in each setting, tested on 10,000 test samples. **(A)** Two segments in a discretized circle. Segment 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).

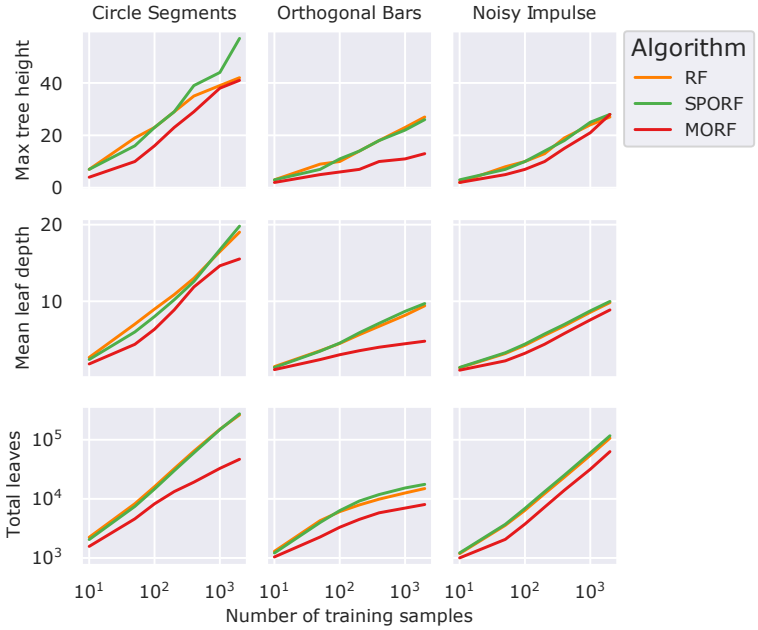

Figure 2: Empirical random forest complexities in each simulation with respect to their maximum tree height, mean leaf depth, and total number of leaves across training sample size. MORF is able to learn simpler trees in all cases due to its restricted projection distribution.

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 [32].

5.2 A simulated multivariate time-series Experiment (D) here demonstrates how multivariate data with implicit structure can be learned by MORF. The simulation is run as in the prior simulations in Section 5.1. We simulate a multivariate time-series problem, where the signals are governed by a

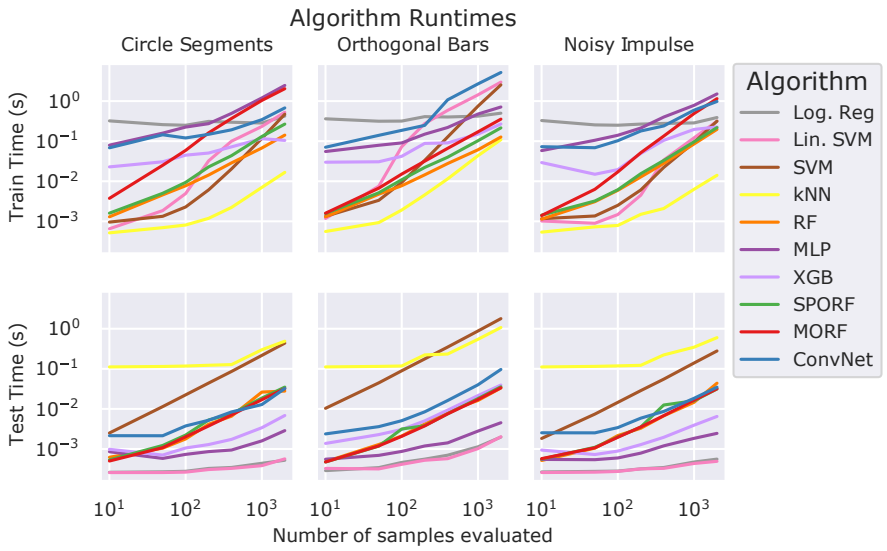

Figure 3: Algorithm train times (above) and test times (below) across increasing sample sizes. MORF runtime is not particularly costly and well below ConvNet runtime in most examples.

linear dynamical system of the form

$$x(t+1) = Ax(t) + B_i u(t),$$

where the governing linear state matrix, $A \in \mathbb{R}^{3\times3}$, is the same, but the class separation is modeled by the input matrices, $B_i \in \mathbb{R}^{3\times3}$, defined below. The input to the system, $u(t)$, is the same for all classes. The system is set up such that the pair (A, B) is controllable, and that A is marginally stable (i.e. eigenvalues, $|\lambda| \leq 1$ for all eigenvalues of A.

$$B_0 = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0.5 & 0.1 & 0.1 \\ 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.5 \end{pmatrix}$$

An input $u(t)$ was applied at a random time point selected between the 20th and 40th time point of the simulated time series. A total of 100 time points were simulated and MORF was compared to a suite of other classification algorithms over varying sample sizes. Note that the signals have the exact same dynamics encoded through the A matrix, but the input-output relationships are different. This simulates a setting common in multivariate time-series classification (e.g. EEG, see Section 6.2) where there are signals collected over time that one hypothesizes to be relevant to a task, yet the researcher does not know *a priori* what signals are actually relevant and so they collect as much data as possible. This results in the standard curse-of-dimensionality. However, it is assumed that signals relevant to the task live on a low-dimensional manifold, and the goal is to have a model learn the structure of this manifold for the sake of classification. Figure 4 shows examples from the two classes and classification results across various sample sizes.

A challenging aspect of experiment (D) is that i) the linear state dynamics governed by the A matrix are the same and is considerably larger in norm compared to B_i, and ii) the time at which $u(t)$ is applied is random within a small interval. This simulation setting motivates settings where there is a dynamical system with input, such as electroencephalogram (EEG) at rest with dynamics modeled as a linear system, and then a stimulus is applied in the form of input $u(t)$ [33–36]. Depending on the stimulus applied, this might affect the system in different ways through B_i. This is very general setting in where the stimulus can be a flash of light, or indication of movement, or even a direct stimuli to evoke seizures.

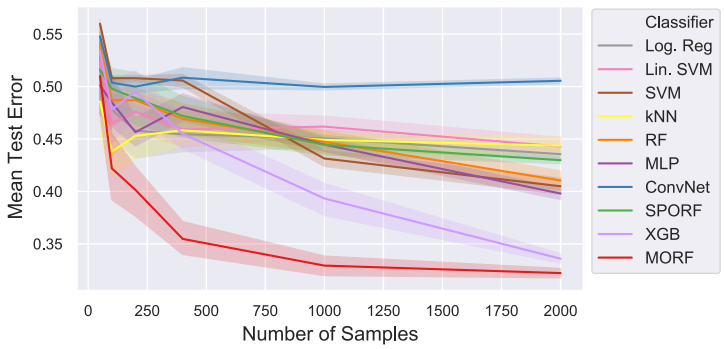

Figure 4: **(D) Multivariate data embedded in a manifold** Algorithm comparisons on classifying two classes of multivariate data. The data consists of samples from two classes of a 3-dimensional stable linear dynamical system with input. Samples are constructed as $X_i \in \mathbb{R}^{3\times T}$, where now data points over time are correlated. MORF learns the structure significantly faster then the other classifiers, with XGBOOST requiring more sample to achieve the same test error rate.

6 Experiments on Real Data We next evaluated MORF on three real data sets with varying manifold structure, sample sizes and classification goals. In each dataset, there is the notion of either a 1D[2] or 2D[3] manifold on which we have *a priori* knowledge of feature locality, time and images respectively. We compare results against a suite of classification algorithms as before.

6.1 2D Locality: MNIST Digit Classification MORF's performance was evaluated on the MNIST dataset, a collection of handwritten digits stored in 28 by 28 square images [37], and compared to the algorithms used in the simulations. 10,000 images were held out for testing and the remaining 50,000 images were used for training. The results are displayed in Figure 5a. Hyperparameters are as described in the simulations, see Appendix C for details. MORF showed an improvement over the other algorithms as compared to ConvNets.

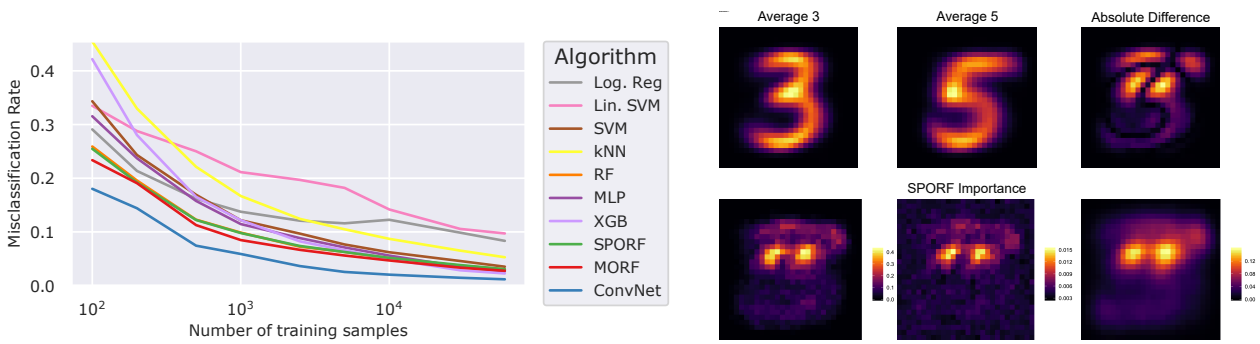

Figure 5: (**Left**) MORF performance on the MNIST digit classification problem improves prediction accuracy over all other non-ConvNet algorithms, notably in small sample sizes. (**Right**) The averages all images from MNIST labeled 3 and 5, respectively, and their absolute difference (top row). Feature importance from MORF (bottom right) shows less noise than SPORF (bottom middle) and is smoother than RF (bottom left).

We then evaluated the ability of MORF to identify important 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.

The feature importance of each pixel is shown in Figure 5b. MORF visibly results in a smoother

[2]Experiments can be found at https://github.com/adam2392/morf-demo
[3]Experiments can be found at https://github.com/neurodata/SPORF/tree/structured/Python

pixel importance, a result most likely from the continuity of neighboring pixels in selected projections. Although Tomita et al. [8] 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. MORF, however, shows little or no importance of these background pixels by virtue of the modified projection distribution.

6.2 1D Locality: Predicting Surgical Outcome In Epilepsy Patients With Intracranial EEG

MORF's performance was next evaluated on an epilepsy dataset presented in Li et al. [38], Li et al. [39]. The dataset comprises 91 subjects with multiple seizures recorded with intracranial electroencephalogram (iEEG). Clinicians annotate a subset of implanted electrodes as part of the clinically hypothesized epileptogenic zone (EZ), and then perform subsequent surgery to resect a super set of those regions. The goal of computational EZ localization is to define a model that extracts features of the iEEG data that can separate healthy from epileptic regions. The classification task is to predict surgical outcome of success (seizure free), or failure (seizure recurrence), after a surgical resection is performed on drug resistant epilepsy patients conditioned on the clinically hypothesized EZ regions. If a feature is informative, then it will highly correlate with the clinical EZ when a surgical outcome is successful and vice versa when not successful. In Li et al. [38], Li et al. [39], a feature of the data, "neural fragility" was computed from the data, which is represented as a spatiotemporal heatmap of channels-by-time. The classification task specifically takes in a data points that are $X_i \in \mathbb{R}^{H \times W}$ of dimension (20×105), where there are 20 quantiles summarizing a distribution of neural fragility and 105 time points around seizure onset. There are 10 quantiles for the neural fragility of electrodes in the clinically hypothesized EZ and 10 quantiles for the neural fragility of electrodes in the rest of the implanted electrodes. The goal is to take X_i and predict $y_i \in \{0, 1\}$, where 0 stands for failed surgical outcome and 1 stands for successful surgical outcome. The classification is setup this way because the clinically annotated EZ electrodes are imperfect and not always representative of the true underlying EZ, which is not observable. In the original paper, there is preprocessing of the data in the form of a thresholding step, which was done to improve the model performance. However, in this setting, we perform no preprocessing before the classification task to demonstrate fair performance on the "raw" data. For full details on the dataset and clinical problem, we refer the readers to Li et al. [38], Li et al. [39].

In Figure 6a, we compare MORF with default hyperparameters, specified in the SPORF package, against standard classification algorithms as in the simulations. The sample sizes for training are relatively low with 60% of subjects used for training and the rest for the held-out test set. The set of subjects in each cross-validation index are the same across all classifiers, thus enabling a fair comparison. In some folds, it is seen that all the classifiers perform very poorly. This occurs most likely because the data are noisy, the implanted iEEG electrodes are not perfect and some subjects are very difficult to treat. Moreover, there are only 91 subjects total used in this classification task. Even in this challenging classification setting, we observe that MORF is able to achieve a superior performance measured by AUC (Figure 6b). In terms of the Cohen's effect size, MORF is significantly (p-value ≤ 0.05) more accurate than all other algorithms besides SPORF per a Wilcoxon paired sign test across the 10 cross-validation folds. We observe that ConvNets perform at chance level on the test set, completely overfitting to the training set in this limited sample size setting.

6.3 1D Locality: Predicting Movement Direction With Intracranial EEG

MORF's performance was next evaluated on stereotactic electroencephalogram (sEEG) data recorded in epilepsy patients undergoing a motor control task presented in Kerr et al. [33], Breault et al. [40]. The classification task presented here is to predict movement direction (up, down, left, or right) based on the sEEG data alone, rather then performing explicit feature engineering, such as computing power in frequency bands. We compare MORF to other classification algorithms. The interesting aspect of this data is that there are no motor regions recorded. Thus, our hypothesis is that only a subset of the

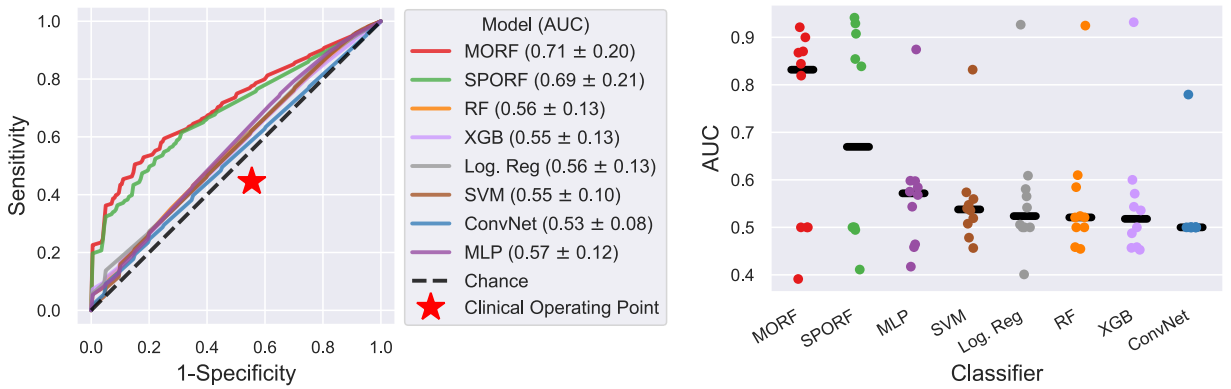

Figure 6: MORF performance on the epilepsy seizure outcome prediction problem improves prediction accuracy over all other algorithms in 10-fold cross validation (CV). **(Left)** Shows a ROC curve with the mean ROC curve plotted AUC for each classifier. **(Right)** Shows a strip plot of the AUC values in 10-fold CV with the median marked in each setting (solid black line). Note that even the ConvNet and multi-layer perceptron (MLP) perform poorly, most likely because the sample size is very low. Compared to the next best non SPORF classifier, MORF improves over the MLP with a Cohen's D effect size of 0.83 (95% CI = [2.32, -0.102]).

recording electrodes over time are important in decoding movement directionality. This is analogous to Experiment D mentioned in Section 5.2. Each subject performed the task for several trials, each consisting of a movement instruction followed by a movement generated by the subject. Using only the sEEG data, we sought to decode the movement directionality. For full details on the dataset and clinical problem, we refer the readers to Kerr et al. [33], Breault et al. [40].

Here we perform 5-fold cross validation for each subject including all the sEEG recording electrodes time-locked to a movement onset marking. Each fold is *a priori* generated per subject, where there is a set of testing trials left out. The overall task is very challenging because there are no motor brain regions being recorded. Nonetheless, we expect that other brain regions are involved in the motor control process. MORF is able to achieve a superior performance measured by AUC relative to the other classifiers, as seen in Figure 7. Notably, MORF and SPORF perform the best in this setting with a limited set of training samples, whereas the ConvNet performs slightly better then chance on the test set, overfitting to training set. Across the set of all folds of all subjects, MORF was never worse than another classifier in terms of the median pairwise difference in Cohen's kappa while being significantly (p-value ≤ 0.05) better than the MLP and ConvNet per a Wilcoxon paired sign test on those same pairwise differences.

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, and have shown great success, but ignore feature continuity and specialize for specific data modalities. We expand upon sparse oblique projections and introduced manifold-aware projection distributions that exploits prior knowledge of the local topology of a feature space to improve learning rates and accuracy for classification. The open source implementation of MORF subsumes SPORF and provides a flexible classification method for a variety of data modalities and tailored projection dictionaries. We showed in various settings that appropriate domain knowledge can improve the projection distribution and better match ConvNet results (or even outperforming ConvNets significantly) while maintaining interpretability, fast run time, and theoretical justification.

The flexibility in choices of MORF's dictionary opens a much larger combinatorial space to sample from compared to a traditional random forest. More complex possibilities may lead to improved per-

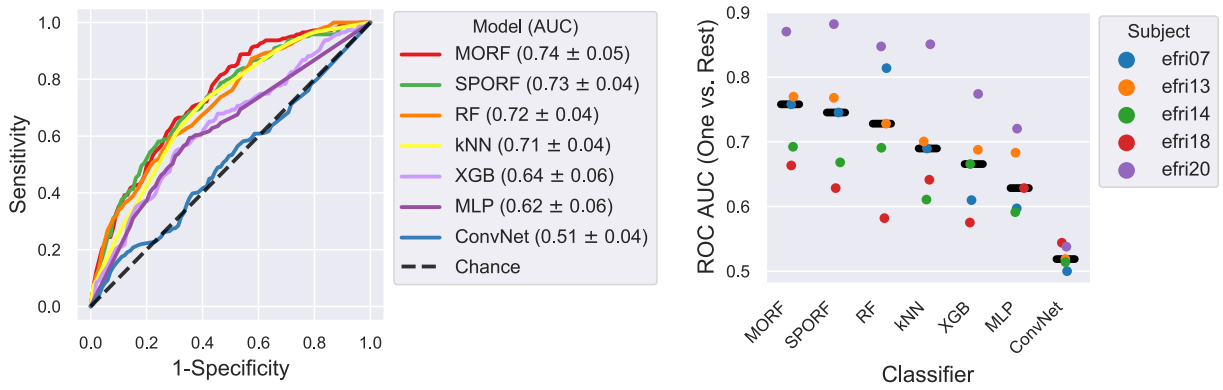

Figure 7: MORF performance on decoding movement direction from the raw sEEG data in non-motor brain regions. Subjects are undergoing a motor-control task. The naming of subjects is simply derived from their clinical monitoring session and does not reflect any specific numbering scheme. **(Left)** Shows ROC curve of moving down in the motor task decoded with all classifiers on the same set of data and their AUC scores. **(Right)** Shows a summary AUC stripplot where each dot represents the held-out trial median AUC score for a certain subject over 5-fold CV, and the median of the overall AUC for each classifier is shown (solid black line). In almost all subjects, MORF gains in AUC compared to the other classifiers with fixed hyperparameters and fixed trials in each of the 5 folds.

formance, but potentially at the cost of greater sampling requirements. Similarly, research into other task-specific projection dictionaries may lead to improved results in computer vision tasks, through better texture quantification for instance, or in other manifold-valued settings such as graphs. Although, unlike ConvNets, MORF is not globally translation equivariant, it can be locally translation equivariant given atoms reminiscent of Gabor filters, for instance. Without local equivariance, discriminative features must be constant in their indices or the training data must be rich enough to fully encapsulate possible observations [14]. The MORF projection distributions may also be incorporated into other state of the art Forest algorithms such as XGBOOST. Additionally, the fact that oblique decision forests lead to partitions of convex polytopes is of interest in that it has been shown that deep nets with Rectified Linear Units (ReLUs) or hard tanh activation layers also partition the feature space into convex polytopes with different linear functions on each region [41]. This shared "partition and vote scheme" offers insight into their relationship with one another as well as the functioning of the brain [42]

8 Acknowledgements This work is supported by the Defense Advanced Research Projects Agency (DARPA) Lifelong Learning Machines program through contract FA8650-18-2-7834 and through funding from Microsoft Research. AL is supported by NIH T32 EB003383, the NSF GRFP (DGE-1746891), the Arcs Chapter Scholarship, Whitaker Fellowship and the Chateaubriand Fellowship. The authors have no conflicts of interest to declare.

References

[1] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

[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, pages 161–168, New York, NY, USA, 2006. ACM. 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. Springer New York Inc., New York, NY, USA, 2001.

[4] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

ISSN 0885-6125. doi: 10.1023/A:1022648800760.

[5] Gerard Biau, Luc Devroye, and Gabor Lugosi. Consistency of Random Forests and Other Averaging Classifiers. *J. Mach. Learn. Res.*, 9:2015–2033, June 2008. ISSN 1532-4435.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

[7] 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*, pages 498–506, 2017. doi: 10.1137/1.9781611974973.56.

[8] 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. *arXiv e-prints*, art. arXiv:1506.03410, Jun 2015.

[9] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep Neural Decision Forests. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1475, Santiago, Chile, December 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.172.

[10] Gérard Biau, Erwan Scornet, and Johannes Welbl. Neural Random Forests. *arXiv:1604.07143 [cs, math, stat]*, April 2018. arXiv: 1604.07143.

[11] 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)*, volume 2, pages 775–781 vol. 2, June 2005. doi: 10.1109/CVPR.2005.288.

[12] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, Nov 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.70.

[13] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. doi: 10.1109/ICCV.2007.4409066.

[14] 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*, pages 1297–1304, June 2011. doi: 10.1109/CVPR.2011.5995316.

[15] 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*, pages 2190–2197, Nov 2011. doi: 10.1109/ICCV.2011.6126496.

[16] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Found. Trends. Comput. Graph. Vis.*, 7(2–3):81–227, February 2012. ISSN 1572-2740. doi: 10.1561/0600000035.

[17] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Comput.*, 9(7):1545–1588, October 1997.

[18] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2020. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[19] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *ArXiv*, abs/1705.07874, 2017.

[20] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics.* Holden-Day, 1977.

[21] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probablistic Theory of Pattern Recognition*, volume 31. 01 1996. ISBN 978-1-4612-6877-2. doi: 10.1007/978-1-4612-0711-5.

[22] Yi Lin and Yongho Jeon. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006. ISSN 0162-1459. URL https://www.jstor.org/

stable/27590719. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

[23] Stefan Wager and Susan Athey. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523):1228–1242, July 2018. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.2017.1319839. URL https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1319839.

[24] Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *Ann. Statist.*, 47 (2):1148–1178, 04 2019. doi: 10.1214/18-AOS1709. URL https://doi.org/10.1214/18-AOS1709.

[25] Misha Denil, David Matheson, and Nando De Freitas. Narrowing the gap: Random forests in theory and in practice. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 665–673, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/denil14.html.

[26] Stefan Wager and Guenther Walther. Adaptive Concentration of Regression Trees, with Application to Random Forests. *arXiv:1503.06388 [math, stat]*, April 2016. URL http://arxiv.org/abs/1503.06388. arXiv: 1503.06388.

[27] Gilles Louppe. Understanding random forests: From theory to practice, 2014.

[28] 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. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] 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, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

[30] 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.

[31] Laurent Younes. Diffeomorphic learning, 2018.

[32] 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, pages 855–863, Cambridge, MA, USA, 2014. MIT Press.

[33] Matthew S D Kerr, Pierre Sacré, Kevin Kahn, Hyun-Joo Park, Mathew Johnson, James Lee, Susan Thompson, Juan Bulacio, Jaes Jones, Jorge González-Martínez, Catherine Liégeois-Chauvel, Sridevi V Sarma, and John T Gale. The Role of Associative Cortices and Hippocampus during Movement Perturbations. *Front. Neural Circuits*, 11:26, 2017. ISSN 1662-5110. doi: 10.3389/fncir.2017.00026. URL http://www.ncbi.nlm.nih.gov/pubmed/28469563http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5395558.

[34] Adam Li, Zachary Fitzgerald, Jennifer Hopp, Emily Johnson, Nathan Crone, Juan Bulacio, Jorge Martinez-Gonzalez, Sara Inati, Kareem Zaghloul, and Sridevi V. Sarma. Virtual Cortical Stimulation Mapping of Epilepsy Networks to Localize the Epileptogenic Zone. In *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, volume 2019, pages 2328–2331. Institute of Electrical and Electronics Engineers (IEEE), oct 2019. ISBN 9781538613115. doi: 10.1109/EMBC.2019.8856591. URL https://pubmed.ncbi.nlm.nih.gov/31946366/.

[35] A. Li, K.M. Gunnarsdottir, S. Inati, K. Zaghloul, J. Gale, J. Bulacio, J. Martinez-Gonzalez, and S.V. Sarma. Linear time-varying model characterizes invasive EEG signals generated from complex epileptic networks. In *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, 2017. ISBN 9781509028092. doi: 10.1109/EMBC.2017.8037439.

[36] Marggie Jones, Barry McDermott, Bárbara Luz Oliveira, Aoife O'Brien, Declan Coogan, Mark Lang, Niamh Moriarty, Eilis Dowd, Leo Quinlan, Brian Mc Ginley, Eoghan Dunne, David Newell, Emily Porter, Muhammad Adnan Elahi, Martin O' Halloran, and Atif Shahzad. Gamma Band Light

Stimulation in Human Case Studies: Groundwork for Potential Alzheimer's Disease Treatment. *J. Alzheimers. Dis.*, 70(1):171–185, 2019. ISSN 1875-8908. doi: 10.3233/JAD-190299. URL https://pubmed.ncbi.nlm.nih.gov/31156180https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6700637/.

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

[38] Adam Li, Chester Huynh, Zachary Fitzgerald, Iahn Cajigas, Damian Brusko, Jonathan Jagid, Angel Claudio, Andres Kanner, Jennifer Hopp, Stephanie Chen, Jennifer Haagensen, Emily Johnson, William Anderson, Nathan Crone, Sara Inati, Kareem Zaghloul, Juan Bulacio, Jorge Gonzalez-Martinez, and Sridevi V. Sarma. Neural fragility as an eeg marker of the seizure onset zone. *bioRxiv*, 2021. doi: 10.1101/862797. URL https://www.biorxiv.org/content/early/2021/02/02/862797.

[39] A. Li, S. Inati, K. Zaghloul, and S. Sarma. Fragility in epileptic networks: The epileptogenic zone. In *2017 American Control Conference (ACC)*, pages 2817–2822, 2017. doi: 10.23919/ACC.2017.7963378.

[40] Macauley Smith Breault, Zachary B. Fitzgerald, Pierre Sacré, John T. Gale, Sridevi V. Sarma, and Jorge A. González-Martínez. Non-motor brain regions in non-dominant hemisphere are influential in decoding movement speed. *Frontiers in Neuroscience*, 13:715, 2019. ISSN 1662-453X. doi: 10.3389/fnins.2019.00715. URL https://www.frontiersin.org/article/10.3389/fnins.2019.00715.

[41] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the Expressive Power of Deep Neural Networks. *arXiv:1606.05336 [cs, stat]*, June 2017. URL http://arxiv.org/abs/1606.05336. arXiv: 1606.05336.

[42] Carey E. Priebe, Joshua T. Vogelstein, Florian Engert, and Christopher M. White. Modern Machine Learning: Partition & Vote. *bioRxiv*, page 2020.04.29.068460, April 2020. doi: 10.1101/2020.04.29.068460. URL https://www.biorxiv.org/content/10.1101/2020.04.29.068460v1. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[43] Richard Guo, Ronak Mehta, Jesus Arroyo, Hayden Helm, Cencheng Shen, and Joshua T. Vogelstein. Estimating Information-Theoretic Quantities with Uncertainty Forests. *arXiv:1907.00325 [cs, stat]*, November 2019. URL http://arxiv.org/abs/1907.00325. arXiv: 1907.00325.

[44] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5: 101–141, December 2004. ISSN 1532-4435.

Appendices.

Appendix A. Proofs.

A.1 Convex Polytope Partition Results As mentioned in Section 4.1, a random projection tree partitions the feature space into a finite number of (possibly unbounded) convex polytopes. The proof of that is as follows.

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. ■

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 the finite sample rule g_n and the Bayes decision rule g^*, respectively.

Proof of Theorem 1: Consistent oblique random forests posterior estimates The Generalized Random Forest (GRF) results of Athey et al. [24] establish conditions on which random forests making axis-aligned splits lead to consistent regression estimates. Critically the use of splits which may be linear combinations of features does not violate the GRF properties given our Specification 1-2 and Assumptions 1-3. Specifically, the GRF result relies on the consistency result of Theorem 1 in Wager and Athey [23]. We need only to verify foundations of Theorem 1 [23] that take into account the use of axis-aligned splits, those being Theorems 3 and 5 of Wager and Athey [23]. Thus, it suffices to confirm that those results are unchanged under oblique splits and our added Specification 2.

Theorem 3 [23] proves an asymptotic upper bound on the diameter of a leaf, $\text{diam}(L(x))$, by applying an asymptotic upper bound result from Lemma 2 [23] on the diameter of dimension j in the leaf, $\text{diam}_j(L(x))$. The leaf $L(x)$ is a polytope formed from a combination of axis-aligned and oblique splits. Considering only the axis-aligned conditions forming $L(x)$, by the positive probability of splitting on each dimension per Specification 2, the upper bound of Lemma 2 Wager and Athey [23] holds. As the addition of oblique conditions cannot increase the size of the leaf, the same upper bound holds. Similarly, the diameter $\text{diam}(L(x))$ of the leaf is smaller than the diameter of the polytope formed from just axis-aligned conditions. By the diameter bound from Lemma 2 [26] of each feature, the upper bound of Theorem 3 [23] holds for the axis-aligned polytope and so also $L(x)$.

Theorem 5 [23] hinges on Lemma 4 [23] which brings up the concept of a *potential nearest neighbor* (PNN) [23] [26].

Definition 1. *$x_i \in \{x_1, \ldots, x_s\} \in \{\mathbb{R}^p\}^s$ is a potential nearest neighbor (PNN) of $x \in \mathbb{R}^p$, if there is an axis-aligned hyperrectangle containing only x and x_i. A k-PNN set is a collection of k points and x in an axis-aligned hyperrectangle containing no other points. A predictor T for x is a k-PNN predictor if given*

$$\{z\} = \{(x_1, y_1), \ldots, (x_s, y_s)\} \in \{\mathbb{R}^p \times \mathcal{Y}\}^s,$$

T outputs the average of the y_i among a k-PNN set of x with respect to the x_i.

In the case of oblique split decision trees, we have the following result.

Lemma 1. *Let T be a decision tree which makes oblique splits (including axis-aligned splits) at each interior node with finite dictionary $\mathcal{A}$ of m vectors encoding the set of allowable oblique axes. If T has leaves between size k and $2k-1$, then T is a k-PNN predictor on $\mathbb{R}^m$.*

Proof. Let $\mathcal{X}$ denote the vector space of possible samples, where $x \in \mathcal{X} \subset \mathbb{R}^p$. Since $\mathcal{A} \in \{\mathbb{R}^p\}^m$, let $A \in \mathbb{R}^{p \times m}$ denote the matrix whose columns are the elements of $\mathcal{A}$. Then $\mathcal{B} = A^T\mathcal{X} \subset \mathbb{R}^m$ is a vector space of dimension at most $\min(p, m)$ in a space of dimension m. Bases of $\mathcal{B}$ correspond to bases or oblique combinations of them from $\mathcal{X}$ and so every oblique split in $\mathcal{X}$ is an axis-aligned split in $\mathcal{B}$. The points which fall into a leaf of T are the only points which satisfy the linear system formed by the set of splits, which are the only points that fall into the hyperrectangle in $\mathcal{B}$ defined by that system. As any decision tree making axis-aligned splits with leaves of sizes between k and $2k-1$ is a k-PNN predictor [22], T is thus a k-PNN predictor in $\mathcal{B} \subset \mathbb{R}^m$. ∎

In this expanded feature space from which we can view oblique splits as axis-aligned, as in the above proof, we can scale down the marginals to be within $[0, 1]$. While this space no longer satisfies the Lipschitz criteria and if $m > p$ may have a density of 0 at all points outside of the p dimensional subspace, Lemma 4 [23] requires neither of these conditions from the original assumptions. So it holds, albeit with the finite constant m instead of p. Thus Theorem 5 [23] holds with simply a modified constant which doesn't change the final established asymptotics in Theorem 1 [23]. So Theorem 1 [23] holds in our oblique forest setting.

In the spirit of Guo et al. [43], we now show one of our main results that the consistent regression esimate results of Athey et al. [24] extend to classification and so oblique random forests produce consistent estimation rules $\{p_n(y \mid x)\}_{n=1}^{\infty}$ of the posterior $P(Y = y \mid X = x)$ which we denote as $p(y \mid x)$. It is important to note that the gradient-based splitting method outlined in Section 2.3 of Athey et al. [24] is equivalent to the CART [6] method and the gini-impurity method in the binary classification case. To estimate the posterior probability in a multiclass setting, we adopt the one-vs-all approach [44].

Theorem 1. *Under Assumptions 1-3, the posterior probability estimate from a oblique random forest built to Specifications 1-2 is consistent.*

Proof. To show that the posterior probability estimates are consistent, we need to show that $p_n(y \mid x) \xrightarrow{P} p(y \mid x)$ as $n \to \infty$. 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 et al. [24], 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 (GRFs) [24], a random forest built according to the following Specification 1 and solving an estimation problem satisfying the following assumptions 1A-6A yields a consistent estimator $p_n(y \mid x)$ for $p(y \mid x)$. We list these assumption and verify that they hold for posterior probability estimates.

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

This holds per Assumption 1 from Section 4.1.

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, $g : \{Y\} \to \mathrm{R}$ is a univariate summary of Y, and $\xi_{\mu(x)} : \mathrm{R} \to \mathrm{R}$ 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 $\hat{\mu(x)}$ that at least approximately solves the estimating equation

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

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

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

As we have just verified that Theorem 3 from Athey et al. [24] gives us consistent posterior probability estimates and by Lemma 1 this consistency results holds for oblique forests additionally satisfying Specification 2, our Theorem 1 follows. ∎

Proof of Corollary 1: Consistent classification rule Theorem 1 established consistency for each posterior probability estimate $p_n(y \mid x)$. We now proceed to show consistency for the classification rule $g_n(x) = \text{argmax}_y\, p_n(y \mid x)$. As before, define $p(y \mid x) := P(Y = y|X = x)$

Lemma 2. *Let $x \in \mathcal{X}$ with true, but unknown, unique maximum $y^* := \text{argmax}_y\, p(y \mid x)$, and define the finite sample estimate $\hat{y} := \text{argmax}_y\, p_n(y \mid x)$. If $p_n(y \mid x)$ 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 $p_n(y) := p_n(y \mid x)$. Then it follows that

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

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

$$\begin{aligned}
&\sum_{y \neq y^*} P[(p_n(y) - p_n(y^*)) - (p(y) - p(y^*)) > \varepsilon_y] \\
&\leq \sum_{y \neq y^*} P\left[\left|(p_n(y) - p_n(y^*)) - (p(y) - p(y^*))\right| > \varepsilon_y\right]
\end{aligned}$$

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

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

Corollary 3. *Under Assumptions 1-3, the classification rule from a oblique random forest built to Specifications 1-2 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 due to a chance guess between them.

Otherwise, for any $\varepsilon > 0$, by the law of total probabilities,

$$\begin{aligned} P\left[|L_n - L^*| > \varepsilon\right] &= P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon\right] \\ &= P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} = y^*\right] \times P\left[\hat{y} = y^*\right] + \\ &\quad P\left[|p_n(\hat{y} \mid x) - p(y^* \mid x)| > \varepsilon \mid \hat{y} \neq y^*\right] \times P\left[\hat{y} \neq y^*\right]. \end{aligned}$$

In the case that $\hat{y} = y^*$, by Lemma 1 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 2 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,

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

Appendix B. Pseudocode.

Algorithm 1 Learning a Manifold Oblique decision tree, modified from Tomita et al. [8].

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 `MORF` decision tree T

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

Algorithm 2 As in Tomita et al. [8]. 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 $\mathcal{Y} = \{1, \ldots, K\}$
Output: (1) dimension j^*, (2) split value τ^*

function $(j^*, \tau^*) =$ FINDBESTSPLIT$(\mathbf{X}, \mathbf{y})$
  for $j = 1, \ldots, p$ **do**
    Let $\mathbf{x}^{(j)} = (x_1^{(j)}, \ldots, x_n^{(j)})$ be the jth row of $\mathbf{X}$.
    $\{m_i^j\}_{i \in [n]} =$ sort$(\mathbf{x}^{(j)})$ ▷ m_i^j is the index of the i^{th} smallest value in $\mathbf{x}^{(j)}$
    $t = 0$ ▷ initialize split to the left of all observations
    $n' = 0$ ▷ number of observations left of the current split
    $n'' = n$ ▷ number of observations right of the current split
    if (task is classification) **then**
      for $k = 1, \ldots, K$ **do**
        $n_k = \sum_{i=1}^{n} I[y_i = k]$ ▷ total number of observations in class k
        $n'_k = 0$ ▷ number of observations in class k left of the current split
        $n''_k = n_k$ ▷ number of observations in class k right of the current split
      end for
    end if
    for $t = 1, \ldots, n-1$ **do** ▷ assess split location, moving right one at a time
      $(\{(n'_k, n''_k)\}, n', n'', y_{m_t^j}) =$ increment$(\{(n'_k, n''_k)\}, n', n'', y_{m_t^j})$
      $Q^{(j,t)} =$ score$(\{(n'_k, n''_k)\}, n', n'')$ ▷ measure of split quality
    end for
  end for
  $(j^*, t^*) = \underset{j,t}{\operatorname{argmax}}\, Q^{(j,t)}$
  for $i = 0, 1$ **do** $c_i = m_{t^*+i}^{j^*}$ **end for**
  $\tau^* = \frac{1}{2}(x_{c_0}^{(j^*)} + x_{c_1}^{(j^*)})$ ▷ compute the actual split location from the index j^*
  return (j^*, τ^*)
end function

Appendix C. Hyperparameters.

Table 1: ConvNet hyperparameters for each experiment.

Experiment	Classifier	Architecture Sequence
Circle	ConvNet	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	ConvNet	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	ConvNet	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)
Experiments D, E	ConvNet	Conv2d(32, window=2, stride=1) MaxPool2d(window=2, stride=2) Conv2d(32, window=5, stride=1) MaxPool2d(window=2, stride=2) Conv2d(64, window=5, stride=1) MaxPool2d(window=2, stride=2) Linear(64), Linear(n_classes)
MNIST	ConvNet	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)
Surgical Outcome	ConvNet	Conv2d(32, window=2, stride=1) MaxPool2d(window=2, stride=2) Conv2d(32, window=5, stride=1) MaxPool2d(window=2, stride=2) Conv2d(64, window=5, stride=1) MaxPool2d(window=2, stride=2) Linear(64), Linear(n_classes)
Predicting Movement	ConvNet	Conv2d(32, window=3, stride=1) MaxPool2d(window=2, stride=2) Conv2d(64, window=3, stride=1) MaxPool2d(window=2, stride=2) Conv2d(64, window=3, stride=1) MaxPool2d(window=2, stride=2) Linear(64), Linear(n_classes)

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

Experiment	Classifier	Hyperparameters
All	Lin. SVM	C=1, penalty="l2";kernel="linear";loss="squared_hinge"
All	Log. Reg	C=1, penalty="l2"
All	MLP	activation="relu"; alpha=0.0001; hidden_layer_sizes=(100,); solver="adam"
All	RF	n_trees=500, max_features='sqrt'
All	XGB	n_boosting_rounds = 10; learning_rate=0.3; max_depth=6; subsample=1; tree_method='auto' (all default)
All	SPORF	n_trees=500, max_features='sqrt'
All	SVM	C=1; gamma=1/(n_features*Var(X)); kernel="rbf"
All	kNN	n_neighbors=5; p=2
Circle	MORF	n_trees=500, max_features=0.5; patch_height_max=1; patch_height_min=1; patch_width_max=12; patch_width_min=3
H/V Bars	MORF	n_trees=500, max_features='sqrt'; patch_height_max=2; patch_height_min=2; patch_width_max=9; patch_width_min=2
Impulse	MORF	n_trees=500, max_features=0.3; patch_height_max=1; patch_height_min=1; patch_width_max=12; patch_width_min=2
Experiment D	MORF	n_trees=500, max_features='sqrt'; patch_height_max=2; patch_height_min=2; patch_width_max=10; patch_width_min=5;
Experiment E	MORF	n_trees=500, max_features='sqrt'; patch_height_max=2; patch_height_min=1; patch_width_max=20; patch_width_min=5;
MNIST	MORF	n_trees=500, max_features='sqrt' patch_height_max=2; patch_height_min=2; patch_width_max=5; patch_width_min=2;
Surgical Outcome	MORF	n_trees=500, max_features='sqrt'; patch_height_max=sqrt(height); patch_height_min=1; patch_width_max=sqrt(width); patch_width_min=1;