Tree Space Prototypes: Another Look at Making Tree Ensembles Interpretable

Sarah Tan, Matvey Soloviev, Giles Hooker, Martin T. Wells
Cornell University
{ht395, ms2837, gjh27, mtw1}@cornell.edu

Abstract

Ensembles of decision trees are known to perform well on many problems, but are not interpretable. In contrast to existing explanations of tree ensembles that explain relationships between features and predictions, we propose an alternative approach to interpreting tree ensembles by surfacing representative points for each class, in which we explain a prediction by presenting points with similar predictions – prototypes. We introduce a new distance for Gradient Boosted Tree models, and propose new prototype selection methods with theoretical guarantees, with the flexibility to choose a different number of prototypes in each class. We demonstrate our methods on random forests and gradient boosted trees, showing that our found prototypes perform as well as or even better than the original tree ensemble when used as a nearest-prototype classifier. We also present a use case of debugging dataset errors using our proposed methods.

Introduction

Ensembles of decision trees such as random forests (Breiman 2001) and boosted trees (Friedman 2001) have been shown to perform well across a variety of problems (Caruana and Niculescu-Mizil 2006). However, while their decision tree components may be considered interpretable (Freitas 2014), this is no longer true for ensembles with hundreds or thousands of trees. Current attempts to interpret tree ensembles include seeking one tree that best represents the ensemble (Hara and Hayashi 2018; Zhou, Zhou, and Hooker 2018), model-agnostic explanations not exclusive to tree ensembles (Ribeiro, Singh, and Guestrin 2016), feature importance (Ishwaran), partial dependence plots (Friedman 2001), etc. However, many of these describe how features affect predictions, and their complexity increases with the number of features.

Prototypes are representative points that provide a condensed view of a data set (Hart 1968; Bien and Tibshirani 2011). The value of prototypes for case-based reasoning (Richter and Aamodt 2005) has been discussed in studies of human decision making (Kim, Rudin, and Shah 2014). Prototypes have also been used to summarize large data sets (Mirzasoleiman et al. 2013) when not all points can be inspected. In this paper, we propose an alternative to feature-based explanations for tree ensembles. Rather than explaining how features affect a prediction, we propose to explain a prediction by presenting points with similar predictions.

A key question is how to define similarity. Unsupervised distances such as Euclidean distance on feature values do not capture relationships between features and labels, whether actual or predicted. Instead, we need a distance that takes into account: (1) the predictions made by the tree ensemble; (2) how the predictions came about (i.e. how the tree ensemble used the features to arrive at the predictions). We take inspiration from the random forest (RF) proximity matrix (Breiman and Cutler 2002). This n-by-n matrix (n is the number of points) describes the proportion of trees in the forest where a pair of points end up in the same leaf (and hence receive the same prediction). Moreover, since this pair of points followed the same path down the tree, they necessarily have the same values for features deemed important by the tree to make predictions. Hence, RF proximity characterizes the similarity between points, *as "seen" from the point of view of the tree ensemble*.

It turns out that this notion of proximity is especially suitable for identifying prototypes and using them as a basis for an interpretable classification mechanism. We will present methods to find prototypes which can be used with a nearest-prototype classifier under the RF proximity described above, which sometimes even exceeds the accuracy of the original tree ensemble. Furthermore, we introduce a generalization of the RF proximity function to *gradient boosted trees* (GBTs), in which the individual trees can have different contributions to the predictor.

We further introduce a new class-aware prototype selection method with the flexibility to choose a variable number of prototypes per class, a generalization of methods that choose the same number of prototypes per class without adapting to the structure of the distance or class imbalance in the data set. Here, we are motivated by the belief that a user with a budget of k prototypes that she can inspect, say a physician trying to diagnose a rare disease where patients with the disease are very diverse, is better served by a flexible allocation of k across the different classes. In our experiments, this flexibility returned prototypes with greater diversity.

To summarize, our contributions are: (1) An alternative approach to interpreting tree ensembles by surfacing representative points for each class; (2) a new distance formulation for GBT models; and (3) a new prototype selection method with theoretical guarantees, which has the flexibility

to choose a different number of prototypes in each class.

Related Work

Tree ensemble distance. Breiman and Cutler defined RF proximity in the documentation accompanying their software (Breiman and Cutler 2002). It is common to set distance as $1 - \text{proximity}$ (Zhao et al. 2016; Xiong et al. 2012; Shi and Horvath 2006), as we do in this paper. RF proximity has found a variety of applications, including clustering (Shi and Horvath 2006), outlier detection (Zhou et al. 2015), imputation (Stekhoven 2015), etc., however less is known of its theoretical properties. The connection between random forests and kernel methods has been pointed out (Lin and Jeon 2006; Scornet 2016) and proximity itself can be expressed as a kernel (Louppe 2014). While we were inspired by RF distance, to the best of our knowledge, our paper presents the first proposal for GBT distance and the first method to seek prototypes for GBT models.

Not many RF implementations provide prototypes; the exception is the R `randomForest` package (Liaw and Wiener 2002) and RAFT, a visualization tool (Breiman and Cutler 2002). In their documentation, they described a heuristic prototype-finding procedure that is partially implemented in the `randomForest` package. The procedure currently generates a single new point not from the data set, a distinct goal from ours, which is to select a subset of representative points from existing points.

Prototype selection. There is a long line of literature on prototype selection methods, also known as instance reduction, data summarization, exemplar extraction, etc. We point the reader to the review by (Garcia et al. 2011), who suggest that prototype selection methods can be grouped into three categories: condensation (Hart 1968), edition (Wilson 1972), or hybrid methods that remove both noisy and redundant points from the prototype selection set. We briefly mention a few methods: k-medoids clustering is a classic problem for which different algorithms have been proposed, such as PAM (Kaufman and Rousseeuw 1987) and greedy submodular approaches (Lin and Bilmes 2011; Gomes and Krause 2010), which we compare against and extend by adding the flexibility to choose varying numbers of prototypes by class. Kim et al. used a similar greedy submodular approach with maximum mean discrepancy objective to select prototypes and criticisms (Kim, Khanna, and Koyejo 2016), to summarize a dataset. We do not compare against set cover methods (Bien and Tibshirani 2011) as they tend to select significantly more prototypes than k-medoids (Bien and Tibshirani 2011) to achieve their objective of maximal coverage, at the cost of interpretability.

Point-based explanations. Besides feature-based explanations, point-based explanations have been proposed to explain model predictions. Examples include counterfactual explanations that determine the changes necessary to flip a point's prediction (Wachter, Mittelstadt, and Russell 2017), designing models that automatically provide prototypes (Kim, Rudin, and Shah 2014; Li et al. 2018), and identifying points most "influential" for a prediction (Koh and Liang 2017; Yeh et al. 2018; Khanna et al. 2019). There is a subtle distinction between prototype selection methods and influential point methods, as points that best represent a class (prototypes) may not be the most influential. Moreover, since trees are not differentiable except trivially within each node, influential point methods that typically take gradients of loss functions are not easily applicable to tree ensembles. Hence we do not compare against them, but mention them for completeness.

Background and Notation

In this section, we provide some technical background and notation for later sections. We assume that we are given a training set of points S, from which we will learn k prototypes from q classes to better understand a classifier $c : S \to [q]$. To learn these prototypes, we will introduce a number of distance functions $d : S^2 \to \mathbb{R}^+$ capturing how the classifier represents differences between points and classes.

Tree Ensembles

Let t be the number of trees in the RF model. The ith tree ($i \in [t]$) has τ_i leaves, each of which represents a region $R_{j,i}$ ($j \in [\tau_i]$) of feature space. Each individual tree induces a classifier $c_i^{\text{Tree}}(s) = \sum_{j=1}^{\tau_i} \alpha_{j,i} \mathbb{I}(s \in R_{j,i})$, where $\alpha_{j,i}$ is the predicted value in the jth leaf of the ith tree (for binary classification, this is just the proportion of points in that leaf with label 1) and $\mathbb{I}$ is the indicator function. The RF classifier is the average of this, taken over all trees:

$$c^{\text{RF}}(s) = \frac{1}{t} \sum_{i=1}^{t} c_i^{\text{Tree}}.$$

Next, we consider the GBT classifier, which is learned iteratively:

$$c_i^{\text{GBT}}(s) = c_{i-1}^{\text{GBT}}(s) + \gamma_i c_i^{\text{Tree}}(s)$$

where the initial value c_0^{GBT} is initialized, depending on implementation, as zero, the fraction of elements of S with label 1 in the case of binary classification, etc. γ_i is a step size, typically found using line-search. The GBT classifier then is the one that incorporates all t trees:

$$c^{\text{GBT}}(s) = c_t^{\text{GBT}}(s).$$

RF Distance. Tree structure lends itself to a natural definition of distance between points as "seen" by a tree, if we consider a pair of points that travel down the same path in a tree and end up in the same leaf closer than another pair of points that do not end up in the same leaf.

Definition 1. *(Breiman and Cutler 2002) The RF proximity of a pair of points is an unweighted average of the number of trees in the RF model in which the points end up in the same leaf:*

$$\begin{aligned} &\mathsf{proximity}^{\text{RF}}(s, s') \\ = \quad &\frac{1}{t} \sum_{i=1}^{t} \sum_{j=1}^{\tau_i} \mathbb{I}(s \in R_{j,i}) \mathbb{I}(s' \in R_{j,i}). \end{aligned} \tag{1}$$

The RF distance between a pair of points is then:

$$d^{\text{RF}}(s, s') = 1 - \mathsf{proximity}^{\text{RF}}(s, s').$$

Since the regions $\{R_{j,i}\}_{j=1}^{T_i}$ partition the feature space, each point $s \in S$ can be in at most one region, and so the inner sum in Equation (1) takes on value 0 or 1 for each tree. Thus the proximity, as a convex combination of these, lies between 0 and 1, and so does the distance function. It is easily confirmed that the proximity of a point to itself is 1, and hence $d(s, s) = 0$, but it should be noted that d is not in general a metric, but a pseudosemimetric as it does not satisfy the triangle inequality – as noted by (Xiong et al. 2012). This it not uncommon in the metric learning literature, and in fact, no locally adaptive distance (distance that varies across feature space (Lin and Jeon 2006)) can satisfy the triangle inequality (Xiong et al. 2012). Later, we will adapt RF distance to construct a distance function for GBTs.

The k-Medoids Problem

The goal of the *k-medoids clustering problem* is to find a subset $M \subseteq S$ of medoids, $|M| = k$ such that the sum distance from each object to the nearest medoid is minimized. In the prototype selection literature, medoids have been taken as prototypes (Bien and Tibshirani 2011), hence our interest in them. Formally, the k-medoids algorithm aims to find the set $M \subseteq S$ that minimizes the objective function

$$f(M) = \sum_{s \in S} \min_{m \in M} d(s, m). \tag{2}$$

This problem is known to be NP-hard (Papadimitriou 1981). However, (Gomes and Krause 2010) present a greedy algorithm that starts with an empty set and repeatedly adds the single object $s \in S \setminus M$ that increases the value of a related function by the most, which they show produces a reasonable approximation in polynomial time.

If the points are labelled by a classifier, it is natural to only consider, for each point, medoids that belong to the same class. Thus, we define the *q-classwise k-medoids problem* as finding the subset $M \subseteq S$ of k medoids such that the sum distance from each object to the nearest medoid *belonging to the same class* is minimized, i.e. that minimizes

$$f(M) = \sum_{s \in S} \min_{m \in M : c(m) = c(s)} d(s, m). \tag{3}$$

Even in the presence of multiple classes, it is possible to use the single-class algorithm of (Gomes and Krause 2010) by applying it separately to every class in turn to generate k_1, ..., k_q prototypes for each class ($\sum_i k_i = k$). However, it is not clear what the right choice of k_i for each class is, and one could easily lose accuracy by overprovisioning one compact class that would be adequately covered by a small number of prototypes while not having sufficiently many prototypes for another class whose points are spread into many clusters. With the naive choice that $k_1 = \ldots = k_q = k/q$, we call this the uniform greedy submodular (SM-U) prototype selection method, and use it as one of our baselines.

However, it turns out that an analysis similar to that for the single-class case can also be applied directly to the q-classwise objective function. Based on this, we will introduce a greedy algorithm that operates on all classes in the q-classwise k-medoids problem simultaneously. Since this algorithm in effect chooses the class where adding another prototype yields the largest improvement, we will call it *adaptive* in contrast with the uniform algorithm.

Submodular optimization

Optimization problems such as (3) are often approached using approximation algorithms that are guaranteed to find solutions within some factor of the optimum. One approach to the k-medoids problem (Gomes and Krause 2010; Mirzasoleiman et al. 2013) is to identify a related positive monotone submodular function and use a greedy search for a good element of its domain, which is then guaranteed to be within a factor of $(1 - 1/e)$ of the optimum for that function, where e is the Euler constant. We quickly review the relevant result.

Definition 2. *A function $f : \mathcal{P}(S) \to \mathbb{R}$ that maps subsets of S to reals is* monotone *if $f(X) \leq f(Y)$ whenever $X \subseteq Y$. It is* submodular *if whenever $X \subseteq Y$, adding a particular element $s \in S$ to Y will not be more useful than adding it to X:*

$$f(Y \cup \{s\}) - f(Y) \leq f(X \cup \{s\}) - f(X).$$

Proposition 1. *(Nemhauser, Wolsey, and Fisher 1978) Suppose $f : \mathcal{P}(S) \to \mathbb{R}^+$ is a non-negative monotone submodular function. Let $T_0 = \varnothing$ and*

$$T_i = T_{i-1} \cup \arg\max_{s \in S} f(T_{i-1} \cup \{s\})$$

be the result of greedily maximizing f for i steps. Also, let

$$T_i^* = \arg\max_{T \subset S : |T| = k} f(T)$$

be the set of size i that maximizes f. Then

$$f(T_i) \geq (1 - 1/e) f(T_i^*).$$

We will use this proposition to define a greedy algorithm on an appropriate submodular function to find an approximate solution to the q-classwise k-medoids problem (3).

Method

Our goal is to find prototypes for tree ensembles, as an alternative approach to interpreting these models. In this section, we describe two methodological contributions of this paper: defining a distance function for GBT models, and new prototype selection methods that choose a variable number of prototypes based on which class could benefit the most from another prototype.

Constructing a Distance Function for GBT

Unlike the RF distance function in Definition 1, in GBT each tree is no longer generated by an identical process. Hence, each tree can no longer be weighted equally. We propose to weigh the contribution of each tree to the proximity function by the size of its contribution to the overall prediction. Here we measure size by the variance among the predictions made by $c_i^{\text{Tree}}(s)$. γ provides a correction to account for the quadratic approximation to the loss that is used by gradient boosting. We thus arrive at the following definition:

Definition 3. *The GBT proximity of a pair of points is a weighted average of the number of trees in the GBT model in which the points end up in the same leaf:*

$$\begin{aligned}&\mathsf{proximity}^{\mathrm{GBT}}(s, s') \\ &= \sum_{i=1}^{t}\sum_{j=1}^{\tau_i} \frac{w_i}{\sum_{i=1}^{t} w_i}\mathbb{I}(s \in R_{j,i})\mathbb{I}(s' \in R_{j,i}),\end{aligned}$$

where the ith tree's weight w_i is

$$w_i = \gamma_i^2 \cdot \mathrm{Var}\{c_i^{\mathrm{Tree}}(s) : s \in S\}$$

The GBT distance between a pair of points is then:

$$d^{\mathrm{GBT}}(s, s') = 1 - \mathsf{proximity}^{\mathrm{GBT}}(s, s').$$

While other choices of measures of the size of predictions in each tree can be made, e.g. using the L^1 norm, we do not pursue them here and focus instead on demonstrating the properties of the chosen distance function.

Adaptive Prototype Selection Methods

We now introduce two new prototype selection methods that exploit the submodular approximation guarantees of Proposition 1, and one that tries to directly optimize for (train or validation) accuracy.

Our goal is to find a good approximately optimal solution for the q-classwise k-medoids problem (3). We will achieve this by using a greedy algorithm on an appropriate non-negative, monotone, submodular function (Prop. 1). However, the function (3) itself is not monotone submodular: in fact, adding more prototypes to M *decreases* the value of $f(M)$. This can be avoided by negating f, but then the function will take non-positive values. Therefore, adapting an idea of (Gomes and Krause 2010), we will define a related function g as

$$g(M) = f(P) - f(P \cup M), \tag{4}$$

where P is an appropriately chosen set of *phantom exemplars*, one from each class. The resulting algorithm is Algorithm 1.

Algorithm 1: Adaptive greedy submodular prototype selection (SM-A)

Input: Set of points S, distance function $d : S^2 \to [0, 1]$, class assignment $c : S \to [q]$
Output: Set of prototypes M, $|M| = k$

1 Create set of phantom exemplars $P = \{p_1, \ldots, p_q\}$ and set $d(p_i, s) = d(s, p_i) = 1$ for all s
2 $M \leftarrow \varnothing$
3 **for** *i=1* **to** *k* **do**
4 $\quad s^* \leftarrow \arg\max_{s \in S} [f(P) - f(P \cup M \cup \{s\})]$
5 $\quad M \leftarrow M \cup \{s^*\}$

We want to derive a guarantee on the approximation of this algorithm using Proposition 1; to that end, we first need to show that g satisfies the necessary conditions.

Lemma 1. *The objective function (4) is non-negative, monotone and submodular.*

Proof. See supplementary material. □

By selecting the set of phantom exemplars P in such a fashion that $d(p, s) \geq d(s', s)$ for all $p \in P$ and $s, s' \in S$, we ensure that $f(T \cup P) = f(T)$ for all nonempty sets $T \subseteq S$. Hence, the set T_i^* that maximizes g among all sets of size i also minimizes f among all such sets.

Let T_i be the result of running the greedy maximization algorithm on (4) for i steps, and f be the original objective function (3). Then by Prop. 1 and choice of P,

$$f(T_i) \leq f(P) + (1 - 1/e)(f(T_i^*) - f(P)),$$

i.e. the approximation T_i takes us $1 - 1/e$ of the way from $f(P)$ to the optimum. Crucially, this means that the approximation guarantee depends on $f(P)$, i.e. how good the phantom exemplars alone would be as a solution to the k-medoids problem.

We also consider a variant of this algorithm that we call weighted adaptive greedy submodular (SM-WA), in which each class is weighed differently: line 4 is replaced by

$$s^* \leftarrow \arg\max_{s \in S} \frac{1}{|C(s)|} \left[f(P) - f(P \cup M \cup \{s\})\right],$$

where $C(S)$ denotes all points in S that are in the same class as s. It is easily verified that this objective function is also submodular.

Supervised Greedy Prototype Selection

Instead of optimizing the k-medoids value function f of equation (3), we can instead directly pick prototypes, in a greedy fashion, that yield the best (train or validation set) improvement in classification performance. The resulting method, which we call supervised greedy (SG), beats the unsupervised k-medoids approaches in terms of accuracy in several cases (cf. Table 1), but we do not know of any theoretical guarantees that it satisfies, as these accuracy metrics are not submodular. This is nearly identical to Algorithm 1, except that line 1 is unnecessary and we replace line 4 with

$$s^* \leftarrow \arg\max_{s \in S} \left[\mathsf{accuracy}(S, M \cup \{s\})\right],$$

where accuracy denotes the accuracy metric used for evaluation.

Experiments and Analysis

We present experiments that demonstrate the value of the tree ensemble distances and prototype selection methods.

Datasets and features. We use six classification datasets. Four are tabular data, and two are images: CALTECH-256, where we select two classes: guitar and mandolin, and MNIST, where we sampled digits 4 and 9, two commonly confused classes (Hadsell, Chopra, and LeCun 2006). More details on how we sampled MNIST are in the supplementary material. For MNIST, we use the raw pixel values as our feature representation. For CALTECH-256, we extracted deep features using a ResNet-50 model pre-trained on ImageNet.

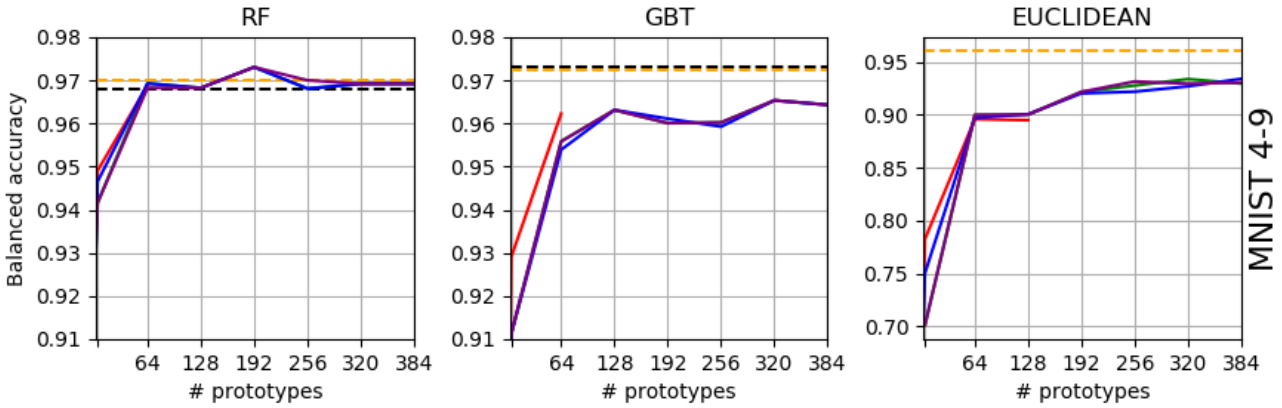

Figure 1: Test-set balanced accuracy as a function of k, the number of prototypes. The black dashed line represents the original model, and the dashed orange line represents the 1-NN baseline using all training points (i.e. treating all training points as prototypes). Different prototype selection methods were used: supervised greedy (SG), uniform greedy submodular (SM-U), adaptive greedy submodular (SM-A), weighted adaptive greedy submodular (SM-WA). Note the different y-axis scales. SG ends abruptly because it does not add prototypes if none that increase train or validation accuracy can be found, unlike other methods that add prototypes (with diminishing returns, as the objective is submodular) as long as requested.

Training procedure. We trained RF models with 1000 trees using Python's `scikit-learn` package. The maximum tree depth was not restricted. For GBT, we modified `scikit-learn` to train GBT models with one gamma multiplier per tree. We cross-validated the number of trees (up to 200), maximum depth (3 to 5), and other parameters as described in the supplementary material.

Metric. We use balanced accuracy as our primary performance metric, since some of our data sets are imbalanced, and also count the number of prototypes selected as a proxy for interpretability (the number of points a user has to inspect). We do not use ranking metrics such as AUC because nearest-neighbor classifiers do not output scores.

Evaluating Tree Ensemble Distances and Prototypes: Quantitative Evaluation

One way to validate that the selected prototypes are reasonable is to use them in in a nearest-prototype classifier (Bien and Tibshirani 2011; Kim, Khanna, and Koyejo 2016). This evaluation is in line with recent ideas on evaluating explanations by checking their accuracy on independent test-data (Ribeiro, Singh, and Guestrin 2016).

We use a nearest-prototype classifier to classify test-set data. Figure 1 illustrates test-set balanced accuracy as a function of the number of prototypes on MNIST with digits 4 and 9. First, we check if **tree ensemble distance improves over Euclidean distance**. We examine the plots for RF and GBT tree ensemble distances (left and center) compared to Euclidean distance (right) in Figure 1. A fair comparison holds the prototype selection method constant (i.e. look at the same colored line across left to right). RF and GBT distances clearly outperform Euclidean distance. This is unsurprising, as the tree ensemble distances are supervised whereas Euclidean distance is not. We defer an analysis of the difference between the RF and GBT distances to the next section.

Next, we demonstrate the value of prototype selection by comparing all **selection methods to 1-nearest-neighbors (1-NN) that does not select prototypes** and instead uses all (training set) points as prototypes. A fair comparison holds the distance constant (i.e. look at the different colored lines in the same plot). With 16 prototypes, the nearest-prototype classifier using RF distance is as accurate as the original RF model (black dashed line), and at 24 prototypes onwards starts to approach the accuracy of 1-NN. Note that in the limit (k=size of training set) with the maximum number of prototypes, prototype selection is exactly 1-NN.

Finally, we see how **our proposed prototype selection methods compare to other methods**. Moreover, we observe that the accuracy of the prototype selection method varies according to k, the number of prototypes. This suggests that k should be tuned separately for each prototype selection method, in line with (Bien and Tibshirani 2011) where the optimal number of prototypes was found to vary significantly by selection method.

Table 1 presents test-set balanced accuracy at the optimal number of prototypes, k, tuned separately for each prototype selection method. We make several points: (1) For all datasets, at least one prototype selection method outperformed 1-NN, suggesting the value of prototype selection not just for data condensation and interpretability (reducing the number of points that need to be shown to a user), but also classification accuracy. (2) Again we see that tree ensemble distances outperform Euclidean distance. (3) SM-WA is competitive against SM-U. (4) Despite the lack of theoretical guarantees, SG had clear advantages on a number of datasets with high achieving high accuracy. (5) SG tends to select less prototypes than other selection methods.

Evaluating Prototypes: Qualitative Evaluation

Figure 3 displays RF distances embedded in a two-dimensional space using t-sne (Maaten and Hinton 2008), and prototypes found by SM-A (left) compared to the pro-

Model		Breastcancer	Diabetes	T-COMPAS	RHC	MNIST 4-9	CALTECH256 G-M
RF	Original	**0.92**	0.72	0.56	0.68	0.97	0.81
	1-NN	0.91 (341)	0.67 (460)	0.57 (600)	0.65 (3441)	0.97 (3000)	0.78 (129)
	SM-U	**0.92** (12)	0.76 (5)	0.61 (10)	**0.69** (11)	0.97 (187)	0.83 (16)
	SG	0.90 (**4**)	**0.77** (5)	**0.68** (**5**)	0.66 (12)	0.97 (**18**)	0.83 (**3**)
	SM-A	**0.92** (11)	**0.77** (**4**)	0.57 (15)	0.68 (**9**)	0.97 (163)	0.79 (15)
	SM-WA	**0.92** (15)	**0.77** (6)	0.60 (13)	**0.69** (13)	0.97 (243)	**0.86** (18)
GBT	Original	0.94	0.69	0.55	0.70	0.97	0.84
	1-NN	0.92 (341)	0.70 (460)	0.58 (600)	0.65 (3441)	0.97 (3000)	0.84 (129)
	SM-U	0.92 (21)	0.70 (12)	0.53 (26)	0.65 (62)	0.96 (249)	0.84 (11)
	SG	**0.95** (**3**)	**0.78** (**4**)	**0.67** (5)	0.69 (15)	0.96 (**23**)	0.82 (**2**)
	SM-A	0.92 (22)	0.69 (12)	0.57 (**4**)	0.65 (**4**)	0.96 (247)	0.84 (11)
	SM-WA	0.92 (20)	0.69 (12)	0.56 (27)	0.65 (**4**)	0.96 (261)	0.84 (11)
EUCLIDEAN	1-NN	**0.91** (341)	0.68 (460)	0.53 (600)	0.59 (3441)	0.96 (3000)	0.81 (129)
	SM-U	0.89 (19)	0.71 (26)	0.51 (62)	0.61 (40)	0.93 (313)	0.82 (**6**)
	SG	0.87 (**3**)	**0.75** (4)	**0.58** (**22**)	**0.67** (19)	0.90 (**65**)	0.74 (9)
	SM-A	0.88 (18)	0.68 (**3**)	0.52 (51)	0.60 (**8**)	0.93 (377)	**0.88** (**6**)
	SM-WA	**0.91** (15)	0.73 (5)	0.51 (61)	0.61 (33)	0.93 (380)	**0.88** (**6**)

Table 1: Test-set balanced accuracy for different prototype selection methods at optimal number of prototypes (according to validation accuracy) : supervised greedy (SG), adaptive greedy submodular (SM-A), weighted adaptive greedy submodular (SM-WA), against uniform greedy submodular (SM-U) and 1-NN baselines. Best results for each dataset and distance in **bold**.

totypes found by SG (right) on MNIST. We see that the prototypes selected by SM-A cover the space of points well, which is not the case for SG. We confirm this by computing the distance from each point to its nearest prototype (SG: mean 0.61, sd 0.23; SM-A: mean 0.23, sd 0.26). Instead, the prototypes selected by SG are on the border between the two classes, and it is common to see these prototypes alternating (i.e. a 9 prototype followed by a nearby point of class 4 being selected as a prototype). This once again highlights that SG selects prototypes to maximize accuracy.

Figure A2 in the supplementary material presents the prototypes found by the different methods when using RF distances on the CALTECH256 G-M data. First, similar to what we found for MNIST, even if we requested 16 prototypes, SG found that only 3 prototypes were necessary. These include canonical views of popular guitar models and a frontal picture of a mandolin. These do not cover all representative guitars or mandolins, but are good enough to correctly classify most images. The submodular methods select prototypes that ensure coverage, surfacing different poses and shapes that are in many cases different from the archetypal ones. Both adaptive methods – SM-A and SM-WA – tend to select more mandolins than guitars, including close-ups and people, not found in the guitar prototypes. Hence, different prototype methods select different types of prototypes depending on whether the objective function is optimizing for coverage or accuracy. Moreover, the flexibility to choose a different number of prototypes by class was utilized.

Evaluating Tree Ensemble Distance: Fixing Mislabeled Points Use Case

We now demonstrate how the proposed methods can assist with user tasks, such as correcting mislabeled points. We wish to present a reasonable number of points to the user, in some meaningful ordering, for the user to correct any wrong labels. This experiment is similar to that ran by (Koh and Liang 2017; Yeh et al. 2018).

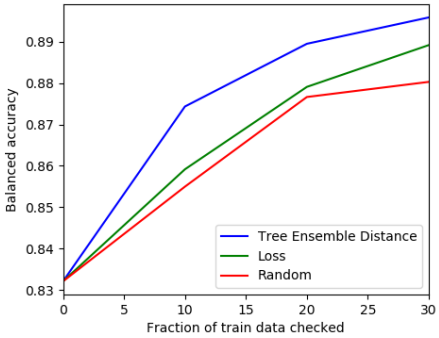

Figure 2: Performance of different algorithms, including one based on tree ensemble distance, to select points from MNIST 4-9 with corrupted labels for a simulated user to inspect. The simulated user flips the labels of points checked, and a model is retrained on the corrected data.

We corrupt the MNIST 4-9 data set by flipping the labels of a third (33%) of points, and use RF distance to construct a ranking of points, which we compare to two baselines: (1) random ranking; (2) ranking based on loss of training points. The RF distance ranking is constructed thus: for every training point, compute $k = 10$ nearest neighbors based on RF distance, then compute the proportion of neighbors that share the same label as the point. Note that the loss ranking is strong baseline, as found by (Koh and Liang 2017).

We present a simulated user with a proportion (up to

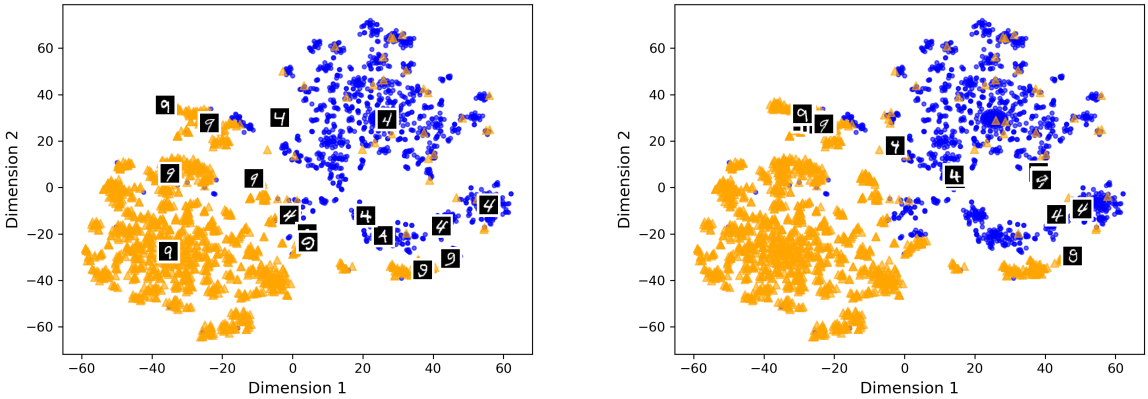

Figure 3: Visualization of distances using t-sne for optimal RF with mean depth 16 on the MNIST (4 vs 9) dataset, using the adaptive greedy submodular (SM-A) prototype selection method (left) and supervised greedy (SG) (right) method. Orange represents the digit 9, blue represents the digit 4. and the black and white images are prototypes.

30%) of training points, as ranked by the different methods, to inspect and correct. Similar to (Koh and Liang 2017; Yeh et al. 2018), the simulated user is an oracle who only corrects points that are flipped, of all the points presented to her. The model is then retrained on the corrected data. We repeat the experiment 20 times, randomizing the points corrupted each time, and average the results. Figure 2 plots the mean test-set performance of the model retrained on simulated user corrected data. With the same interpretability budget (amount of points the simulated user had to inspect), tree ensemble distance based ranking was better at assisting in correcting mislabeled points, generating corrected data that had higher test-set accuracy than other rankings.

Analysis: Tree Ensemble Distance by Tree Depth

We now study the learned tree ensemble distances to gain intuition into their behavior. Figure 4 illustrates the distribution of distances for RF and GBT distances compared to Euclidean distance on one of the datasets, Breastcancer.

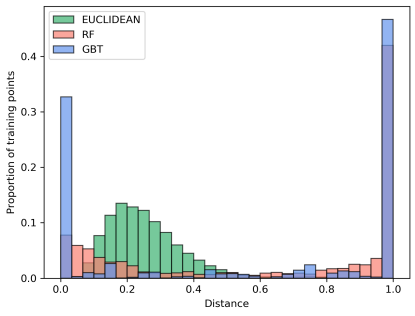

Figure 4: Distribution of RF and GBT distance compared to Euclidean distance on one of the datasets, Breastcancer.

Unlike RF and GBT distances which are derived from supervised models, Euclidean distance is unsupervised, hence has no preference for extreme distances. RF distance is more granular than GBT distance, with less "clumping" and less extreme values (0 or 1 distance). Many default implementations of RF algorithms allow trees to grow to unrestricted depth (Breiman 2001), hence on the same data set, trees in RF models tend to be deeper than trees in GBT models. We confirm this in Table A1 in the supplementary material, which presents statistics of tree depth in RF and GBT models. The shallower the tree, the less leaves, the higher the probability of a pair of points ending up in the same leaf – one possible explanation for the more granular behavior of RF distance compared to GBT.

The larger the data set, the deeper the RF trees (cf. Table A1 in the supplementary material), whereas our GBT trees are limited to depth 3 to 5. Hence, the larger the data set, the more different we expect RF and GBT distances to be. The supplementary material presents further analysis on the degree to which differences between the GBT and RF distances are caused by different tree depth, different weights used in constructing the distance, or that different patterns in the data are being learned by RF compared to GBT models.

Discussion

Of the four prototype selection methods used in this paper, three – supervised greedy, adaptive greedy submodular, weighted adaptive greedy submodular – are able to select different number of prototypes for each class. Unlike uniform greedy submodular, which returns round(k/q) prototypes per class regardless of class imbalance or distance behavior, these three methods pick the next prototype from the class that needs it the most, as measured by the objective function. We find that on the datasets that we investigated, this sometimes leads to a small numeric improvement in test-set performance. A qualitative evaluation of the prototypes generated reveals that adaptive algorithms tend to lead to better coverage in classes which exhibit more diversity which may be beneficial for interpretability. The tree ensemble distances outperformed Euclidean distance, demonstrating the benefit of distances directly derived from the learned tree ensemble models.

References

Bien, J., and Tibshirani, R. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics*.

Breiman, L., and Cutler, A. 2002. Random forests manual. `https://www.stat.berkeley.edu/~breiman/RandomForests`. Accessed July 6, 2019. Year 2002 based on copyright year indicated in the authors' Fortran code.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Caruana, R., and Niculescu-Mizil, A. 2006. An empirical comparison of supervised learning algorithms. In *ICML*.

Freitas, A. A. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations*.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*.

Garcia, S.; Derrac, J.; Cano, J. R.; and Herrera, F. 2011. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *TPAMI*.

Gomes, R., and Krause, A. 2010. Budgeted nonparametric learning from data streams. In *ICML*.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*.

Hara, S., and Hayashi, K. 2018. Making tree ensembles interpretable: A bayesian model selection approach. In *AISTATS*.

Hart, P. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*.

Ishwaran, H. Variable importance in binary regression trees and forests. *Electronic Journal of Statistics*.

Kaufman, L., and Rousseeuw, P. J. 1987. Clustering by means of medoids. In *Statistical Data Analysis Based on the L1 Norm*. Birkhuser Basel.

Khanna, R.; Kim, B.; Ghosh, J.; and Koyejo, O. 2019. Interpreting black box predictions using fisher kernels. In *AISTATS*.

Kim, B.; Khanna, R.; and Koyejo, O. O. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*.

Kim, B.; Rudin, C.; and Shah, J. A. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *NIPS*.

Koh, P. W., and Liang, P. 2017. Understanding black-box predictions via influence functions. In *ICML*.

Li, O.; Liu, H.; Chen, C.; and Rudin, C. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

Liaw, A., and Wiener, M. 2002. Classification and regression by randomforest. *R News*.

Lin, H., and Bilmes, J. 2011. A class of submodular functions for document summarization. In *ACL*.

Lin, Y., and Jeon, Y. 2006. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*.

Louppe, G. 2014. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR*.

Mirzasoleiman, B.; Karbasi, A.; Sarkar, R.; and Krause, A. 2013. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*.

Papadimitriou, C. H. 1981. Worst-case and probabilistic analysis of a geometric location problem. *SIAM Journal on Computing*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the predictions of any classifier. In *KDD*.

Richter, M. M., and Aamodt, A. 2005. Case-based reasoning foundations. *The Knowledge Engineering Review*.

Scornet, E. 2016. Random forests and kernel methods. *IEEE Transactions on Information Theory*.

Shi, T., and Horvath, S. 2006. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*.

Stekhoven, D. J. 2015. missforest: Nonparametric missing value imputation using random forest. *Astrophysics Source Code Library*.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology* 31(2).

Wilson, D. 1972. Asymptotic properties of nearest neighbor rules using edited data sets. *Transactions on Systems, Man and Cybernetics*.

Xiong, C.; Johnson, D.; Xu, R.; and Corso, J. J. 2012. Random forests for metric learning with implicit pairwise position dependence. In *KDD*.

Yeh, C.-K.; Kim, J.; Yen, I. E.-H.; and Ravikumar, P. K. 2018. Representer point selection for explaining deep neural networks. In *NeurIPS*.

Zhao, P.; Su, X.; Ge, T.; and Fan, J. 2016. Propensity score and proximity matching using random forest. *Contemporary Clinical Trials*.

Zhou, Q.-F.; Zhou, H.; Ning, Y.-P.; Yang, F.; and Li, T. 2015. Two approaches for novelty detection using random forest. *Expert Systems with Applications*.

Zhou, Y.; Zhou, Z.; and Hooker, G. 2018. Approximation trees: Statistical stability in model distillation. *arXiv preprint arXiv:1808.07573*.

Supplementary Material for Tree Space Prototypes: Another Look at Making Tree Ensembles Interpretable

Proof of Lemma 1

Lemma 1. *The objective function (4) is non-negative, monotone and submodular.*

Proof (Lemma 1). Observe that whenever $X \subseteq Y$, we have $f(X) \geq f(Y)$, since adding more points to a set can only make the closest point to a given point closer. From this, monotonicity and non-negativity is immediate, since $f(P) \geq f(P \cup M)$.

To establish submodularity, we will show that the function f of (3) satisfies

$$f(Y) - f(Y \cup \{t\}) \leq f(X) - f(X \cup \{t\})$$

whenever $X \subseteq Y \subseteq S$. The inequality of definition 2 then follows for g by plugging into its definition (4).

For any point $s \in S$, define $p_M(s)$ to be the closest point to s in M of the same class, that is,

$$p_M(s) = \underset{m \in M : c(m) = c(s)}{\arg\min} d(s, m).$$

Then we can rewrite $f(M)$ as

$$\sum_{s \in S} d(s, p_M(s)),$$

and it suffices to show that

$$\begin{aligned} & d(s, p_Y(s)) - d(s, p_{Y \cup \{t\}}(s)) \\ \leq \; & d(s, p_X(s)) - d(s, p_{X \cup \{t\}}(s)). \end{aligned}$$

for all $s \in S$. Both sides of this inequality are non-negative (+), since adding points can only shorten the distance to the closest point. Suppose $p_{Y \cup \{t\}}(s) \in Y$. Then it must be equal to $p_Y(s)$, since the closest point is present in Y, and so the first line is 0, and the inequality follows from (+).

Suppose instead $p_{Y \cup \{t\}}(s) \notin Y$. Then it must be t. So $p_{X \cup \{t\}}(s) = t$ as well (since $X \subseteq Y$), and the inequality reduces to $d(s, p_Y(s)) \leq d(s, p_X(s))$. But this is immediate, since $Y \supseteq X$ and adding more points can only shorten the distance to the closest point. □

Training Procedure Details

In this section we give more details on how the RF and GBT models were trained.

RF parameters. We cross-validated the number of features to consider when looking for the best split, out of p features in the data set. We considered $\sqrt{p}, 0.33p, 0.5p, 0.7p$, and 7 features, and take the option that minimizes validation loss.

GBT parameters. We kept only the first t trees (maximum 200) where t was cross-validated to minimize validation loss. We also cross-validated the maximum tree depth (between 3 and 5) and learning rate (0.1 or 0.01).

Analysis: Difference between Tree Ensemble Distances

As pointed out in the main paper, GBT distance can be quite different from RF distance. Figure A1 visualizes RF and GBT distances embedded in a two-dimensional space using t-sne and several MNIST prototypes. While digits 4 and 9 are clearly separable in Figure A1, consistent with the high performance (97% accuracy; cf. Table 1) of the RF and GBT models, the models appear to be learning different representations, with GBT grouping points together in smaller and more compact clusters than RF. Moreover, on this data set, the prototypes selected for RF compared to GBT are different.

A natural next question may be the following: to what degree are differences between GBT and RF distances caused by different tree depth, different weights used in constructing the distance, or that different patterns in the data are being learned by RF compared to GBT models? While the top right corner of Figures A1 visualizes distances from GBT models trained with default settings (short), and the bottom right corner of the same figures visualizes distances from RF models trained with default settings (unrestricted depth), the bottom left corner visualizes RF models trained *to the same depth* as the corresponding default GBT model on that dataset. While the short RF model has smaller and more compact clusters than the default RF model, the RF and GBT models of same depth can still be told apart.

The top left corner visualizes an unweighted distance function derived from the same GBT model as in the top right corner. Note that the visualization in the top right corner is of a weighted distance function. This unweighted GBT distance (top left corner) looks more similar to the default RF model's distance (bottom right corner).

		GBT	RF Depth			
Dataset	n	Depth	Min	Mean	Max	Var
Breastcancer	569	3	2	3.02	4	0.40
Diabetes	768	3	5	7.36	12	1.04
T-COMPAS	1000	4	6	8.70	14	1.18
RHC	5735	3	11	14.7	21	1.41
MNIST 4-9	5000	3	8	10.95	16	1.44
CAL256 G-M	215	2	2	2.51	3	0.50

Table A1: Statistics of RF and GBT models tree depth across different datasets. n is the number of observations in the dataset. All RF models had 1000 trees. All GBT trees had an optimal number of trees (based on validation set loss) less than or equal to 200.

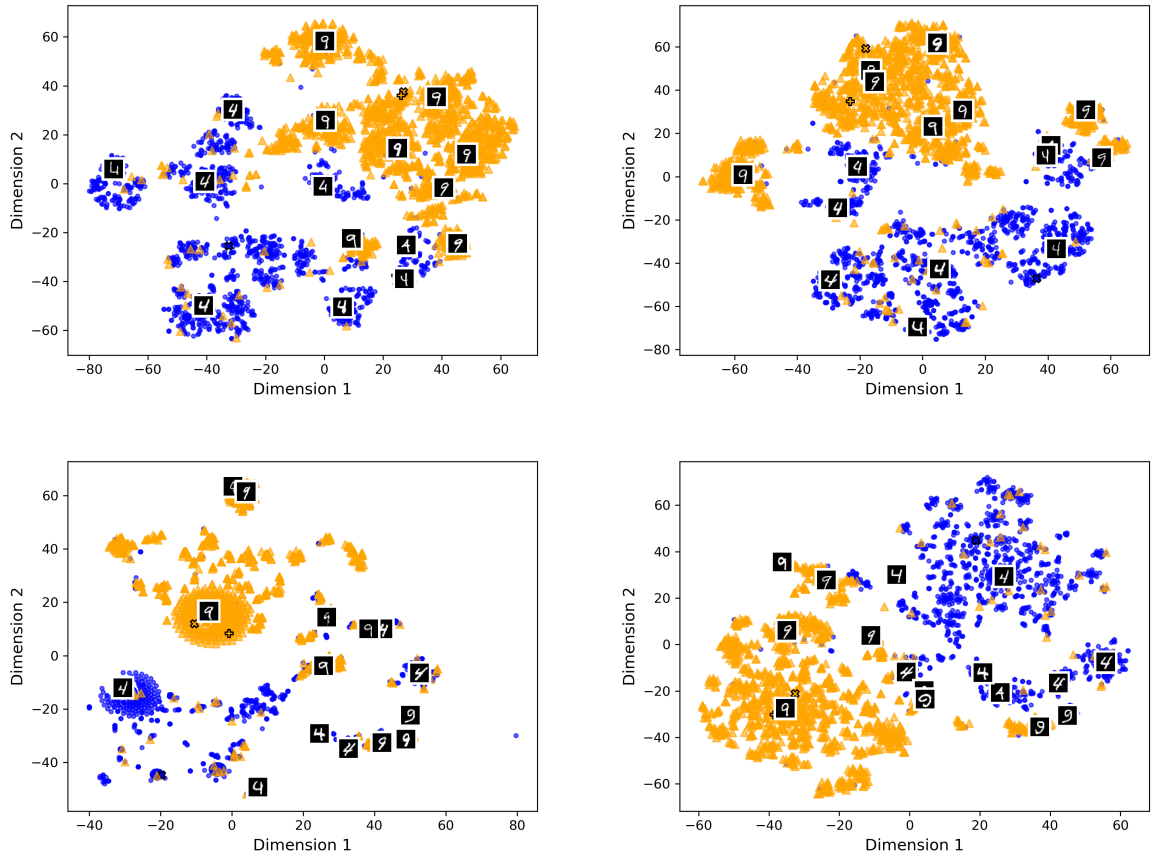

Figure A1: Visualization of distances using t-sne for optimal GBT with unweighted trees (top left), trees weighted by vg^2 (top right), RF with short trees matching GBT depth of 3 (bottom left), and optimal RF (bottom right) with mean depth 16 on the MNIST (4 vs 9) dataset, using the adaptive greedy submodular (SM-A) prototype selection method. Orange represents the digit 9, blue represents the digit 4. The two points marked by x and + are the same across all subfigures, to indicate how points move across different distance representations. Note that the bottom right subfigure is the same as the left subfigure in Figure 3.

Prototypes Selected by Different Methods

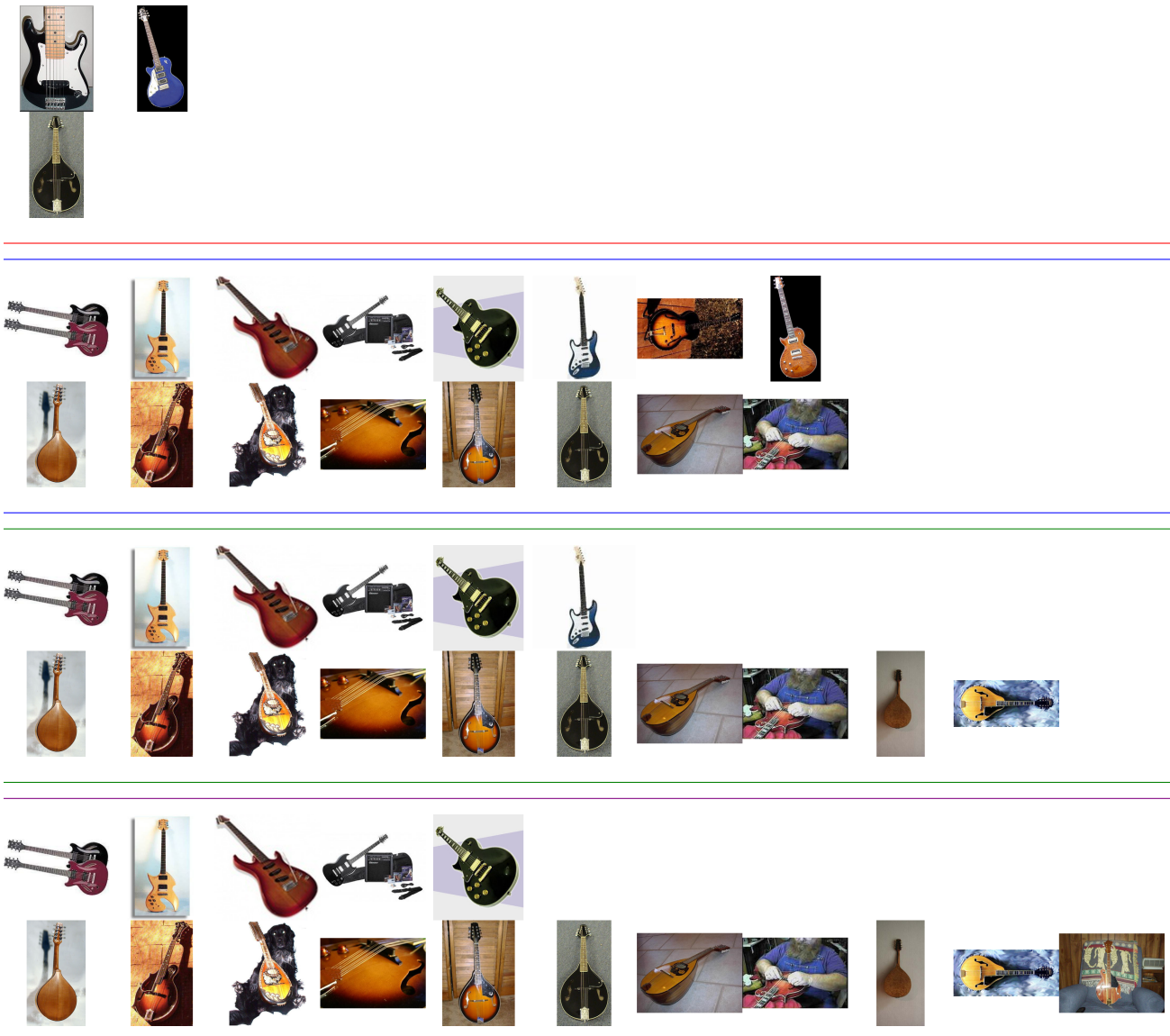

Figure A2: 16 CALTECH256 G-M prototypes found using RF distances for the different prototype selection methods. There are four rows, one for each prototype selection method. Within each row are two subrows, the first for guitar prototypes and the second for mandolin prototypes. The four rows are – first row: supervised greedy (SG). Second row: uniform greedy submodular (SM-U). Third row: adaptive greedy submodular (SM-A). Fourth row: weighted adaptive greedy submodular (SM-WA).