Generalizations of the Random Forest Kernel

Matthew Olson
Department of Statistics, University of Pennsylvania
Philadelphia, PA
maolson@wharton.upenn.edu

Adam Kapelner
Queens College, City University of New York
New York, NY, USA
kapelner@qc.cuny.edu

Abraham J. Wyner
Department of Statistics, University of Pennsylvania
Philadelphia, PA, USA
ajw@wharton.upenn.edu

Richard Berk
Department of Statistics, University of Pennsylvania
Philadelphia, PA, USA
berkr@wharton.upenn.edu

ABSTRACT

A random forest contains an implicit measure of similarity between points in the input space known as the proximity function. The proximity function simply gives the fraction of trees in a forest for which two points fall in the same terminal node. While useful in a great number of applications, this measure is also quite crude, as it ignores all of the information contained in a tree other than its terminal nodes, such as the quality and depth of its splits. In this paper, we construct a rich class of kernels derived from a random forest which generalize the proximity function. Moreover, we demonstrate that these kernels are useful alternatives to the proximity function in probability estimation and visualization examples.

CCS CONCEPTS

• **Information systems → Data mining**;

KEYWORDS

Random forest, kernel learning, visualization, probability estimation

ACM Reference Format:
Matthew Olson, Adam Kapelner, Abraham J. Wyner, and Richard Berk. 2018. Generalizations of the Random Forest Kernel. In *Proceedings of KDD Conference (KDD 2018 Research Paper).* ACM, New York, NY, USA, Article 4, 9 pages.

1 INTRODUCTION

Random forests are among the best off-the-shelf classifiers in existence. A common explanation for their excellent performance relies on ensemble principles: a random forest achieves bias reduction from its deep, un-pruned trees, and variance reduction from averaging de-correlated trees [3]. More recent work has painted random forests through the lens of kernel regression [2, 15, 17]. From this point of view, a random forest works well because it is able to detect training data that is *similar* to a target point when making a prediction [12]. The similarity measure implicit in a random forest is the *proximity function*, which measures the fraction of trees in the forest for which two points appear in the same terminal node.

KDD 2018 Research Paper, August 2018, London, United Kingdom

The proximity function notion of similarity is intriguing since it is learned directly from the data. For example, one finds that in simulated examples with a sparse signal, the proximity function tends to be more “pointed" in the direction of signal variables and “flatter" in the direction of noise variables [15]. In contrast, most kernel methods in machine learning utilize a fixed, predefined kernel, such as a radial basis function. While there have been efforts in the kernel learning literature to learn ideal kernels for different generative processes [10], the random forest does this implicitly and automatically.

We will illustrate in this paper that the proximity is far from the only notion of similarity one can extract from a random forest. One obvious shortcoming of the proximity function is that it only utilizes part of the total knowledge encapsulated in a tree, namely, terminal node membership. In later sections we will provide examples in which terminal node membership is largely uninformative. In these examples, split quality, especially near the root of the tree, is much more important. We will propose a new similarity kernel derived from a random forest that incorporates a richer set of information from the tree. Moreover, we will illustrate the practical advantages of this kernel over the proximity function in probability estimation and visualization tasks.

2 BACKGROUND

We will begin with a mathematical framework for a random forest. This framework will make it natural to draw a close connection between random forests and kernel functions. Specifically, we will find that the *proximity function* falls easily out of the definition of a random forest’s probability estimates. Subsequent sections will make strides toward generalizing this kernel by incorporating additional information learned from the training data during the tree growing process.

2.1 Random Forest Ensembles

In this section we will describe a random forest as originally defined in Leo Breiman’s seminal 2001 paper [3]. A *random forest* is a collection of T unpruned CART decision trees, where each tree is grown on a bootstrap sample of the data, and the split variable at each node of the tree is chosen to be the best among a subset of F randomly chosen predictors. Software implementations often include F as a parameter named `mtry`, as in the *R* library *randomForest* [11], or `max_features`, as in the *Python* machine learning library *sklearn* [16]. We model the randomness injected into each tree - such as the bootstrap sample and the subset of variables considered at each

node - through a random variable $\theta \in \Theta$. For our purposes, θ will be used primarily as a way to index trees in the forest. Finally, observe that each random forest tree partitions the input space $\mathcal{X}$ into a set of hyper-rectangles. For a tree generated by parameter θ, we will denote by $\mathcal{R}_\theta(x)$ the hyper-rectangle that contains a point $x \in \mathcal{X}$.

We can now leverage this notation to more formally define a random forest tree. We will make a couple of simplifying assumptions to lighten the notation, which are often considered in the literature [2]. In particular, we assume that each random forest tree is grown to maximal depth so that each terminal tree node contains exactly one sample point. We will also omit the use of bootstrap sampling. After fitting a random forest to training data $(x_1, y_1), \ldots, (x_n, y_n)$, we denote the prediction made by the tree generated by the parameter θ at a point $x \in \mathcal{X}$ by

$$f(\theta, x) = \sum_{i=1}^{n} \mathbb{I}\left(x_i \in \mathcal{R}_\theta(x)\right) y_i.$$

In words, to make a prediction, simply drop x down the tree, find which cell $\mathcal{R}_\theta(x)$ it occupies, and assign it the label y_i corresponding the unique training point x_i that lies in that cell.

A random forest makes a class prediction by taking the majority vote among $f(\theta_1, x), \ldots, f(\theta_T, x)$. It is also common practice to extract probability estimates from a fitted random forest by considering the fraction of trees in the forest that vote for a certain class [4, 11, 15]. In our notation, we may write the random forest probability prediction at a point x by

$$\begin{aligned}
\widehat{p}_{rf}(y = 1|x) &= \frac{1}{T}\sum_{t=1}^{T} f(\theta_t, x) \\
&= \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} \mathbb{I}\left(x_i \in \mathcal{R}_{\theta_t}(x)\right) y_i \\
&= \sum_{i=1}^{n}\left(\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\left(x_i \in \mathcal{R}_{\theta_t}(x)\right)\right) y_i.
\end{aligned}$$

The item in parenthesis on the last line has the straightforward interpretation as the fraction of trees in the forest for which x_i and x occupy the same terminal node. This quantity is important enough to warrant a definition.

Definition 2.1 (The Proximity Function). The random forest **proximity function** $K^{prox} : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is defined by

$$K^{prox}(x, z) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\left(x_i \in \mathcal{R}_{\theta_t}(x)\right).$$

With this definition, we may equivalently write the probability estimate as

$$\widehat{p}_{rf}(y = 1|x) = \sum_{i=1}^{n} K^{prox}(x, x_i) y_i. \tag{1}$$

Thus, random forest probability estimates have the satisfying interpretation as a weighted sum of the training data, where the weights are given by a similarity function between the target point x and each training points x_i. The larger the value of $K^{prox}(x, x_i)$, the more weight (x_i, y_i) is given when informing the predicted value.

2.2 The Proximity Kernel

The proximity function is an intuitive quantity that measures the similarity between points in the predictor space: the more often that two points x and z appear in the same terminal node of a randomly grown tree, the more similar we expect them to be. Our notation for the proximity function suggests that it is a positive semi-definite kernel (i.e. Mercer kernel), which turns out to be the case [2]. Furthermore, since the proximity function is an average over a large number of trees, one can also argue that $K^{prox}(x, z)$ is a finite sample approximation to $\mathbb{P}_\theta\left(x \in \mathcal{R}_\theta(z)\right)$, and that the convergence to this probability happens almost surely as $T \to \infty$ [2]. Expressed in slightly different terms, the proximity function can then be seen to be an approximation to the probability that a randomly chosen recursive partition of the input space contains the points x and z.

Historically, it has been much more common in the literature to motivate the proximity function through its numerous applications rather than its link to probability estimation. For instance, the proximity function is used for visualization, outlier detection, missing data imputation, archetypal analysis, and clustering [4]. One distinctive advantage of the proximity function over fixed kernels - such as radial basis functions - is that its shape is adaptively learned from the data. This makes it especially appealing for situations in high dimensions with sparse signals. More recently, there has been a focus on tying the proximity function to prediction. Some work has evaluated the proximity function's utility as a kernel in support vector machines [6, 7], while [15] and [17] explore its use in kernel regression.

3 RANDOM TREE KERNELS

We saw that K^{prox} is one natural notion of distance implied by a random forest ensemble. However, this measure is quite crude in some sense, as the only information it uses from a particular tree in the ensemble is terminal node membership. For a given tree, closeness is binary: two points are either in the same terminal node, or they or not. In this section, we will see that there are many other types of tree-based distance measures that incorporate much richer information, including a tree's structure and split quality. We will then demonstrate that alternative kernels that account for this information can lead to measurable improvements in probability estimation and visualization. A more exhaustive comparison in these areas can be found in Section 4.

3.1 Topology-Induced Kernels

In Section 2, we saw that the terminal nodes of a tree produce a partition of the input space. Moreover, this partition is recursive and hierarchical: one can extract sequentially more coarse partitions from the tree by pruning terminal nodes. As a toy example, consider Figure 1. The first split in the tree separates the data in two vertical halves, while the second level of splits separates the right cell into two further horizontal halves. According to the proximity measure of similarity, the points x and z have a similarity of 0, despite sharing the same cell of the first partition.

This binary notion of distance can be too crude in some circumstances. For instance, suppose we want to analyze the performance of school-age children on a standardized test. We might produce a

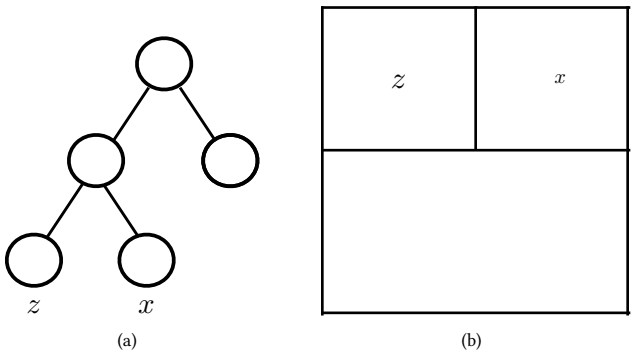


Figure 1: Equivalence between tree representation (a) and training data partition (b).

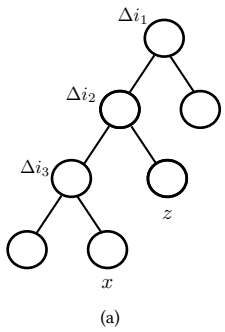


Figure 2: Tree along with node impurity decreases to illustrate the calculations of $K^{\Delta}(x,z)$.

hierarchical partition of these students according to school, grade, and class. We would expect that two students taken from the same school and grade would display some level of similarity on this test, even if they were not in the same class. This motivates the need to consider similarity between partition cells at depths other than terminal nodes.

One such notion of tree-based distance is the *path metric*, which is simply the shortest distance between two nodes in a tree where each edge has a cost of one [19]. We can then define a measure of distance $d : \mathcal{X} \times \mathcal{X} \to \mathbb{N}$ between two points in our input space to be the path distance between the terminal nodes containing each of these points. For instance, in Figure 1, the path distance between x and z in this metric is $d(x,z) = 2$.

Previous work has constructed a tree kernel from the path metric, $K_{\lambda}^{path}(x,z) = e^{-\lambda d(x,z)}$ [7]. As the parameter λ grows to infinity, K_{λ}^{path} reduces to K^{prox}. As λ decreases to zero, the similarity of two points (with respect to this metric) only depends on the fraction of the trees in the forest for which both points share splits near tree roots. In Section 3.2, we will prove that the K_{λ}^{path} is in fact a kernel, and in Section 4 we will analyze its performance when used in kernel regression for probability estimation.

Metrics based solely on *path distance* suffer from the obvious shortcoming that they ignore the informativeness of the splits producing the tree nodes. This problem is especially relevant in a random forest, for which the most informative splits in a given tree tend to occur near the root for large values of F [18].

In order to make this more clear, let us consider a simple probability model in which x is drawn uniformly on $[0,1]^2$ and conditionally on x, $p(y = 1|x_1 > 0) = 0.3$ and $p(y = 1|x_1 \leq 0) = 0.7$. Only the first coordinate of x informs the conditional probability of the response. Suppose we fit a random forest tree to a data set generated by *i.i.d.* (x,y) pairs generated from this probability model. If a node in the tree splits on the first coordinate (and assuming this happens at the optimal population split $x_1 = 0$), all subsequent splits by its descendants will be due to noise alone. Furthermore, note that with high probability, any terminal node will contain an ancestor node that splits on the first coordinate. Thus, the path distance between two nodes is irrelevant, especially near the bottom of the tree. What does matter is the whether two nodes share a certain *informative* split.

One may wonder if there are other types of kernels that can be built from tree topology alone. A natural way to approach this problem is to form the graph adjacency matrix associated with the tree, and to consider various metrics on the rows of this matrix. However, it was shown that practically all metrics one might consider, including Cosine similarity, Tanimoto similarity, Pearson correlation, or resistance measures, all reduce to degenerate measures similar to the proximity matrix [22]. This occurs since the degree of a tree leaf is one. In other words, one must look elsewhere to find other tree based metrics.

3.2 Incorporating Split Information into Similarity Measures

The previous section motivates our search for tree-based similarity measures that incorporate split information at each node. The most natural measure of split quality is the magnitude of the resulting impurity decrease [5]. This quantity also has an important role in tree-based variable importance [5, 13]. Let us recall the node splitting process that occurs when building a CART tree rooted at a node t. Among all possible candidate splits that produce two daughter nodes t_L and t_R, the CART procedure chooses the split for which the impurity decrease

$$\Delta i(\mathrm{t}) \equiv \phi(\mathrm{t}) - (p_L \phi(\mathrm{t}_L) + p_R \phi(\mathrm{t}_R)) \tag{2}$$

is maximized, where p_L and p_R are the fraction of training points from t that are located in nodes t_L and t_R, respectively, and ϕ is a measure of impurity, such as the Gini index.

We propose a new kernel K^{Δ} that measures the similarity between two points x and z in a way can be thought of as a weighted graph distance. For each of these points, we can keep track of the values of the impurity decreases Δi as we traverse from the root node to each of the terminal nodes containing x and z, respectively. Intuitively, we assign a similarity to x and z depending upon what fraction of the total decrease in node purity from root to leaf is shared for both points.

In order to make our discussion concrete, we will develop our definition in the context of the tree in Figure 2. The path leading to the terminal node containing x results in a total decrease in

impurity of $\Delta i_1 + \Delta i_2 + \Delta i_3$, while the path leading to z results in a total decrease of $\Delta i_1 + \Delta i_2$. The terminal cells that contain x and z share only one common split that leads to an impurity decrease of Δi_1. We do not count the decrease Δi_2 which occurs at the least common ancestor node as common for the obvious reason that x and z are assigned different "directions" after passing through this split. Given these quantities, we define, the Δ-similarity between x and z as

$$K^{\Delta}(x,z) \equiv \sqrt{\left(\frac{\Delta i_1}{\Delta i_1 + \Delta i_2 + \Delta i_3}\right)\left(\frac{\Delta i_1}{\Delta i_1 + \Delta i_2}\right)}.$$

An alternative way of viewing this quantity is as follows. The total amount of decrease in node impurity leading to x is $\Delta i_1 + \Delta i_2 + \Delta i_3$, and a fraction $\frac{\Delta i_i}{\Delta i_1 + \Delta i_2 + \Delta i_3}$ of this decrease is shared with z. Conversely, a fraction of $\frac{\Delta i_i}{\Delta i_1 + \Delta i_2}$ of the total path sum of impurity decrease leading to z is shared with x. In order to aggregate these two fractions into one measure, we simply take their geometric mean. We chose the geometric mean as an aggregator because of its connection to the cosine kernel, which appears in the proof of Proposition 3.2. Later discussion will mention some alternatives.

More formally, fix a tree with M total nodes, and index the nodes according to $m = 1, \ldots, M$. Suppose that $A = \{i_1, \ldots, i_a\}$ indexes the nodes leading from the root to the terminal node containing x, and $B = \{j_1, \ldots, j_b\}$ indexes the nodes leading from the root to the terminal node containing z, and ℓ is the index of the least common ancestor between these terminal nodes. This notation allows us to define the kernel K^{Δ}.

Definition 3.1 (The Δ Kernel). The kernel $K^{\Delta} : \mathcal{X} \times \mathcal{X} \to [0,1]$ is defined by

$$K^{\Delta}(x,z) = \sqrt{\left(\frac{\sum_{k \in C} \Delta i(\mathrm{t}_k)}{\sum_{a \in A} \Delta i(\mathrm{t}_a)}\right)\left(\frac{\sum_{k \in C} \Delta i(\mathrm{t}_k)}{\sum_{b \in B} \Delta i(\mathrm{t}_b)}\right)}$$

where $C = (A - \{\ell\}) \cap (B - \{\ell\})$.

This new measure of similarity satisfies some comforting properties. First, if the terminal nodes containing x and z contain no common splits, then $K^{\Delta}(x,z) = 0$. If x and z share the same terminal node, then clearly $K^{\Delta}(x,z) = 1$. Furthermore, recall the one-dimensional probability model given in Section 3.1 for which only the first coordinate contained information about the response. After splitting on the first coordinate, any further splits will lead to small decreases in impurity since the residual values in nodes are equally likely to have the value $y = 1$ or $y = 0$. Thus, the distance between any points located in a subtree rooted at a node cause by a split on the first dimension should be small. Despite being far apart in "node-space," these points are close with respect to K^{Δ}. Finally, note that we defined K_{λ}^{path} and K^{Δ} for individual trees. Their extensions to an ensemble of trees is obvious: compute these kernels for each tree in the ensemble and average.

We will now argue that the three "kernels" K^{prox}, K_{λ}^{path}, and K^{Δ} are all positive semi-definite Mercer kernels. There are two important reasons for doing this. The first is that we want to ensure that these functions have the basic properties we would expect from a well-defined similarity measure, such as symmetry. The second reason is numerical: the routines used in popular kernel methods all require positive semi-definite input matrices (such as in the multidimensional scaling applications we consider later in the paper). Note that it has already been proved in the literature that K^{prox} is a Mercer kernel, but from a much different approach than Proposition 3.2.

Proposition 3.2. *The functions K^{prox}, K_{λ}^{path}, and $K^{\Delta} : \mathcal{X} \times \mathcal{X} \to [0,1]$ are all positive semi-definite Mercer kernels.*

Proof. The proof will rely on basic facts about Mercer kernels that the reader can find in [21] or [20]. In each case, the target kernel is an average of kernels produced by each tree. Since it is the case that if $K_1, \ldots, K_T$ are all kernels so is $K \equiv 1/T(K_1 + \cdots + K_T)$, it remains only to show that the kernel induced by each tree is a Mercer kernel. We will also rely heavily on the fact that if $K : \mathcal{W} \times \mathcal{W} \to \mathbb{R}^+$ is a kernel and $g : \mathcal{X} \to \mathcal{W}$ is a function, then $K_g(x_1, x_2) \equiv K(g(x_1), g(x_2))$ is a kernel. In each case below, we produce a mapping g and a base kernel K.

(a) K^{prox}
Suppose that the tree has M *terminal nodes*, and index each of these nodes by $m = 1, \ldots, M$. Let $g : \mathcal{X} \to [0,1]^M$, where $(g(x))_m$ is the indicator function of whether x lies in the terminal node with index m. Then $K^{prox}(x,z) = \langle g(x), g(z) \rangle$.

(b) K_{λ}^{path}
Suppose that the tree has M *total nodes*, and index these nodes by $m = 1, \ldots, M$. Let $g : \mathcal{X} \to [0,1]^M$, where $(g(x))_m$ is the indicator function of whether x lies in a terminal node that is a descendant of node m. We will show that $K_{\lambda}^{path}(x,z) = \exp\left(-\lambda ||g(x) - g(z)||^2\right)$. In order to see this, notice that $||g(x) - g(z)||^2$ is the path distance $d(x,z)$ between the terminal nodes containing x and z: $||g(x)||^2$ gives the depth of the path leading to the terminal node containing x, $||g(z)||^2$ in the case of z, and $\langle g(x), g(z) \rangle$ is the depth of the least common ancestor of the terminal nodes containing x and z. Finally, $||g(x) - g(x)||^2 = ||g(x)||^2 + ||g(z)||^2 - 2\langle g(x), g(z) \rangle$, which is equivalent to the node distance between x and z in light of the preceding interpretations. The kernel K is taken to be the *Gaussian* kernel $\exp\left(-\lambda ||x - z||^2\right)$.

(c) K^{Δ}
Suppose that the tree has M *total nodes*; we will index all these nodes, excluding the root node, by $m = 1, \ldots, M-1$. Suppose $\phi : \{1, \ldots, M-1\} \to \mathbb{R}^+$ is any function mapping tree nodes to positive scalars. In the case of K^{Δ}, this function can be taken to be $\phi(m) = \sqrt{\Delta i_m}$. Now let $g : \mathcal{X} \to [0,1]^{M-1}$ where $(g(x))_m$ is $\phi(pa(m))$ if the terminal node containing x is a descendant of node m, and 0 otherwise. The kernel K is taken to be the cosine kernel $\frac{\langle x,z \rangle}{||x|| ||z||}$.

□

We conclude this section by mentioning that the types of kernels described here are far from exhaustive, and there are numerous avenues for refinement. Focusing on the case of K^{Δ}, the use of the geometric mean to aggregate the fraction of shared decrease in impurity was chosen since it could be easily mapped to the *cosine* kernel mentioned in the proof of Proposition 3.2. One can also show that the harmonic mean of these two quantities gives rise to

a kernel [1] (or any of the dozens of kernels mentioned in [20]). As another variation, one might consider weighting the decreases in impurity by the depth that these splits occur in the tree. The main point is that the typical way in which one uses trees in a random forest to measure similarity is far from the only one. In the next two sections, we will give simple use-case examples to illustrate the power of this generality.

3.3 Kernel Comparison I: Probability Estimation

We saw in Section 2.1 that the probability estimates produced by a random forest are similar to those produced by kernel regression using the proximity kernel. We can use the class of kernels discussed in the previous section to produce competing kernel regression probability estimates. In particular, in this section we will compare the probability estimates resulting from a random forest to those obtained with kernel regression using K^{Δ} and K_{λ}^{path} for $\lambda = 0.4$, and kernel regression with a radial basis function with variance parameter 0.1. The value of $\lambda = 0.4$ was tuned with out-of-bag (OOB) data. This procedure is described further in Section 4.

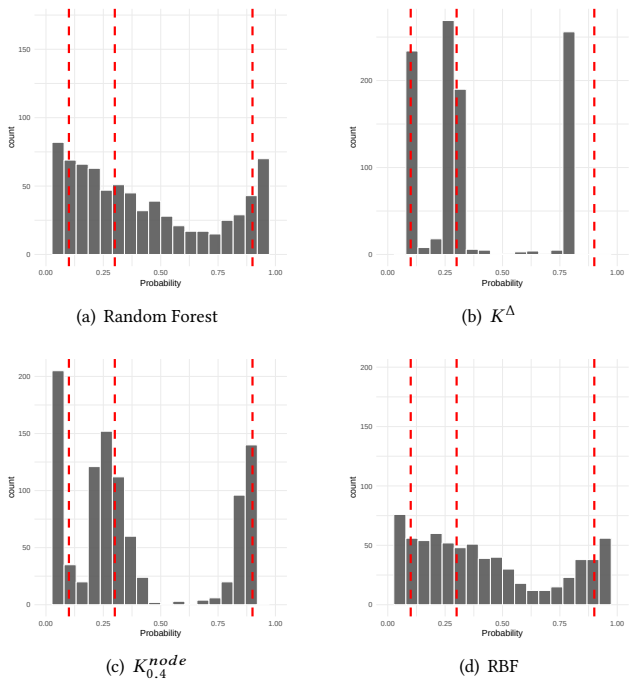
The probability model we consider generates observations in the following manner:

$$\begin{aligned} z &\sim \text{Unif}(1,2,3,4) \\ x|z &\sim \mathcal{N}(\mu_z,\ I_{2\times 2}) \\ y|z &\sim \text{Ber}(p_z) \end{aligned}$$

where $\mu_1 = (-2,-2), \mu_2 = (-2,2), \mu_3 = (2,-2), \mu_4 = (2,2)$ and $p_1 = 0.1, p_2 = 0.3, p_3 = 0.3, p_4 = 0.9$. In words, we draw observations uniformly at random from four clusters, where x is drawn from a cluster dependent normal distribution, and y is a Bernoulli random variable with a cluster dependent success probability. It is worth noting at this point that a simple depth-2 decision tree with four total terminal nodes is optimal for probability estimation in this problem.

We generate $n = 1,000$ observations from this model and compare the resulting probability estimates from each of the four methods on a hold-out set of size $n = 1,000$ ($F = 1$ and 250 trees). Figure 3 plots histograms of the estimated probabilities from each method. Observe first that our underlying probability model only admits three possible probabilities - 0.1, 0.3, and 0.9 - so an ideal estimator would produce a histogram where the mass concentrates at these values. These values are marked in each plot by red vertical lines.

The random forest estimates and the radial basis function probability estimates are both very similar, and quite disperse, with no clear modes. The remaining two sets of estimates all have three sharp modes, although some bias occurs in each. Note that the quality of probability estimate produced by K_{λ}^{path} will depend critically on tuning λ correctly. As $\lambda \to \infty$, the kernel will collapse to K^{prox}, while as $\lambda \to 0$, the kernel effectively causes the tree to collapse to its root. The variance parameter for the RBF was chosen to be very small in order to make the kernel overly "sharp," so that the final estimator resembled something like one-nearest neighbors. It is clear that if the kernel is too sharp, the probability estimates will suffer from high variance.



(a) Random Forest (b) K^{Δ}

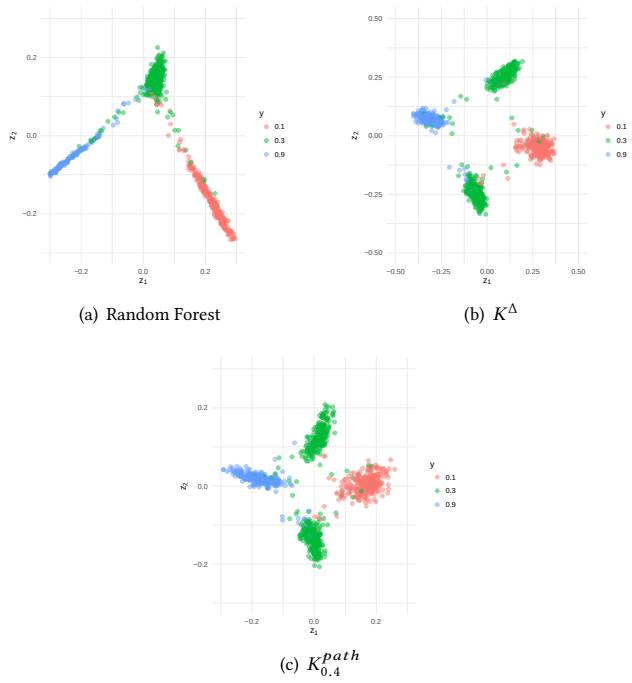
(c) $K_{0.4}^{node}$ (d) RBF

Figure 3: Histogram of probability estimates using a random forest, and kernel regression with kernels K^{Δ}, $K_{0.4}^{path}$, and a radial basis function with variance parameter 0.1.

In this light, we can analyze the kernel weights to see why K^{Δ} and $K_{0.4}^{path}$ perform better than the random forest in this example. The random forest proximity matrix tends to be very sparse, since the training points that impact each prediction - the "voting points" - are concentrated in small neighborhoods [12]. This is not surprising. The proximity measure of similarity is zero or one depending upon whether two points are in the same terminal node of a tree, and terminal nodes map to rectangles with small volume since each random forest tree is grown deep. The kernel matrix for the RBF is also very nearly sparse, since we chose a very small variance parameter. On the contrary, the weights in the other two kernels are much less sparse, meaning that more training points inform the probability estimates. In this particular example, the neighborhoods in which the probability model takes constant values is relatively large, so one can leverage more training data when producing probabilities.

3.4 Kernel Comparison II: Visualization

We will also consider a second application for random forests: dimensionality reduction and data visualization [4]. Given the a set of training points $x_1, \ldots, x_n$ and a kernel K, one can form a "distance" matrix $D \in \mathbb{R}^{n\times n}$ where $(D)_{i,j} = 1 - K(x_i, x_j)$. If K is positive semi-definite, bounded above by 1, and satisfies $K(x_i, x_i) = 1$ for $i = 1, \ldots, n$, it can be shown that there exists an embedding $\psi : \mathcal{X} \to \mathbb{R}^m$ for $m \leq n$ such that $||\psi(x_i) - \psi(x_j)||^2 = D_{i,j}$. One can find such an embedding through multidimensional scaling (MDS). After fitting a random forest, it is common in practice to use the



(a) Random Forest (b) K^{Δ}

(c) $K^{path}_{0.4}$

Figure 4: Proximity plots produces by random forests, K^{Δ}, and $K^{path}_{0.4}$.

proximity kernel K^{prox} to produce an embedding of the data in $\mathbb{R}^2$ using the first few coordinates from MDS. The resulting plots are known as *proximity plots*.

Proximity plots tend to have a characteristic star-shape, which has caused some to question how well they represent the data. The authors in [8] claim that

> "Proximity plots for random forests often look very similar, irrespective of the data, which casts doubt on their utility. They tend to have a star shape, one arm per class, which is more pronounced the better the classification performance."

Returning to discussion in the previous section, since the proximity function has a binary similarity criteria based on node membership, points x and z in the input space can only be considered to be similar if they are also close in a Euclidean sense. In other words, if x and z are far apart in the input space $\mathcal{X}$, the value of $K^{prox}(x,z)$ will also likely be very small, even if $p(y=1|x)$ is close to $p(y=1|z)$. The star shape quality of proximity plots is likely to be due to this phenomenon [8].

We will illustrate the proximity plots produced by applying MDS to K^{Δ} and K^{path}_{λ} can be qualitatively different that traditional proximity plots. We revisit the probability model from Section 3.3 with a slight modification. We keep the set-up exactly the same, except we generate x from a normal distribution in 12 dimensional space instead of 2. The first two coordinates of each mean μ_i are the same as before, but now we add on an additional 10 identical coordinates to each mean. Since our data now lives in a higher dimensional space, visualization is a more interesting problem.

In Figure 4 we plot the first two scaling coordinates using each of the three kernels, where the plot points are colored according to their true conditional class probability values. As reflected in our discussion, the proximity plot produced by the random forest has its characteristic start shape with three branches. The proximity plots produced by K^{Δ} and $K^{path}_{0.4}$ look very similar and break the data up into four distinct clusters. This representation is appealing in the sense that the data can be perfectly separated by a decision tree of depth two which splits on the first two coordinates. In all cases, points with similar conditional class probability are grouped together.

4 EXPERIMENTS

In this section we will conduct a more extensive comparison between the random forest and the derived kernel methods. Specifically, we will augment the probability estimation study in Section 3.3 with eight synthetic data sets, and the visualization study in Section 3.4 with five real world data sets from the UCI repository. Our goal here is not to prove that one method uniformly dominates another, but rather to provide evidence that K^{path}_{λ} and K^{Δ} are useful kernels that produce qualitatively different results from the usual random forest machinery.

4.1 Probability Estimation

We will compare the probability estimates produced by the following three estimators:

(1) $\widehat{p}_{rf}(x) = \frac{1}{T}\sum_{t=1}^{T} f(\theta_t, x)$

(2) $\widehat{p}_{\Delta}(x) = \sum_{i=1}^{n} \frac{K^{\Delta}(x_i,x)y_i}{\sum_{i=1}^{n} K^{\Delta}(x_i,x)}$

(3) $\widehat{p}_{path}(x) = \sum_{i=1}^{n} \frac{K^{path}(x_i,x)y_i}{\sum_{i=1}^{n} K^{path}(x_i,x)}$.

Each estimator is constructed from the same random forest in each simulation, which has $T = 250$ trees and the suggested default value of $F = \sqrt{p}$ for classification, where p is the number of input variables. All models are trained on a data set of size $n = 500$, and are evaluated in terms of misclassification error rate, Brier score, and AUC over a test set of size $n = 1{,}000$. The results we report in this section were constructed as averages over 25 repetitions of each experiment. Finally, we note that when constructing K^{path}, we choose λ by tuning over the out-of-bag (OOB) training data to optimize for the Brier score. We consider five model classes, three of which we compare in low and high dimensional settings, for a total of eight simulation settings. The probability models were chosen to span a wide range of phenomena, including additive and non-additive response surfaces, sparse and dense features, as well as more traditional models such as logistic regression.

4.1.1 Mease Model. The first model is the *Mease model* from [14]. Each predictor x is drawn uniformly at random on $[-28, 28]^2$, and the conditional probability of the response y is given by

$$p(y=1|x) = \begin{cases} 1 & \text{if } ||x||_2 \leq 8 \\ \frac{28-||x||_2}{20} & \text{if } x < 0 \\ 0 & \text{otherwise .} \end{cases}$$

(a) Misclassification Error

	Mease	*One-d*	*One-d (sparse)*	*Friedman*	*Friedman (sparse)*	*Logistic*	*Logistic (sparse)*	*XOR*
K^{Δ}	0.190	0.293	0.302	0.345	0.388	0.337	0.349	0.325
K^{path}	0.189	0.300	0.303	0.321	0.367	0.339	0.344	0.338
random forest	0.206	0.365	0.328	0.286	0.346	0.363	0.350	0.386

(b) Brier Score

	Mease	*One-d*	*One-d (sparse)*	*Friedman*	*Friedman (sparse)*	*Logistic*	*Logistic (sparse)*	*XOR*
K^{Δ}	0.206	0.043	0.098	0.398	0.410	0.126	0.155	0.121
K^{path}	0.108	0.079	0.102	0.378	0.401	0.094	0.137	0.123
random forest	0.156	0.196	0.121	0.349	0.385	0.154	0.127	0.207

(c) Area Under Curve (AUC)

	Mease	*One-d*	*One-d (sparse)*	*Friedman*	*Friedman (sparse)*	*Logistic*	*Logistic (sparse)*	*XOR*
K^{Δ}	0.875	0.712	0.700	0.736	0.695	0.724	0.723	0.697
K^{path}	0.882	0.710	0.700	0.763	0.709	0.722	0.723	0.688
random forest	0.864	0.670	0.694	0.798	0.723	0.689	0.703	0.650

Table 1: Table of probability estimation results.

4.1.2 One Dimensional Model. We consider two versions of the *one dimensional model* which was used in [15] and mentioned in Section 3.1. In this model, x is drawn uniformly on $[0,1]^p$, and y is drawn according to

$$p(y=1|x) = \begin{cases} 0.3 & \text{if } x_1 \geq 0 \\ 0.7 & \text{if } x_1 < 0. \end{cases}$$

In a low dimensional setting, we take $p = 2$, and in a higher dimensional setting we take $p = 50$. Since all of the signal is contained in the first coordinate, the latter model is very sparse.

4.1.3 Friedman Model. The *Friedman model* is a highly nonlinear model taken from [9] which unlike the previous examples, does not have an additive structure. The predictor x is drawn from a p dimensional spherical normal distribution centered at the origin. Condition on x, the response is drawn as

$$\log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = 2(1 - x_1 + x_2 - \cdots + x_6)(x_1 + \cdots + x_6).$$

We also consider dense and sparse version of this model, taking $p = 10$ and $p = 26$.

4.1.4 Logistic Model. The *logistic model* generates data according to the probability model implied by a logistic regression. We draw x uniformly on $[-1,1]^p$, where we take $p = 3$ and $p = 23$. In the former case, every variable in the input space is related to the response, where in the latter case the last 20 variables are noise variables.

$$\log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = x_1 + x_2 + x_3$$

4.1.5 XOR Model. Finally, the *xor model* is a simple data generating process that takes x uniformly on $[-1,1]$, and the probability of y varies according to the parity of the signs of the coordinates of x:

$$p(y=1|x) = \begin{cases} 0.3 & \text{if } x_1 x_2 \geq 0 \\ 0.7 & \text{if } x_1 x_2 < 0. \end{cases}$$

4.1.6 Results. The results from these experiments are summarized in Table 1. Aside from the *Friedman model*, the random forest tended to perform slightly worse than the kernel regression with K^{path} and K^{Δ} across all metrics. The largest performance gaps occurred in the dense *one dimensional* and *xor model*. These are two models in particular in which we would expect these performance gaps: the best predictor is given by a depth one tree in the first case and a depth 2 tree in the second case. The action of K^{Δ} is to emphasize informative splits, which would occur near the root of each random tree, and the tuning implicit in K^{path} would encourage effectively smaller trees.

We also notice that K^{path} and K^{Δ} have similar performance, but K^{path} does slightly better across all metrics in a few models. Again, this might not be surprising since K^{path} incorporates explicitly OOB tuning, while K^{Δ} is entirely determined by the training data used in each tree. While no method uniformly dominates the others, we do see noticeable advantages of our alternative kernels, especially in cases where we expect the underlying data to be generated from a simple model.

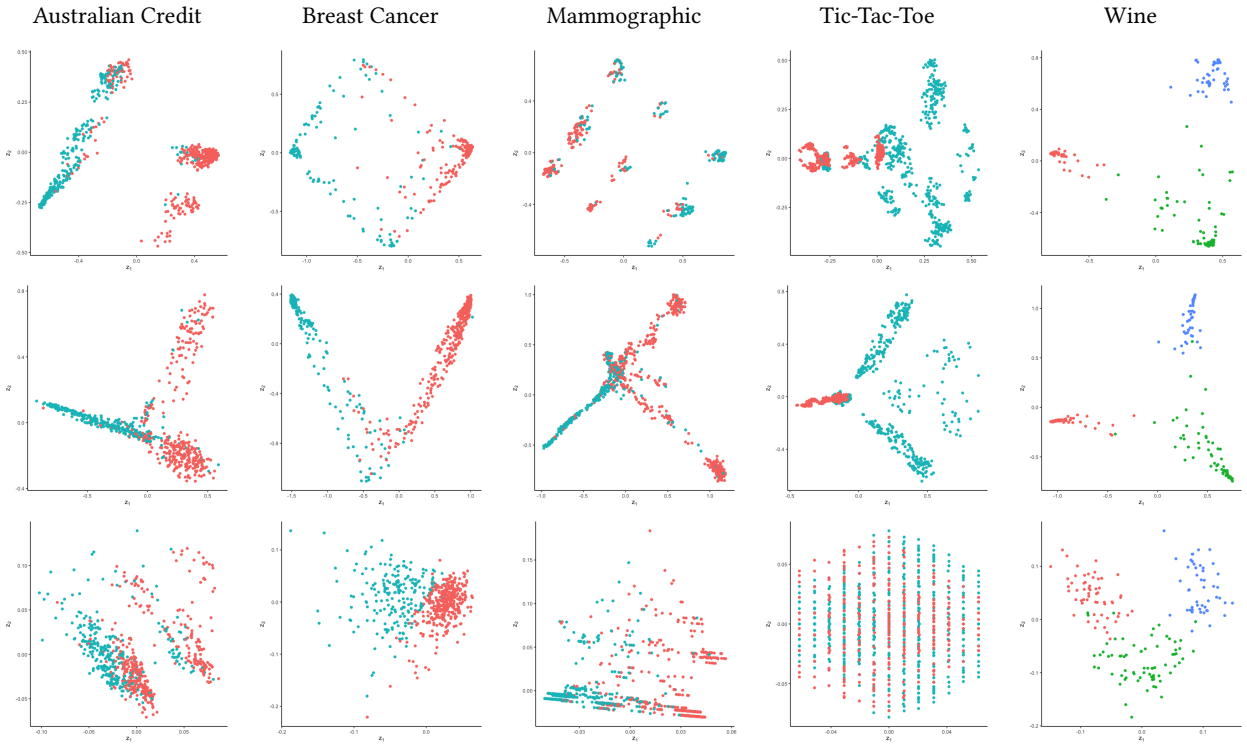


Figure 5: Data set visualizations. The first two rows show proximity plots from a random forest and K^{Δ}, while the last row shows principal components. Points are colored according to class response.

4.2 Visualization

Our preliminary analysis with synthetic data in Section 3.4 indicated that K^{Δ} (and K^{path}) was able to produce qualitatively different proximity plots than a random forest. Here, we construct proximity plots using this kernel and a random forest on five real data sets from the UCI Machine Learning Repository: *Australian credit*, *Wisconsin breast cancer*, *mammographic*, *tic-tac-toe*, and *wine*. We also compare these visualizations with the first two principle components for each data set. We chose to exclude K_{λ}^{path} from the data since it required an explicit tuning parameter, and for the purposes of visualization it is not clear what an objective criteria for this choice is.

Figure 5 shows the proximity plots produced for each data set. The first row consists of traditional proximity plots produced by a random forest, the second row shows those from K^{Δ}, and the third row plots the first two principle components of the data. Each plot shows the class label plotted in a different color: all data sets have two classes, with the exception of *wine*, which has three.

Again, we observe that in each case the random forest proximity plots all have the characteristic star-shape, where each branch contains training data with similar labels. The number of arms in each is either one or two. The proximity plots produced by K^{Δ} are qualitatively much different, with a range of shapes and clusters. Interestingly, both the random forest and K^{Δ} both produce visualizations that tend to separate the data according to class label better than PCA. This is to be expected since these kernel methods see class labels at training time, where PCA does not. The difference in quality of visualization is especially apparent in the *tic-tac-toe* data set. One reason for this might be the presence of categorical variables in the data, for which all 9 predictors in this data set consist of categorical variables. Trees are a much more natural structure for dealing with discrete data compared to PCA, which expects its inputs to be real valued.

Name	n	p	Classes
Australian Credit	690	14	2
Breast Cancer	569	30	2
Mammographic	961	5	2
Tic-Tac-Toe	958	9	2
Wine	179	13	3

Table 2: Dataset Summary

5 DISCUSSION

The goal of this paper was to emphasize the close relationship between random forests and kernel methods. This relationship is both implicit in the way in which a random forest creates probability estimates, and explicit in a number of applications - such as visualization - that deal directly with the proximity function. Our main contribution was to generalize the proximity kernel to other notions of tree-based similarity that take topology and split information into consideration. We then demonstrate that these generalizations

are useful alternatives to traditional random forests in probability estimation, and produce qualitatively different visualizations.

There are a number of avenues for future work. First, one might consider using these kernels in other kernelized tasks, such as support vector machine classification, KernelPCA, or priors for Gaussian processes [6]. It is also natural to consider using these kernels in transfer learning applications. Recall from our previous discussion that one advantage of random forest kernels over fixed kernels (such as radial basis functions) is that they adaptively learn their shape from the training data. Thus, it is plausible that these kernels would perform well on related tasks.

We will also note here that kernel methods based on random forests tend to be computationally expensive, with memory and computational requirements of $O(n^2T)$. This limited us to data sets and simulation settings with $n = 1,000$ or fewer observations. However, all of the operations in kernel construction can be trivially parallelized over trees which reduces this burden.

Finally, we would like to emphasize that the K^{Δ} kernel is only one possible type of similarity metric that incorporates split information. Our proof of Proposition 3.2 illustrates that through a simple feature embedding and appropriate choice of base kernel, one can construct many additional variations.

REFERENCES

[1] Luis Antonio Belanche Muñoz and Alessandra Tosi. 2012. Averaging of kernel functions. In *ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012,..*. 363–368.

[2] Leo Breiman. 2000. *Some infinity theory for predictor ensembles.* Technical Report. Technical Report 579, Statistics Dept. UCB.

[3] Leo Breiman. 2001. Random Forests. *Machine Learning* 45 (2001), 5–32.

[4] Leo Breiman. 2002. Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA* 1 (2002).

[5] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees.* CRC press.

[6] Alex Davies and Zoubin Ghahramani. 2014. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293* (2014).

[7] Cristofer Englund and Antanas Verikas. 2012. A novel approach to estimate proximity in a random forest: An exploratory study. *Expert systems with applications* 39, 17 (2012), 13046–13050.

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning.* Vol. 1. Springer series in statistics New York.

[9] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics* 28, 2 (2000), 337–407.

[10] Tommi Jaakkola and David Haussler. 1999. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems.* 487–493.

[11] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[12] Yi Lin and Yongho Jeon. 2006. Random forests and adaptive nearest neighbors. *J. Amer. Statist. Assoc.* 101, 474 (2006), 578–590.

[13] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. 2013. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems.* 431–439.

[14] David Mease, Abraham J Wyner, and Andreas Buja. 2007. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research* 8, Mar (2007), 409–439.

[15] Matthew Olson and Abraham J. Wyner. [n. d.]. Making Sense of Random Forest Probabilities: a Kernel Perspective. ([n. d.]).

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.

[17] Erwan Scornet. 2016. Random forests and kernel methods. *IEEE Transactions on Information Theory* 62, 3 (2016), 1485–1500.

[18] Erwan Scornet, Gerard Biau, Jean-Philippe Vert, et al. 2015. Consistency of random forests. *The Annals of Statistics* 43, 4 (2015), 1716–1741.

[19] Charles Semple and A Mike. 2003. *Phylogenetics.* Oxford University Press on Demand.

[20] John Shawe-Taylor and Nello Cristianini. 2004. *Kernel methods for pattern analysis.* Cambridge university press.

[21] Alex J Smola and Bernhard Schölkopf. 1998. *Learning with kernels.* GMD-Forschungszentrum Informationstechnik.

[22] Gleb B Sologub. 2011. On measuring of similarity between tree nodes. (2011).