DEPARTMENT OF STATISTICS
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706

TECHNICAL REPORT NO. 1055

May 29, 2002

Random Forests and Adaptive Nearest Neighbors [1]

by

Yi Lin and **Yongho Jeon**

[1]This research is supported by National Science Foundation Grant DMS-0134987. The authors thank Leo Breiman and Murray Clayton for helpful comments and suggestions.

Random Forests and Adaptive Nearest Neighbors

Yi Lin and Yongho Jeon

Abstract

In this paper we study random forests through their connection with a new framework of adaptive nearest neighbor methods. We first introduce a concept of potential nearest neighbors (k-PNN's) and show that random forests can be seen as adaptively weighted k-PNN methods. Various aspects of random forests are then studied from this perspective. We investigate the effect of terminal node sizes and splitting schemes on the performance of random forests. It has been commonly believed that random forests work best using largest trees possible. We derive a lower bound to the rate of the mean squared error of regression random forests with non-adaptive splitting schemes and show that, asymptotically, growing largest trees in such random forests is not optimal. However, it may take a very large sample size for this asymptotic result to kick in for high dimensional problems. We illustrate with simulations the effect of terminal node sizes on the prediction accuracy of random forests with other splitting schemes. In general, it is advantageous to tune the terminal node size for best performance of random forests. We further show that random forests with adaptive splitting schemes assign weights to k-PNN's in a desirable way: for the estimation at a given target point, these random forests assign voting weights to the k-PNN's of the target point according to the local importance of different input variables. We propose a new simple splitting scheme that achieves desirable adaptivity in a straightforward fashion. This simple scheme can be combined with existing algorithms. The resulting algorithm is computationally faster, and gives comparable results. Other possible aspects of random forests, such as using linear combinations in splitting, are also discussed. Simulations and real datasets are used to illustrate the results.

Key Words: Adaptive estimation, classification trees, randomized trees, regression trees, random features.

1 Introduction

Nearest neighbor methods have been studied extensively in the fields of nonparametric statistics and pattern recognition, and continue to be very popular because of their simplicity and because they are very successful for many practical problems. See Fix and Hodges (1951), Dasarathy (1991), Ripley (1996), and Hastie, Tibshirani, and Friedman (2001) for summaries of the extensive literature on the topic. Random forests are among the recent additions to the nonparametric statistics and machine learning toolbox. See Breiman (1999) for an overview. Random forests have been shown to give excellent performance on a number of practical problems, but the mechanism for this is not

fully understood. In this paper we provide a framework connecting random forests with nearest neighbor methods, and study the statistical properties of random forests through this connection.

Nearest neighbor methods and random forests can be used for both regression and classification. In this paper we mainly focus on regression problems, but we give some discussion to classification problems as well. Consider independent and identically distributed observations $\{(\mathbf{x}_i, y_i), i = 1, ..., n\}$ of a random pair $(\mathbf{X}, Y)$. Here $\mathbf{X} = (X^{(1)}, ..., X^{(d)}) \in R^d$ is the input vector, and $Y \in R$ is the response variable. We wish to estimate the regression function $g(\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x})$. For any point $\mathbf{x}_0 \in R^d$, consider an estimator $\hat{g}(\mathbf{x}_0)$ of $g(\mathbf{x}_0)$ based on $\{(\mathbf{x}_i, y_i), i = 1, ..., n\}$. The mean squared error at $\mathbf{x}_0$ is

$$\begin{aligned} \mathrm{MSE}[\hat{g}(\mathbf{x}_0)] &= E[\hat{g}(\mathbf{x}_0) - g(\mathbf{x}_0)]^2 \\ &= [E(\hat{g}(\mathbf{x}_0) - g(\mathbf{x}_0))]^2 + \mathrm{var}(\hat{g}(\mathbf{x}_0)) \\ &= \mathrm{bias}^2 + \mathrm{variance}. \end{aligned}$$

The integrated mean squared error is $\mathrm{IMSE}(\hat{g}) = E_{\mathbf{X}}\mathrm{MSE}[\hat{g}(\mathbf{X})]$.

Given a distance metric, the k nearest neighbor (k-NN) method estimates $g(\mathbf{x}_0)$ by looking at the k sample points that are closest to $\mathbf{x}_0$: $\hat{g}(\mathbf{x}_0) = \sum_{i=1}^n w_i y_i$, where the weight w_i is $1/k$ for the k nearest neighbors of $\mathbf{x}_0$, and is zero otherwise. Different distance measures can be used to form the neighborhood of a target point. The nearest neighbor approach is based on the assumption that the regression function is approximately constant in the neighborhood. Ideally, for a target point $\mathbf{x}_0$, given the number k, the distance measure should be chosen to yield the k nearest neighbors that have mean response values as close to $g(\mathbf{x_0})$ as possible. Hastie and Tibshirani (1996) gave a nice discussion on the choice of distance measure.

Example 1 *To illustrate, we generated* $n = 1000$ *sample input points* $\{\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}), i = 1, ..., n\}$ *uniformly in* $[0, 1]^2$. *The response was generated according to* $Y = g(\mathbf{X}) + \epsilon$, *where* $g(\mathbf{x}) = [x^{(2)}]^2$, ϵ *was simulated independently of* $\mathbf{X}$ *from a normal distribution with zero mean and standard deviation* 0.2. *Figure 1, panel (a), gives a neighborhood of* $\mathbf{x}_0 = (0.5, 0.5)$ *defined by the ordinary Euclidean metric and containing* $k = 100$ *sample points. If we knew the underlying regression function, we could find the neighborhood of* $\mathbf{x}_0$ *that contains the* 100 *sample points whose regression function values are the closest to* $g(\mathbf{x}_0)$. *Thus the assumption that the regression function is approximately constant in the neighborhood is satisfied. That ideal neighborhood is shown in panel (b) of Figure 1. It stretches out all the way in* $x^{(1)}$ *direction.*

It would be possible to do even better in reducing bias in Example 1, if we had the exact knowledge of the regression function and the density of $\mathbf{X}$. By choosing the neighborhood carefully

we can make the bias cancel out, because some of the neighbors have upward bias whereas others have downward bias. However, this seems too much to hope for in practice, and we will not consider it further. In this paper an ideal neighborhood refers to one that contains neighbors that are most similar to the target point in terms of the mean response $g(\cdot)$.

Figure 1 here.

Several authors have studied the problem of how to adaptively choose the metric distance for nearest neighbor methods. Friedman (1994) introduced a k-NN classification method that estimates the local relevance of each input variable for each individual point to be classified, and uses this information to separately customize the metric used to define distance from that object in finding its nearest neighbors. Hastie and Tibshirani (1996) proposed another adaptive NN classification method using a local linear discriminant analysis to estimate an effective metric for computing neighborhoods. Earlier related proposals for adaptive nearest neighbor classification include Short and Fukunaga (1981) and Myles and Hand (1990). The central idea of these existing adaptive nearest neighbor methods is to select k-NN's according to an adaptively chosen local metric. In Section 2 we introduce a different paradigm by considering the set of potential nearest neighbors (k-PNN's), which consists of all the sample points that can be made a k-NN by choosing a reasonable distance metric. (Thus the number of k-PNN's is much larger than k.) In addition, we show in Section 2 that there is a natural connection between the adaptive weighted PNN approach and random forests.

Random forests are classification and regression methods based on growing multiple randomized trees. Breiman (1999) formalized the concept of a random forest with M trees as an estimator consisting of a collection of randomized trees $\{h(\mathbf{x}, \mathbf{\Theta}_m), m = 1, ..., M\}$ where the $\{\mathbf{\Theta}_m\}$ are independent identically distributed random vectors. The m-th randomized tree is grown using the training set and $\{\mathbf{\Theta}_m\}$, resulting in an estimator $h(\mathbf{x}, \mathbf{\Theta}_m)$, where $\mathbf{x}$ is an input vector. The predictions of the M randomized trees are averaged to give the final prediction. Let $\mathbf{\Theta}$ be the generic random vector having the same distribution as $\mathbf{\Theta}_m$, $m = 1, ..., M$. Breiman (1999) showed that, as M goes to infinity, the mean squared generalization error of the random forest goes almost surely to that of $E_{\mathbf{\Theta}} h(\mathbf{x}, \mathbf{\Theta})$.

In random forests, a randomized tree is typically constructed without pruning (Breiman, 1999). The tree building continues until each terminal node contains no more than k training sample points for some pre-specified k (or until the terminal node is pure in classification, in which case the result of the voting is the same as if we continue tree building until there are at most k sample points in the terminal node). Different random forests differ in how randomness is introduced

in the tree building process. Bagging (Breiman, 1996) proceeds by growing trees on bootstrap samples of the training dataset. Randomized outputs (Breiman, 1998) grows trees on the training set with randomly perturbed output variable. Random split selection (Dietterich, 2000) grows trees on the original training dataset. For a fixed number S, at each node, S best splits (in terms of minimizing deviance) are found and the actual split is randomly uniformly selected from them. Random feature selection (Amit and Geman, 1997; Breiman, 1999) searches for the best split over a random subset of the features at each node. Random subspace (Ho, 1998) grows each tree with a random subset of the features. Perfect random trees (Cutler, 1999; Cutler and Zhao, 2000) uses an extreme randomness: at each node, randomly choose a variable to split on, and on the chosen variable choose randomly uniformly a split point between two randomly chosen points coming from different classes.

The statistical mechanism of random forests is under active investigation. Breiman (1999) showed that the expected misclassification rate is bounded by an expression that depends on two parameters: the expected correlation between pairs of classifiers, and the expected accuracy of the individual classifiers. This bound is loose, but suggestive. Breiman (2000) showed that in the population space for binary classification problems, random forests are approximately kernel smoothers. Cutler and Zhao (2000) showed that their perfect random trees method is fitting a block-wise linear surface, and can be computed directly using a recursive equation. Friedman and Hall (1999), and Bühlmann and Yu (2000) provided some explanations of how bagging works.

In this paper we concentrate on random forests that use the original training data (not bootstrap samples) in the construction of randomized trees. Most tree building schemes use splits on the input variables. We concentrate on random forests with such splitting schemes, everywhere except in Section 5, where we consider random forests using linear combinations of input variables for splitting.

We start by looking at random forests locally at a target point $\mathbf{x}_0$. Given the training data $\{(\mathbf{x}_i, y_i), i = 1, ..., n\}$, at a target point $\mathbf{x}_0$, the prediction from the m-th randomized tree is $\sum_{i=1}^{n} W_{im} y_i$, with the weight W_{im} being $1/k_m$ if $\mathbf{x}_i$ is among the k_m sample points in the terminal node containing the target point $\mathbf{x}_0$; and is zero otherwise. This weight depends on the training data $\{(\mathbf{x}_i, y_i), i = 1, ..., n\}$. Given the training data, it is a function of the random vector $\mathbf{\Theta}_m$ in the definition of the random forest. Averaging over M trees, the random forest prediction at $\mathbf{x}_0$ is $\sum_{i=1}^{n} \bar{W}_i y_i$, with $\bar{W}_i = 1/M \sum_{m=1}^{M} W_{im}$. Therefore the random forest can be seen as a weighted average of y_i's, with the weights depending on the training data and $\{\mathbf{\Theta}_m, m = 1, ..., M\}$.

Since $\sum_{i=1}^{n} W_{im} = 1$, we have

$$\sum_{i=1}^{n} \bar{W}_i = 1. \tag{1}$$

It will be clear that the weight $\bar{W}_i$ is zero for most of the sample points. The sample points with positive weights will be called voting points of the random forest for estimating $g(\mathbf{x}_0)$. In general, the voting points are not the nearest neighbors of $\mathbf{x}_0$ under any single distance metric. However, we show in Section 2 that the voting points are all k-PNN's, where k is the terminal node size of the randomized trees. Thus random forests can be viewed as weighted k-PNN methods.

Random forests with different splitting schemes assign weights to k-PNN's in different ways. The splitting schemes can be classified into two broad categories. A non-adaptive splitting scheme partitions the input space in a way that is independent of the response y_i's, given the input vectors $\mathbf{x}_i$'s. One example of non-adaptive splitting schemes is the purely random uniform splitting: for each internal node, we randomly choose a variable to split on, and the split point is chosen uniformly among all possible split points on that variable. This splitting scheme is of theoretical interest and has been considered in Breiman (2000). Most splitting schemes used in practice depend on the response y_i's. One example is the random input selection (Breiman, 1999): at each node, a small group of F input variables are randomly selected, and the best split is searched for over these F input variables. Breiman (1999) compared random input selection with $F = 1$ and $F = \log_2 d + 1$ in classification, and found similar results. For convenience, we will also refer to the random input selection with $F = 1$ as random side selection.

Example 1 (cont.) *To illustrate the difference between the adaptive and non-adaptive splitting schemes, we applied the purely random uniform scheme and the random side selection to the data in Example 1 and* $\mathbf{x}_0 = (0.5, 0.5)$. *The tree building process continues until each terminal node contains at most* $k = 2$ *sample points. Both random forests were built to contain* $M = 1000$ *randomized trees. For each of the two random forests, we get the voting weight of each sample point for estimating* $g(\mathbf{x}_0)$. *Figure 1, (c) and (d) give the voting points and their weights, where the sizes (areas) of the symbols are proportional to the weights.*

It can be seen that in the purely random uniform splitting case, the voting points and weights are roughly equally spread out in the two dimensions. In the random side selection case, the voting weights are spread out in the $x^{(1)}$ direction, but are concentrated around 0.5 in the $x^{(2)}$ direction. This is qualitatively similar to the ideal neighborhood discussed earlier. To give an idea of the relative spreads in the two directions, we calculate the weighted total distance of the voting points from $\mathbf{x}_0$ in each direction: $\sum_{i=1}^{n} \bar{W}_i |x_i^{(j)} - 0.5|$, $j = 1, 2$. We repeat the example 100 times and average. For the purely random uniform scheme, the average weighted total distance is 0.0303

in the $x^{(1)}$ direction, and 0.0311 in the $x^{(2)}$ direction. For the random side selection, they are 0.0603 and 0.0137, respectively. This suggests that the random forest with random side selection adaptively assign weights among the voting points, which will be shown to be k-PNN's.

We study aspects of random forests from the perspective of adaptively weighted PNN's. The key elements in the construction of random forests are the splitting scheme and the specified maximum terminal node size k. In Section 3, we study the effect of the terminal node size k on the statistical accuracy of random forests. The conventional wisdom is that random forests work best using the largest trees possible ($k = 1$). See Breiman (2000, 2002). We derive a lower bound to the rate of the mean squared error of non-adaptive regression random forests, and show that, if large trees are used, then asymptotically it is impossible for non-adaptive random forests to achieve the optimal rate of convergence in regression. It is desirable to have the terminal node size k go up with the sample size n. We illustrate the effect of terminal node sizes on prediction accuracy of adaptive random forests with simulations. In general, it is advantageous to tune the terminal node size for best performance of random forests, though large trees often give the best performance when the dimension of the problem is high.

In Section 4, we study the effects of splitting schemes. It is curious how the adaptiveness shown in Figure 1(d), is achieved, since each variable has an equal chance of being split in the random side selection. We show that there is a fundamental difference between deterministic splitting and random splitting, and that this difference explains the adaptiveness of the random side selection. We give a heuristic argument to show that, for the estimation at a given target point, the random forest with random input selection assigns voting weights to the k-PNN's of the target point according to the local importance of different input variables. Simulations were carried out to illustrate the discussion. We introduce random point selection, a new simple randomization scheme that achieves adaptiveness in a more straightforward fashion.

Breiman (1999) introduced a random forest, Forest-RC, that uses linear combinations of input variables in splitting. We consider the impact of using linear combinations in Section 5. The random point selection can be naturally used with linear combinations. The resulting algorithm achieves similar results to those of Forest-RC, but is computational faster. Discussion and summaries are given in Section 6. Proofs are given in Section 7.

2 Potential nearest neighbors and random forests

We first introduce the concept of monotone distance measures. Throughout this paper, a hyper-rectangle refers to a region of the form $\otimes_{j=1}^{d}[a^{(j)}, b^{(j)}]$. A hyper-rectangle defined by two points $\mathbf{a}$

and **b** is a hyper-rectangle with the two points as opposing vertices. Any distance metric should satisfy the positivity condition and the triangle inequality. In addition, in Euclidean space it is reasonable to require distance measures to satisfy the following monotonicity condition: for any two points **a** and **b**, any point **c** in the hyper-rectangle defined by **a** and **b** is closer to **a** than **b**. This is intuitively clear since **c** is closer to **a** than **b** in every dimension j, $j = 1, ..., d$. We call any distance measure in the Euclidean space satisfying this monotonicity property a monotone distance measure. Such measures include all the common distance measures in the Euclidean space, such as any scaled L_q measures, $0 < q \leq \infty$.

Now we are ready to introduce the potential nearest neighbors.

Definition 1 *A sample point* $\mathbf{x}_i$ *is called a* k*-potential nearest neighbor (*k*-PNN) to a target point* $\mathbf{x}_0$ *if there exists a monotone distance metric under which* $\mathbf{x}_i$ *is among the* k *closest to* $\mathbf{x}_0$ *among all the sample points.*

Therefore, any k-PNN is a k-NN under a suitably chosen monotone metric. The number of k-PNN's is typically much larger than k, and depends on the number and configuration of the sample points.

Existing adaptive nearest neighbor methods can be seen to choose k sample points from the set of all k-PNN's according to an adaptively chosen local metric. Instead we may consider looking directly at the set of all the k-PNN's, and adaptively choose sample points from this set, or adaptively assigns weights to the points in this set (adaptively weighted k-PNN). If K k-PNN's are chosen by an adaptive selection scheme, in general these K points are not the K nearest neighbors to $\mathbf{x}_0$ under any single distance metric. Therefore the adaptive PNN approach is much different from the existing adaptive NN paradigm.

The proof of the following proposition is straightforward and is omitted.

Proposition 1 *A sample point* $\mathbf{x}_i$ *is a* k*-PNN to* $\mathbf{x}_0$ *if and only if there are fewer than* k *sample points other than* $\mathbf{x}_i$ *in the hyper-rectangle defined by* $\mathbf{x}_0$ *and* $\mathbf{x}_i$*.*

Remark 1 *In an exercise (problem 11.6, page 183), Devroye, Györfi, and Lugosi (1996) introduced the so called "layered nearest neighbor rule" as an example of scale-invariant classification rules. The layered nearest neighbors are "sample points* $\mathbf{x}_i$ *for which the hyper-rectangle defined by* $\mathbf{x}_0$ *and* $\mathbf{x}_i$ *contains no other data point." See Figure 2 for an illustration of the layered nearest neighbors. From Proposition 1 we see that the layered nearest neighbors are the same as* 1*-PNN's.*

Figure 2 here.

Consider (regression and classification) random forests with terminal node size k. We can now show that the voting points for a target point $\mathbf{x}_0$ belong to the set of k-PNN's of $\mathbf{x}_0$, no matter what

splitting scheme is used. The terminal nodes of each randomized tree define rectangular areas. If a sample point $\mathbf{x}_i$ is not a k-PNN of $\mathbf{x}_0$, then there are more than k sample points (including $\mathbf{x}_i$) in the hyperrectangle defined by $\mathbf{x}_i$ and $\mathbf{x}_0$. Therefore $\mathbf{x}_i$ cannot be in the terminal node containing $\mathbf{x}_0$. In other words, only k-PNN's can become a voting point. Thus we can view random forests as a weighted k-PNN method. For some splitting schemes including the purely random uniform splitting, it is easy to see from the characterization in Proposition 1 that all k-PNN's of the target point have positive probabilities of being in the same terminal node as the target point. Thus, for these splitting schemes, all k-PNN's can become voting points.

The following results give some idea of how many k-PNN's there are for a given target point. These results will be used in Section 3. Consider a random sample of n points $\{\mathbf{x}_i, i = 1, ..., n\}$ from a density function $f(\mathbf{x})$ supported on $[0,1]^d$. Let $A_k(n, d, \mathbf{x}_0, f)$ denote the expected number of k-PNN's of a fixed point $\mathbf{x}_0 \in [0,1]^d$. For example, $A_k(n, d, \mathbf{0}, 1)$ is the expected number of k-PNN's of the origin $\mathbf{0} = (0, ..., 0)$ among n uniform random points in $[0,1]^d$.

Theorem 1 *$A_k(n, d, \mathbf{0}, 1) = \{k(\log n)^{d-1}/d!\}[1 + o(1)]$, as $n \to \infty$.*

Lemma 1 *$A_k(n, d, \mathbf{0}, 1) \le A_k(n, d, \mathbf{x}_0, 1) \le 2^d A_k(n, d, \mathbf{0}, 1)$, for any $\mathbf{x}_0 \in [0,1]^d$.*

Theorem 2 *Assume the density $f(\mathbf{x})$ is bounded away from zero and infinity in $[0,1]^d$. Then there exists $0 < \Lambda_1 \le \Lambda_2$, such that*

$$\Lambda_1 k(\log n)^{d-1} \le A_k(n, d, \mathbf{x}_0, f) \le \Lambda_2 k(\log n)^{d-1},$$

for any $\mathbf{x}_0 \in [0,1]^d$ and n. That is, the expected number of k-PNN's is of order $k(\log n)^{d-1}$.

3 Terminal node size of randomized trees

In this section we consider the effect of the maximum terminal node size k on the prediction accuracy of random forests. We first consider random forests with non-adaptive splitting schemes.

Theorem 3 *In the regression problem $Y = g(\mathbf{X}) + \epsilon$ with $E(\epsilon) = 0$ and $var(\epsilon) = \sigma^2$, consider the estimation of g based on a random sample $\{(\mathbf{x}_i, y_i), i = 1, ..., n\}$. Suppose $\mathbf{X}$ is distributed in $[0,1]^d$, and its density is bounded away from zero and infinity in $[0,1]^d$. Consider an estimator $\hat{g}$ resulting from a non-adaptive random forest with terminal node size k. There exists $\Lambda_3 > 0$, such that for any n,*

$$MSE[\hat{g}(\mathbf{x}_0)] \ge \Lambda_3 k^{-1} (\log n)^{-(d-1)} \quad \forall \mathbf{x}_0 \in [0,1]^d. \tag{2}$$

$$IMSE(\hat{g}) \ge \Lambda_3 k^{-1} (\log n)^{-(d-1)}. \tag{3}$$

Theorem 3 states that a lower bound to the rate of convergence of the mean squared error of random forests with non-adaptive splitting schemes is $k^{-1}(\log n)^{-(d-1)}$. On the other hand, it is well known (Stone, 1980) that the optimal rate of mean squared error in regression problems is $n^{-2m/(2m+d)}$, where m is the order of smoothness of the regression function g. The smoothness condition can be made precise by means of Sobolev spaces or Hölder classes, but roughly it means the m-th order derivative of g exists. The rate $n^{-2m/(2m+d)}$ is a standard rate that can be achieved by many nonparametric methods. For smooth functions ($m \geq 1$), the rate $n^{-2m/(2m+d)}$ is clearly much faster than the rate achievable by non-adaptive random forests if the terminal node size k does not vary with the sample size n. To make it possible for non-adaptive random forests to achieve the standard rate, the terminal node size k should be made to increase with the sample size n. Therefore, for non-adaptive random forests, growing large trees (k being a small constant) does not always give the best performance. The intuitive reason why non-adaptive random forests with largest trees may not be optimal is that such random forests only use the 1-PNN's of $\mathbf{x}_0$ in the estimation of $g(\mathbf{x}_0)$, and the number of 1-PNN's is of order $O_p[(\log n)^{d-1}]$, which is not big enough for the estimator to achieve the standard rates of convergence for smooth functions.

The above asymptotic argument only applies to situations where the sample size is large compared with the dimension of the problem. In practice, however, we often encounter high dimensional problems with moderate sample size. In such problems the number $(\log n)^{d-1}/(d-1)!$ may not be smaller than $n^{2m/(2m+d)}$, even if the asymptotics indicate otherwise. Take $d = 10$, $m = 2$ for example. When $n = 100000$, $(\log n)^{d-1}/(d-1)! = 9793$, whereas $n^{2m/(2m+d)} = 27$. So $(\log n)^{d-1}/(d-1)!$ is actually much larger than $n^{2m/(2m+d)}$. This is even more the case for larger d and smaller n. Therefore, in high dimensional problems, it can often happen that growing largest trees gives the best performance for non-adaptive random forests.

We believe the results for non-adaptive random forests apply qualitatively to adaptive random forests: growing largest trees is not optimal in general. We give an intuitive reason. If $\mathbf{x}_0 = \mathbf{x}_i$ for some sample point i, then there is just one 1-PNN for $\mathbf{x}_0$ which is itself. Therefore for largest trees the prediction at any training data point is always the same as the response in the training set, i.e., random forests with the largest trees predict the training set perfectly well. Such predictions are usually sub-optimal for predicting future observations when the data are noisy.

To confirm our intuition, we carried out a simulation study to see how the prediction error of random forests varies with the terminal size k. The dataset (Friedman#2) used is available from the R library "mlbench". It was originated from Friedman (1991) and was also described in Breiman (1996). There are 4 independent input variables uniformly distributed over the ranges

$0 \le x^{(1)} \le 100$, $40\pi \le x^{(2)} \le 560\pi$, $0 \le x^{(3)} \le 1$, and $1 \le x^{(4)} \le 11$. The response is

$$y = \left[(x^{(1)})^2 + (x^{(2)}x^{(3)} - (1/(x^{(2)}x^{(4)})))^2\right]^{1/2} + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$. The noise level σ was taken to be 125 to give a signal to noise ratio 3 to 1. In each run, we generated $n = 1000$ training examples and 1000 test examples. The random forest was constructed with 100 randomized trees. The randomized trees were built with random input selection with the size of the random subset being $F = 3$. For each terminal node size k, we calculated the average squared error over the test set. We ran the simulation 100 times and averaged. The result is given in the left panel of Figure 3. We can see the test error goes down as k goes up, the minimum test error is achieved at $k = 20$ or larger.

We then increased the dimension of the regression problem by throwing in six uniform noise input variables, and ran the same experiment on the new problem. The result is given in the middle panel of Figure 3. The minimum test error is achieved at $k = 3$.

Instead of increasing the dimension, in the third case we decreased the sample size of the training set to 200, and repeat the experiment. The minimum of the test error is achieved at $k = 5$ (the right panel of Figure 3). These experiments confirm our expectation that growing largest trees may not be optimal, and that growing largest trees is often close to optimal when the dimension of the problem is high, or the sample size is small.

Figure 3 here.

For classification problems, the asymptotic argument used in regression case can not be applied directly, and the misclassification loss in classification is quite different from the square loss in regression. However, it is still true that random forests with largest trees predict the training set perfectly well, thus are likely to be sub-optimal for predicting future observations when the data are noisy. We believe that, similar to regression random forests, classification random forests with largest trees may not give optimal performance when the dimension is low, and the sample size is large. We illustrate this with a very simple two dimensional simulation. The positive and the negative class are both normal with a common covariance: identity matrix. The two class centers are $(0, 0)$ and $(1, 1)$, respectively. We generated 500 positive examples and 500 negative examples. Random input selection with $F = 1$ was used in classification trees. We used the Gini index when searching for best splits. We varied the terminal node size k from 1 to 50. The random forests were built with 100 randomized trees. The misclassification rate was calculated on a test set of size 1000 generated similarly to the training set. We repeated the whole process 200 times and averaged.

Figure 4 here.

The result is shown in Figure 4. The dotted lines give the standard error calculated from the 200 replicates. We see that as the terminal node size increases from 1 to 50, the misclassification decreases monotonically, from over 30% to under 26%. Therefore largest trees are not optimal in this example.

It is not surprising that random forests with largest trees were optimal in previous empirical studies (Breiman, 1999): the datasets used in the empirical studies tend to be high dimensional, with the sample size small relative to the dimension.

4 Splitting schemes

Given the terminal node size k, the assignment of weights to the k-PNN's is determined by the splitting scheme of the tree construction. A non-adaptive tree growing scheme gives rise to a linear smoother because the weights of the k-PNN's do not depend on the response variable, given the input vector. For most random forests, the tree growing scheme depends on the response variable. In this section we investigate how these random forests assign weights to k-PNN's. In the discussion of this section, we assume that $\mathbf{X}$ is uniform in $[0,1]^d$.

We start with the simple random side selection. Figure 1(d) shows that the random forest with random side selection assigns weights adaptively to the k-PNN's in a desirable way: the weights are spread out in the direction that is not important, but are concentrated in the important direction. This is qualitatively similar to the ideal neighborhood. In ordinary regression trees, the best split is searched for in the current node. Therefore more important variables get more cuts and the neighborhoods become narrower in that direction. Thus desirable local adaptivity is achieved. In the random side selection, each variable has an equal chance of being split on, therefore the adaptiveness shown in Figure 1(d) has to come from a different mechanism. We argue that the adaptiveness comes from the difference between random splitting and deterministic splitting. We start with a formal characterization of this difference by contrasting the behavior of the deterministic bisection and the uniform random split.

First let us consider the deterministic bisection on $[0,1]$. For a fixed point $t_0 \in [0,1]$, consider the interval sequence containing t_0 resulting from the deterministic bisection. Denote the length of the J-th interval in the sequence by D_J. Then it is clear that $D_J = 2^{-J}$, $\forall J \geq 0$.

Now consider the uniform random splitting. Fix $t_0 \in [0,1]$. Starting from $[0,1]$, at a current interval containing t_0, the next split is randomly uniform in the interval. Consider the interval sequence containing t_0 resulting from the uniform random splitting. Denote the length of the J-th interval in the sequence by R_J. Then R_J is a random variable. Let us start with R_1. Consider

the first cut T_1, which is uniform over $[0,1]$. The length of the interval containing t_0 is $R_1 = T_1$ if $T_1 > t_0$, and $R_1 = 1 - T_1$ otherwise. Simple calculation gives

$$E(R_1) = (1-t_0)[(1+t_0)/2] + t_0[1-t_0/2] = 1/2 + t_0(1-t_0).$$

This is larger than D_1 whenever t_0 is not on the boundary of $[0,1]$. An intuitive explanation of this difference is that for the random splitting scheme, the larger child node is more likely to contain t_0.

An explicit expression for $E(R_J)$ is hard to come by for $J > 2$, but we have the following

Theorem 4

$$E(R_J)/D_J \geq 1 + 6t_0(1-t_0)[1.2^J - 1]. \quad \forall J \geq 1.$$

We see that the ratio between the lengths of the intervals containing t_0 resulting from the two splitting schemes goes up exponentially for any t_0 that is not on the boundary of $[0,1]$: deterministic bisection gets to the target point more quickly than random splitting. This sheds light on the working mechanism of the random side selection. Part of the discussion in the rest of this section is heuristic, but the discussion should provide insights into the working mechanism of random forests. We reinforce the heuristic arguments with simulations.

In the ideal case of an equidistant design and a linear signal with no noise, it can be checked that the best split on the variable is equivalent to the deterministic bisection. On the other hand, when we split on a variable with no signal, pure noise, the best split is close to uniformly random splitting. In the random side selection, each input variable has equal chance of being chosen to split. When we split on a variable of strong linear signal, weak noise, the best split is close to bisection; when we split on a variable of very weak signal, the best split is close to a random split. Thus the splits on variables with strong linear signals are getting to the target point more quickly than the splits on variables with weak signals. Therefore the terminal node containing the target point is on average narrower in dimensions with strong linear signals than in dimensions with weak signals. Thus the random side selection achieves desirable local adaptivity.

To obtain further insight let us consider the linear regression function case: $Y = g(\mathbf{X}) + \epsilon$, with $g(\mathbf{x}) = a^{(0)} + \sum_{j=1}^{d} a^{(j)} x^{(j)}$. This is a situation where the relative importance of the input variables on $[0,1]^d$ is clear cut: variables corresponding to larger $|a^{(j)}|$'s are more important. More generally, in a hyper-rectangle with the length of the sides being $r^{(j)}$, $j = 1, ..., d$, the importance of the j-th variables is measured by $|a^{(j)}|r^{(j)}$. This is easily seen by scaling the hyper-rectangle into the hyper-cube $[0,1]^d$.

For a fixed point $\mathbf{x}_0$, consider the hyper-rectangle around $\mathbf{x}_0$ with a fixed volume so that the responses of the points inside the hyper-rectangle are the closest to $g(\mathbf{x}_0)$. This corresponds to

the ideal hyper-rectangle neighborhood of $\mathbf{x}_0$ that contains a fixed number of sample points. Let the length of the sides of this hyper-rectangle be $q^{(j)}$, $j = 1, ..., d$. It is easy to see that they have to satisfy that $q^{(j)}|a^{(j)}| = C$, $\forall j$, for some constant C. That is, in the ideal hyper-rectangle neighborhood, all the variables are equally important.

Now let us consider the random side selection in this linear case. In the random side selection, each input variable has an equal chance of being chosen to split. On the chosen variable, the best split is searched for. In $[0, 1]^d$, the splits on variables with stronger signal behave more similarly to bisection, whereas the sides with weaker signal behave more similarly to the random split. Therefore, on average the length of the more important dimensions shrink faster. In a node with length $r^{(j)}$ in the j-th direction, $j = 1, ..., d$, all directions will have on average the same shrinking rate if $|a^{(j)}|r^{(j)}$ are the same for all direction j. Again this can be seen by scaling the node to $[0, 1]^d$: after such scaling the slopes of the linear regression function are $a^{(j)}r^{(j)}$, and the shrinking rate is not changed by scaling. On the other hand, if the $|a^{(j)}|r^{(j)}$'s are not the same, then similarly we can see that the variable with larger $|a^{(j)}|r^{(j)}$ tends to shrink faster and therefore $|a^{(j)}|r^{(j)}$ decreases faster. Thus in equilibrium the relative lengths of the sides of the node should make all $|a^{(j)}|r^{(j)}$'s be the same, i.e., all variables are equally important in the node. Therefore at equilibrium we get ideal hyper-rectangular neighborhoods.

In the extreme case of one variable being random noise, the corresponding coefficient is 0. Therefore that side always shrinks more slowly than the other sides, and the equilibrium can never be reached. In the case all $a^{(j)}$, $j = 1, ..., d$, are the same, by symmetry in the end the expected spread in each direction should be the same. Thus the equilibrium is achieved in this case.

Example 2 *Assume the same setup as in Example 1, except with the regression function being* $g(\mathbf{x}) = x^{(1)} + 3x^{(2)}$. *Then the ideal adaptive hyper-rectangular neighborhood should have side lengths of ratio* $3 : 1$. *We grew random forests with* 100 *randomized trees, according to random side selection. The terminal node size is* $k = 2$. *Similar to Example 1, we calculated the weighted spread of the voting points for* $\mathbf{x}_0 = (0.5, 0.5)$ *in the two directions. Averaging over* 100 *realizations, they are* 0.0326 *and* 0.0207. *The ratio is* $1.6 : 1$. *We further conducted another experiment: the input vector* $\mathbf{X}$ *was uniformly generated from* $[0, 1] \times [0.4, 0.6]$, *while everything else is the same. Therefore the ratio of the length of the two sides in the starting node is* $5 : 1$. *This time the weighted spread of the voting points in the two directions are* 0.0244 *and* 0.00663. *The ratio is* $3.7 : 1$.

This example suggests that the random side selection is heading to the equilibrium, but does not quite get there. In the example the ideal ratio is $3 : 1$. Starting from an initial ratio of $1 : 1$, the random side selection achieves a ratio $1.6 : 1$; starting from an initial ratio of $5 : 1$, the random

side selection achieves a ratio $3.7:1$. We believe the reason why the random side selection gets to the equilibrium so slowly is the indirect nature in which random side selection achieves adaptivity.

It is possible to design simple splitting schemes that achieve adaptiveness in a more direct way. We consider the following: at the current node, on each input variable we randomly select a split point, and choose the best split among these d splits. We call this scheme the random point selection. This scheme is akin to the ordinary regression tree, but the split point on each input variable is selected randomly, not by exhaustive search. Consider again the linear regression function case $g(\mathbf{x}) = a^{(0)} + \sum_{j=1}^{d} a^{(j)} x^{(j)}$. We show that the random point selection achieves the same equilibrium as the random side selection. In random point selection, more important variables are more likely to be split. In equilibrium all the $|a^{(j)}|r^{(j)}$'s are the same in the node, where $r^{(j)}$ is the length of the j-th direction of the node, $j = 1, ..., d$. This can be seen by scaling the node to $[0,1]^d$, similar to the argument used earlier in the random side selection case. Therefore at equilibrium the relative spread of different directions produced by the random point selection should be close to that of the ideal hyper-rectangular neighborhood. It is our experience that the random point selection goes to the equilibrium much faster than the random side selection. When the random point selection is applied in Example 2, the average weighted spreads are 0.0381 and 0.0123 (ratio $3.1:1$) starting from $[0,1]^2$ and 0.0177 and 0.00654 (ratio $2.7:1$) starting from $[0,1] \times [0.4, 0.6]$. These are close to the ideal ratio of $3:1$.

When the regression function $g(\mathbf{x})$ is not linear, by approximating $g(\mathbf{x})$ locally with linear functions, we expect that the relative spread of the voting weights in different dimensions be determined by the relative magnitude of the partial derivatives of $g(\mathbf{x})$. The relative magnitude of the partial derivatives of the regression function is one reasonable measure of the local importance of the input variables.

Example 3 *To illustrate, we consider a setup that is similar to Example 2, but with* $g(\mathbf{x}) = [x^{(1)}]^2 + [x^{(2)}]^2$. *We consider the estimation at three points* $(0.75, 0.75)$, $(0.25, 0.75)$, $(0.75, 0.25)$. *For random forests with the random point selection, we calculate the weighted spread in the two directions of the voting points. Over* 100 *realizations, the average spreads are* $(0.0219, 0.0229)$, $(0.0367, 0.0120)$, *and* $(0.0133, 0.0418)$, *for the three target points respectively. These ratios are close to the reciprocal of the ratio of partial derivatives at these points (*$1:1$, $3:1$, *and* $1:3$*). When the random side selection is used, the corresponding numbers are* $(0.0248, 0.0249)$, $(0.0345, 0.0156)$, *and* $(0.0174, 0.0421)$.

The random side selection is the random input selection with the size of the random subset being $F = 1$. For random input selection with $F > 1$, at each node the variables are not equally

likely to be split. It is easy to see that more important variables are more likely to be split. In this sense the random input selection with $F > 1$ is similar to the random point selection, and it can be shown with an argument similar to that for the random point selection that in equilibrium the random input selection with $F > 1$ achieves desirable adaptivity.

Both a single regression tree and random forests have the adaptivity property. However, random forests averages over large number of randomized trees, thus stabilizes the result to close to the equilibrium at which the desirable adaptivity is achieved.

The discussions in this section are based on the assumption that the input variables are independent and uniformly distributed. In more general settings the situation is more complicated. In particular, the equilibrium will depend on the density function of the input variables. However, we believe the discussion in the special case sheds light on the general cases.

5 Linear combinations

One of the random forest algorithms considered in Breiman (1999), Forest-RC, uses linear combinations of input variables in the construction of randomized trees. For specified numbers L and F, at any internal node, L variables are randomly selected and added together with coefficients that are uniform random numbers on $[-1, 1]$. F such linear combinations are generated, and then a search is made over these for the best split.

For splitting schemes that use linear combinations of the input variables, the terminal nodes of the randomized trees do not define hyper-rectangular areas. Thus the voting points in the random forests may not be the k-PNN's. For such random forests, we show that the voting weights are allowed to be adaptively assigned according to which linear combination is more important, not just which variable is more important. To illustrate, let us revisit Example 2.

In Example 2, if we do not insist on hyper-rectangular neighborhoods, the ideal neighborhood of the point $\mathbf{x}_0 = (0.5, 0.5)$, that contains sample points whose regression function values are the closest to $g(\mathbf{x}_0)$, should be parallel to and be centered at the line $x^{(1)} + 3x^{(2)} = 2$ (which goes through $\mathbf{x}_0$). Figure 5, left panel, gives the voting points and their weights when Forest-RC is applied to the data in Example 2. We used $L = 2$, $F = 25$, and $k = 4$ in the algorithm. The dotted line is the ideal center line $x^{(1)} + 3x^{(2)} = 2$. The dashed line is the principal component of the weighted voting points. We can see they are well matched, indicating that the voting weights are assigned according to the ideal direction.

Figure 5 here.

The random point selection can be combined with Forest-RC in a natural way. For specified number L and F, at any internal node, F linear combinations are generated in the same way as that in Forest-RC. For each of the linear combinations we randomly use a split point, and the best split among these F random splits is used. This random combination point selection has the advantage of being computationally fast, as no sorting or exhaustive search is needed. In contrast, in Forest-RC, sorting has to be done for every random linear combination, because the best split is searched for on every combination.

Figure 5, right panel, shows the result of the random combination point selection on the data in Example 2. We used $L = 2$, $F = 25$, and $k = 4$. Again the principal component of the weighted voting points matches the ideal center line.

We further test the random combination point selection on several real and synthetic datasets. The datasets and the results are given in Table 1. The datasets are the same as those used in Breiman (1999), except we excluded one dataset whose input variables are all categorical. The Abalone dataset is available at the UCI data repository. The Robot Arm data is provided by Leo Breiman (who obtained the data from Michael Jordan, according to Breiman, 1999). The other datasets are available from the R library "mlbench". For the datasets Friedman#2 and Friedman#3, the noise levels are set so that the signal to noise ratio is 3 : 1. For the first two datasets, the test set error was estimated by training on a random 90% of the data, and testing on the rest of the data. This was repeated 100 times and the test set errors averaged. The numbers in parenthesis are the standard errors based on the repetitions. The Abalone dataset is larger and originally came with a randomly selected 25% of the instances to use as a test set. Following Breiman (1999), we trained on a randomly selected 75% of the data, and tested on the rest. This was repeated 10 times. For Robot Arm data we used the given test set, so there is no standard error estimate. The random combination point selection was run with $L = 2$, $F = 25$, and $k = 4$. This is the setting used in Breiman (1999) for Forest-RC. Tuning (k, for example) may improve performance. The results for Forest-RC are copied from Breiman (1999).

We can see the random combination point selection (shown in Table 1 as Forest-RCP with number of cut points $Q = 1$) gives similar results to that of Forest-RC. A natural extension of the random combination point selection is to use $Q > 1$ random cut points on each linear combination. In fact, Forest-RC can be seen as Forest-RCP with a very large Q. We tried Forest-RCP with $Q = 3$, and the results are similar. Therefore it seems $Q = 1$ is a good choice because it is computationally advantageous.

Breiman (1999) used bagging with Forest-RC. He noted that this seems to enhance accuracy and can be used to give ongoing estimates of the generalization error. We have not used bagging

Data Set	# Training	# Test	# Input	Mean Squared Test Set Error		
				Forest-RC	Forest-RCP	
					$Q = 1$	$Q = 3$
Boston Housing	506	10%	12	10.2	9.26 (0.38)	9.38 (0.42)
Ozone	330	10%	8	16.3	16.73 (0.44)	17.11 (0.51)
Abalone	4177	25%	8	4.6	4.81 (0.13)	4.58 (0.08)
Robot Arm -2	15000	5000	12	4.2	3.9	3.8
Friedman #1	200	2000	10	5.7	5.54 (0.04)	5.34 (0.04)
Friedman #2 +3	200	2000	4	19.6	19.34 (0.10)	19.76 (0.10)
Friedman #3 -3	200	2000	4	21.6	20.46 (0.25)	20.25 (0.23)

$*$ The $+$ and $-$ following a data set name indicate the power of 10 to multiply the result by. For example, the prediction error of Forest-RC on Robot Arm data is 4.2×10^{-2}.

in our experiments, but the extension is straightforward.

6 Summary and discussions

Aspects of random forests can be studied from the perspective of adaptive potential nearest neighbors. For fixed terminal node size k, the voting points of random forests for the estimation at a target point belongs to the set of k-PNN's of the target point. The number of k-PNN's is much larger than k and increases with the sample size. However, asymptotically the rate at which it increases with the sample size is not fast enough for the best performance of random forests. In general tuning k in random forests can enhance estimation accuracy, especially when the dimension of the problem is low, and the sample size is large.

Two simple splitting schemes are the random side selection and the random point selection. Both achieve desirable adaptivity, but in different ways. The random point selection achieves better adaptivity in a straightforward fashion, and is computationally faster. The two basic algorithms can be combined in different ways to develop new algorithms. In particular, the random point selection scheme can be used in existing algorithms to replace the search for best splits. The resulting algorithms are computationally faster.

One difference between random forests and a single regression tree is that, by averaging over a large number of trees, random forests stabilizes the result to close to the equilibrium at which the desirable adaptivity is achieved. Another difference is that in a single tree, the prediction for any

point in the same terminal node is the same, even when the target point is far from being at the center of the terminal node. In random forests, the test target point is always close to the center of the voting points for the target point. This is desirable for enhancing estimation accuracy.

Devroye, Györfi, and Lugosi (1996) (Exercise 11.6) stated that the layered nearest neighbor classification rule is (universally) consistent. Since the layered nearest neighbor rule is equivalent to the unweighted 1-PNN rule, it is hopeful that the techniques introduced in Devroye, Györfi, and Lugosi (1996) can be used to prove the consistency of random forests. This demands further research.

7 Proofs

7.1 Proof of Theorem 1

First we establish the following lemma:

Lemma 2 $A_k(n, d, \mathbf{0}, 1) = k + \sum_{\ell=k+1}^{n} A_k(\ell, d-1, \mathbf{0}, 1)/\ell$.

Proof: Let $\mathbf{x}_i, i = 1, ..., n$, be n uniform random points in $[0,1]^d$. Sort the points according to $x^{(d)}$ direction. Let the points after sorting be $\mathbf{x}_{(i)}$, $i = 1, 2, ..., n$. Denote $\mathbf{x}_{(i)}^{(-d)} = (x_{(i)}^{(1)}, ..., x_{(i)}^{(d-1)})$. Then $\mathbf{x}_{(i)}^{(-d)}$, $i = 1, 2, ..., n$, are iid uniformly distributed in $[0,1]^{d-1}$. Let $Z_i = 1[\mathbf{x}_{(i)}$ is a k-PNN of $\mathbf{0}]$, that is,

$$Z_i = \begin{cases} 1 & : \quad \mathbf{x}_{(i)} \text{ is a k-PNN of } \mathbf{0} \\ 0 & : \quad \text{otherwise} \end{cases}$$

Then we have $A_k(n, d, \mathbf{0}, 1) = E[\sum_{\ell=1}^{n} Z_\ell] = \sum_{\ell=1}^{n} E[Z_\ell]$. Now it is easy to see $Z_\ell = 1$, for $\ell = 1, ..., k$.

For $\ell > k$, we see that $\mathbf{x}_{(\ell)}$ is a k-PNN of $\mathbf{0}$ if and only if $\mathbf{x}_{(\ell)}^{(-d)}$ is a k-PNN of the origin among $\mathbf{x}_{(i)}^{(-d)}$, $i = 1, ..., \ell$, in $[0,1]^{d-1}$. The probability of the latter happening is $A_k(\ell, d-1, \mathbf{0}, 1)/\ell$ by symmetry, since the expected total number of k-PNN of the origin among $\mathbf{x}_{(i)}^{(-d)}$, $i = 1, ..., \ell$, in $[0,1]^{d-1}$, is $A_k(\ell, d-1, \mathbf{0}, 1)$. Therefore

$$E(Z_\ell) = A_k(\ell, d-1, \mathbf{0}, 1)/\ell, \quad \forall \ell > k,$$

and the conclusion of the lemma follows.

Now return to the proof of the theorem. It is clear that $A_k(n, 1, \mathbf{0}, 1) = k$ for any $n \geq k$. Applying the lemma we have

$$A_k(n, 2, \mathbf{0}, 1) = k(1 + \sum_{\ell=k+1}^{n} 1/\ell),$$

the leading term of which is $k \log n$, since $\sum_{i=1}^{n} 1/i = \log n + \gamma + o(1)$, where $\gamma = .57...$ is Euler's constant. We further get (the notation $\approx$ keeps track of the leading term)

$$\begin{aligned} A_k(n, 3, \mathbf{0}, 1) &= k + \sum_{\ell=k+1}^{n} A_k(l, 2, \mathbf{0}, 1)/l \\ &\approx k(1 + \sum_{\ell=k+1}^{n} \frac{\log l}{l}) \\ &\approx k(1 + \int_{k+1}^{n} \frac{\log x}{x} dx) \\ &\approx k(\log n)^2/2. \end{aligned}$$

Continuing with this argument, by induction we get $A(n, d, \mathbf{0}, 1) \approx k(\log n)^{d-1}/(d-1)!$.

7.2 Proof of Lemma 1

Let $\mathbf{x}_i, i = 1, ..., n$, be n uniform random points in $[0, 1]^d$, and $\mathbf{x}_0$ be any fixed point in $[0, 1]^d$. Let $V_i = 1[\mathbf{x}_i \text{ is a k-PNN of } \mathbf{x}_0]$. Then $A_k(n, d, \mathbf{x}_0, 1) = \sum_i E(V_i) = nP(\mathbf{x}_1 \text{ is a k-PNN of } \mathbf{x}_0)$. Now for $\mathbf{x}_1$ to be a k-PNN of $\mathbf{x}_0$, less than k points among $\mathbf{x}_2, ..., \mathbf{x}_n$ should be in the rectangle defined by $\mathbf{x}_0$ and $\mathbf{x}_1$. So conditioning on $\mathbf{x}_1 = \mathbf{x}$, we have

$$P(\mathbf{x}_1 \text{ is a k-PNN of } \mathbf{x}_0 | \mathbf{x}_1 = \mathbf{x}) = B(k-1; n-1, h(\mathbf{x}, \mathbf{x}_0)),$$

where $B(k; n, p)$ is the cumulative distribution function of Binomial(n, p) distribution at k, and $h(\mathbf{x}, \mathbf{x}_0) = \prod_{j=1}^{d} |x^{(j)} - x_0^{(j)}|$. Thus

$$A_k(n, d, \mathbf{x}_0, 1) = n \int_{[0,1]^d} B(k-1; n-1, h(\mathbf{x}, \mathbf{x}_0)) d\mathbf{x}. \tag{4}$$

In the following we proceed in the $d = 2$ case, the general case is similar. Divide the integral above into integrals over E_1, ..., E_4 (see Figure 6). Since $B(k; n, p)$ is decreasing in p, by appropriate changes of variable we can see that

$$n \int_{E_i} B(k-1; n-1, h(\mathbf{x}, \mathbf{x}_0)) d\mathbf{x} \geq n \int_{E_i} B(k-1; n-1, h(\mathbf{x}, \mathbf{0})) d\mathbf{x}, \quad i = 1, ..., 4. \tag{5}$$

Equality holds in E_1, in other areas the inequality holds. This is also easy to see from a symmetry argument: given that $\mathbf{x}_1$ is in E_1, by symmetry the probability of $\mathbf{x}_1$ being a k-PNN of the origin is the same as the probability of $\mathbf{x}_1$ being a k-PNN of $\mathbf{x}_0$. In all other areas, the probability of $\mathbf{x}_1$ being a k-PNN of the origin is less than the probability of $\mathbf{x}_1$ being a k-PNN of $\mathbf{x}_0$. By (5) we get $A_k(n, d, \mathbf{0}, 1) \leq A_k(n, d, \mathbf{x}_0, 1)$.

Figure 6 here.

On the other hand, it is clear that

$$\int_{E_i} B(k-1;n-1,h(\mathbf{x},\mathbf{x}_0))d\mathbf{x} \le \int_{[0,1]^d} B(k-1;n-1,h(\mathbf{x},\mathbf{0}))d\mathbf{x}, \quad i=1,...,4.$$

For example,

$$\int_{E_3} B(k-1;n-1,h(\mathbf{x},\mathbf{x}_0))d\mathbf{x} = \int_{E_3-\mathbf{x}_0} B(k-1;n-1,h(\mathbf{x},\mathbf{0}))d\mathbf{x} \le \int_{[0,1]^d} B(k-1;n-1,h(\mathbf{x},\mathbf{0}))d\mathbf{x}.$$

The other areas can be treated similarly. This gives $A_k(n,d,\mathbf{x}_0,1) \le 2^d A_k(n,d,\mathbf{0},1)$.

7.3 Proof of Theorem 2

Since the density $f(\mathbf{x})$ is bounded away from zero and infinity, there exist positive constants $0 < C_1 \le 1 \le C_2 < \infty$, such that $C_1^d < f(\mathbf{x}) < C_2^d$. The conclusion of the theorem follows from Theorem 1 and that for any $\mathbf{x}_0 \in [0,1]^d$ and n,

$$(C_1/C_2)^d A_k(n,d,\mathbf{0},1) \le A_k(n,d,\mathbf{x}_0,f) \le 2^d (C_2/C_1)^d A_k(n,d,\mathbf{0},1). \tag{6}$$

In the following we prove (6). Following the proof of (4), we have

$$A_k(n,d,\mathbf{x}_0,f) = n\int_{[0,1]^d} B(k-1;n-1,h_1(\mathbf{x},\mathbf{x}_0))f(\mathbf{x})d\mathbf{x}, \tag{7}$$

with $h_1(\mathbf{x},\mathbf{x}_0) = \int_{R(\mathbf{x},\mathbf{x}_0)} f(\mathbf{u})d\mathbf{u}$, where $R(\mathbf{x},\mathbf{x}_0)$ is the hyper-rectangle defined by $\mathbf{x}$ and $\mathbf{x}_0$. Now since $B(k;n,p)$ is decreasing in p, we get from (7),

$$n\int_{[0,1]^d} B(k-1;n-1,C_2^d h(\mathbf{x},\mathbf{x}_0))C_1^d d\mathbf{x} \le A_k(n,d,\mathbf{x}_0,f) \le n\int_{[0,1]^d} B(k-1;n-1,C_1^d h(\mathbf{x},\mathbf{x}_0))C_2^d d\mathbf{x},$$

with $h(\mathbf{x},\mathbf{x}_0) = \prod_{j=1}^d |x^{(j)} - x_0^{(j)}|$. An argument similar to that employed in the proof of Lemma 1 leads to

$$n\int_{[0,1]^d} B(k-1;n-1,C_2^d h(\mathbf{x},\mathbf{x}_0))C_1^d d\mathbf{x} \ge n\int_{[0,1]^d} B(k-1;n-1,C_2^d h(\mathbf{x},\mathbf{0}))C_1^d d\mathbf{x}$$

$$n\int_{[0,1]^d} B(k-1;n-1,C_1^d h(\mathbf{x},\mathbf{x}_0))C_2^d d\mathbf{x} \le 2^d n\int_{[0,1]^d} B(k-1;n-1,C_1^d h(\mathbf{x},\mathbf{0}))C_2^d d\mathbf{x}$$

Now

$$\begin{aligned}
& n\int_{[0,1]^d} B(k-1;n-1,C_2^d h(\mathbf{x},\mathbf{0}))C_1^d d\mathbf{x} \\
= \;& n\int_{[0,C_2]^d} B(k-1;n-1,h(\mathbf{v},\mathbf{0}))(C_1/C_2)^d d\mathbf{v} \\
\ge \;& n(C_1/C_2)^d \int_{[0,1]^d} B(k-1;n-1,h(\mathbf{v},\mathbf{0}))d\mathbf{v} \\
= \;& (C_1/C_2)^d A_k(n,d,\mathbf{0},1).
\end{aligned}$$

$$
\begin{aligned}
& n\int_{[0,1]^d} B(k-1;n-1,C_1^d h(\mathbf{x},\mathbf{0}))C_2^d d\mathbf{x} \\
= \;& n\int_{[0,C_1]^d} B(k-1;n-1,h(\mathbf{v},\mathbf{0}))(C_2/C_1)^d d\mathbf{v} \\
\leq \;& n(C_2/C_1)^d \int_{[0,1]^d} B(k-1;n-1,h(\mathbf{v},\mathbf{0}))d\mathbf{v} \\
= \;& (C_2/C_1)^d A_k(n,d,\mathbf{0},1).
\end{aligned}
$$

Therefore (6) is proved.

7.4 Proof of Theorem 3

We only need to prove (2), since (3) follows from (2). Let $K = K(\mathbf{x}_0;\mathbf{x}_i, i=1,...,n)$ denote the number of k-PNN's of $\mathbf{x}_0 \in [0,1]^d$ among $\{\mathbf{x}_i, i=1,...,n\}$. Then $K(\mathbf{x}_0;\mathbf{x}_i, i=1,...,n)$ is a random number depending on the configuration of $\{\mathbf{x}_i, i=1,...,n\}$. By Theorem 2, there exists $\Lambda_2 > 0$ such that

$$
EK(\mathbf{x}_0;\mathbf{x}_i, i=1,...,n) \leq \Lambda_2 k(\log n)^{d-1}, \tag{8}
$$

for any $\mathbf{x}_0 \in [0,1]^d$ and n. We prove (2) with $\Lambda_3 = \Lambda_2^{-1}\sigma^2$.

Fix any $\mathbf{x}_0 \in [0,1]^d$. Denote the responses of the k-PNN's of $\mathbf{x}_0$ as $y_{(\ell)}, \ell = 1,...,K$. Conditional on $\{\mathbf{x}_i, i=1,...,n\}$, for random forests with node size k, only k-PNN's can become voting points. That is, $\hat{g}(\mathbf{x}_0) = \sum_{\ell=1}^K \bar{W}_{(\ell)} y_{(\ell)}$. Since the splitting scheme is non-adaptive, the weights $\bar{W}_{(\ell)}$'s are independent of y_i's, given $\{\mathbf{x}_i, i=1,...,n\}$. Therefore,

$$
\begin{aligned}
& E[(\hat{g}(\mathbf{x}_0) - g(\mathbf{x}_0))^2 | \{\mathbf{x}_i, i=1,...,n; \bar{W}_{(p)}, p=1,...,K\}] \\
\geq \;& \mathrm{var}[\hat{g}(\mathbf{x}_0) | \{\mathbf{x}_i, i=1,...,n; \bar{W}_{(p)}, p=1,...,K\}] \\
= \;& \mathrm{var}[\sum_{\ell=1}^K \bar{W}_{(\ell)} y_{(\ell)} | \{\mathbf{x}_i, i=1,...,n; \bar{W}_{(p)}, p=1,...,K\}] \\
= \;& \sum_{\ell=1}^K \bar{W}_{(\ell)}^2 \mathrm{var}[y_{(\ell)} | \{\mathbf{x}_i, i=1,...,n; \bar{W}_{(p)}, p=1,...,K\}] \\
= \;& \sum_{\ell=1}^K \bar{W}_{(\ell)}^2 \sigma^2 \\
\geq \;& \sigma^2/K.
\end{aligned}
$$

The last inequality follows from the fact that $\sum_{\ell=1}^K \bar{W}_{(\ell)} = 1$ (from (1)) and Cauchy-Schwarz inequality. We then have,

$$
\mathrm{MSE}[\hat{g}(\mathbf{x}_0)] \geq \sigma^2 E(1/K) \geq \sigma^2/E(K) \geq \Lambda_3 k^{-1}(\log n)^{-(d-1)}.
$$

We used Jensen's inequality in the second inequality, and (8) in the third inequality.

7.5 Proof of Theorem 4

It is easier to start with a general interval $[a, b]$, instead of the interval $[0, 1]$. Consider the uniform random splitting. Let $R_j(a, b, t_0)$ denote the length of the j-th interval in the sequence containing t_0, starting from $[a, b]$. Denote $L_j(a, b, t_0) = E[R_j(a, b, t_0)]$. We will need the following results.

Lemma 3

$$L_{j+1}(a, b, t_0) = (b - a)^{-1} \left[\int_a^{t_0} L_j(x, b, t_0)dx + \int_{t_0}^{b} L_j(a, x, t_0)dx \right].$$

Proof: Let the first cut be T_1. If T_1 is in (a, t_0), then conditioning on the first cut, the expected value of the length of the interval after $(k + 1)$ cuts is $L_k(T_1, b, t_0)$. If T_1 is in (t_0, b), then conditioning on the first cut, the expected value of the length of the interval after $(k + 1)$ cuts is $L_k(a, T_1, t_0)$. Since the first cut is uniform in (a, b), we get the result.

Lemma 4

$$-(1 - t)^2 \log(1 - t) - t^2 \log t \le (\log 4)t(1 - t) \le 1.4t(1 - t), \quad \forall t \in [0, 1].$$

Proof: consider $f(t) = (\log 4)t(1 - t) + (1 - t)^2 \log(1 - t) + t^2 \log t$. It is symmetric around $t = 0.5$. So all we need to show is that $f(t) \ge 0$, $\forall t \in [0, 0.5]$. It is easy to check that $f(0) = 0$, $f'(0) > 0$, $f(0.5) = 0$, $f'(0.5) = 0$, and $f'(t)$ has only one root in $[0, 0.5)$. Thus the result follows.

Lemma 5

$$\int_a^t (b - t)(t - x)/(b - x)dx + \int_t^b (x - t)(t - a)/(x - a)dx \ge 0.6(t - a)(b - t), \quad \forall t \in [a, b].$$

Proof: By straightforward calculation we get

$$\begin{aligned}
& \int_a^t (b - t)(t - x)/(b - x)dx + \int_t^b (x - t)(t - a)/(x - a)dx \\
= & \int_a^t (b - t)[(b - x) - (b - t)]/(b - x)dx + \int_t^b [(x - a) - (t - a)](t - a)/(x - a)dx \\
= & \ 2(t - a)(b - t) + (b - t)^2 \log[(b - t)/(b - a)] + (t - a)^2 \log[(t - a)/(b - a)] \\
\ge & \ 2(t - a)(b - t) - 1.4(t - a)(b - t) \\
= & \ 0.6(t - a)(b - t).
\end{aligned}$$

The second last step follows from Lemma 4 by a scaling argument (scale the interval $[a, b]$ to the interval $[0, 1]$.

Lemma 6

$$L_J(a, b, t_0) \ge \frac{1}{2} L_{J-1}(a, b, t_0) + 0.6^J (b - t_0)(t_0 - a)/(b - a), \quad \forall J \ge 1.$$

Proof: we prove by induction. It is easy to see that when $J = 1$,

$$L_1(a, b, t_0) = \frac{1}{2} L_0(a, b, t_0) + (b - t_0)(t_0 - a)/(b - a).$$

Assume the validity of the theorem for $J = j$. Now for $J = j + 1$, we have

$$\begin{aligned}
& L_{j+1}(a, b, t_0) \\
= \; & (b-a)^{-1} \left[\int_a^{t_0} L_j(x, b, t_0) dx + \int_{t_0}^{b} L_j(a, x, t_0) dx \right] \\
\geq \; & (b-a)^{-1} \int_a^{t_0} [1/2 L_{j-1}(x, b, t_0) + 0.6^j (b - t_0)(t_0 - x)/(b - x)] dx \\
& + (b-a)^{-1} \int_{t_0}^{b} [1/2 L_{j-1}(a, x, t_0) + 0.6^j (x - t_0)(t_0 - a)/(x - a)] dx \\
= \; & 1/2 L_j(a, b, t_0) + (b-a)^{-1} 0.6^j \left[\int_a^{t_0} (b - t_0)(t_0 - x)/(b - x) dx + \int_{t_0}^{b} (x - t_0)(t_0 - a)/(x - a) dx \right] \\
\geq \; & 1/2 L_j(a, b, t_0) + (b-a)^{-1} 0.6^{j+1} (t_0 - a)(b - t_0)
\end{aligned}$$

The first step used Lemma 3, the second step used the induction assumption, the third step used Lemma 3, the fourth step used Lemma 5. Therefore the result holds for $J = j + 1$. The lemma is proved.

Now we come back to the proof of Theorem 4. By Lemma 6 we have

$$L_J(0, 1, t_0)/2^{-J} \geq L_{J-1}(0, 1, t_0)/2^{-(J-1)} + 1.2^J t_0(1 - t_0)$$

Applying this repeatedly, we get

$$L_J(0, 1, t_0)/2^{-J} \geq L_0(0, 1, t_0) + \sum_{j=1}^{J} 1.2^j t_0(1 - t_0) = 1 + 6 t_0(1 - t_0)[1.2^J - 1].$$

References

[1] Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*. **9**, 1545-1588.

[2] Breiman, L. (1996). Bagging predictors. *Machine Learning*. **26** 2, 123-140.

[3] Breiman, L. (1998). Randomizing Outputs to Increase Prediction Accuracy. Technical Report 518, Statistics Department, University of California, Berkeley. Available at www.stat.berkeley.edu.

[4] Breiman, L. (1999). Random Forests. Technical Report, Statistics Department, University of California, Berkeley. Available at www.stat.berkeley.edu.

[5] Breiman, L. (2000). Some Infinity Theory for Predictor Ensembles. Technical Report 577, Statistics Department, University of California, Berkeley. Available at www.stat.berkeley.edu.

[6] Breiman, L. (2002). Manual on setting up, using, and understanding random forests V3.1. Available at www.stat.Berkeley.EDU/users/breiman/.

[7] Bühlmann, P. and Yu, B. (2000). Explaining bagging. Available at http://www.stat.Berkeley.EDU/users/binyu/publications.html.

[8] Cutler, A. (1999). Fast Classification Using Perfect Random Trees. Technical Report 5/99/99, Department of Mathematics and Statistics, Utah State University.

[9] Cutler, A. and Zhao, G. (2000). Voting Perfect Random Trees. Technical Report 5/00/100, Department of Mathematics and Statistics, Utah State University.

[10] Dasarathy, B. (1991). *Nearest neighbor pattern classification techniques*. IEEE Computer Society Press.

[11] Devroye, L., Györfi, L., Lugosi, G. (1996). *A Probability Theory of Pattern Recognition*. Springer, New York.

[12] Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning* **40**, 139-157.

[13] Fix, E. and Hodges, J. (1951). Discriminatory analysis - nonparametric discrimination: Consistency properties. Technical Report 21-49-004, 4, US Air Force, School of Aviation Medicine, Randolph Field, TX.

[14] Friedman, J. H. (1994). Flexible Metric Nearest Neighbor Classification. Available at http://www-stat.stanford.edu/ jhf/#Reports.

[15] Friedman, J. H. and Hall, P. (1999). On Bagging and Nonlinear Estimation. Available at http://www-stat.stanford.edu/ jhf/#Reports.

[16] Hastie, T. and Tibshirani, R. (1996). Discriminant Adaptive Nearest Neighbor Classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **18** 607-616.

[17] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag.

[18] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **20** 832-844.

[19] Myles, J. and Hand, D. (1990). The multiclass metric problem in nearest neighbor classification. *Pattern Recognition.* **23**: 1291-1297.

[20] Ripley, B. D. (1996). *Pattern recognition and neural networks.* Cambridge University Press.

[21] Short, R. and Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory* **27**: 655-665.

[22] Stone, C. J. (1980). Optimal Rates of Convergence for Nonparametric Estimators. *Annals of Statistics* **8** 1348-1360.

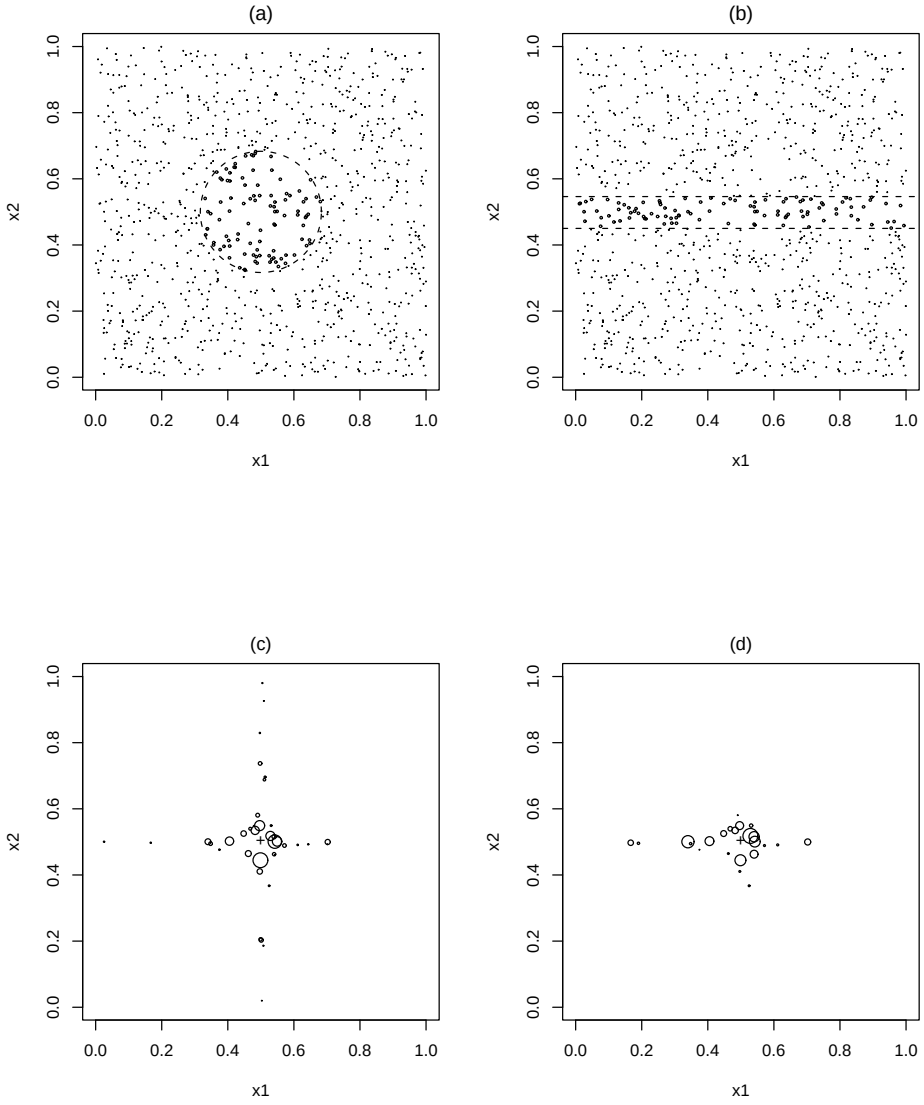

Figure 1: Neighboring points of $\mathbf{x}_0 = (0.5, 0.5)$ used to estimate $g(\mathbf{x}_0)$ in Example 1. (a) Nearest neighbors according to Euclidean distance. (b) Nearest neighbors according to the ideal distance metric. (c) Voting points and their weights in the random forest with the purely random uniform splitting scheme. The areas of the symbols are proportional to the weights. (d) Voting points and their weights in the random forest with the random side selection.

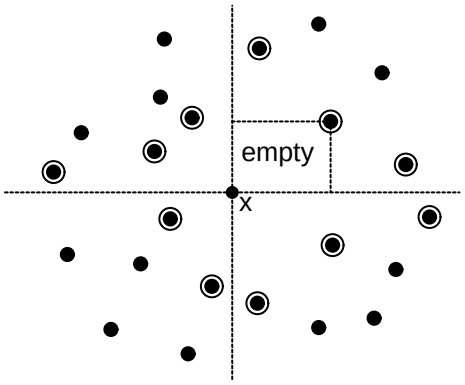

Figure 2: The layered nearest neighbors.

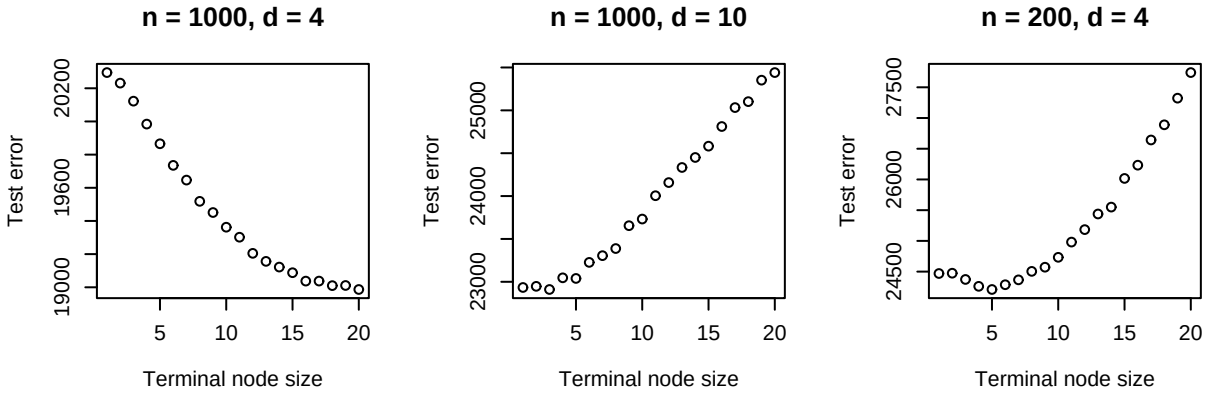

Figure 3: The prediction error of regression random forests varies with the terminal node size k in the three regression problems discussed in Section 3.

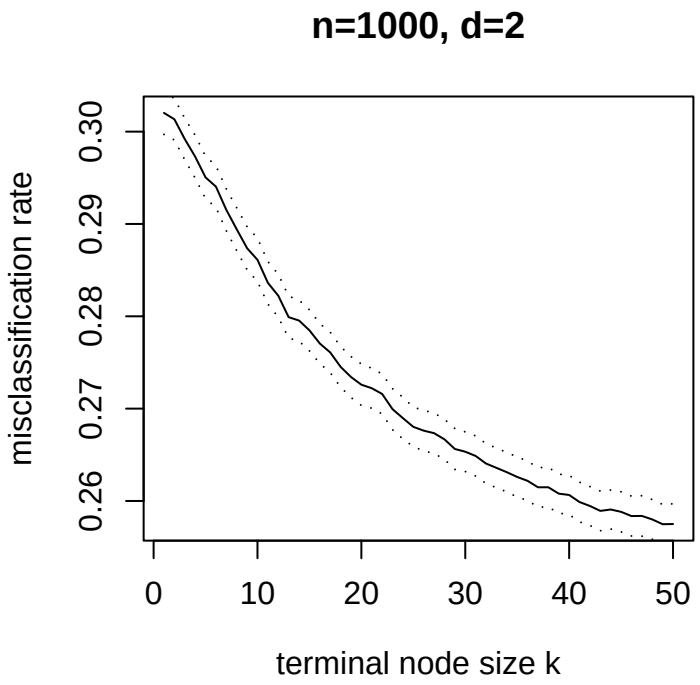

Figure 4: The misclassification rate of the random forest varies with the terminal node size k in the classification problem discussed in Section 3. The dotted lines show the standard error estimate.

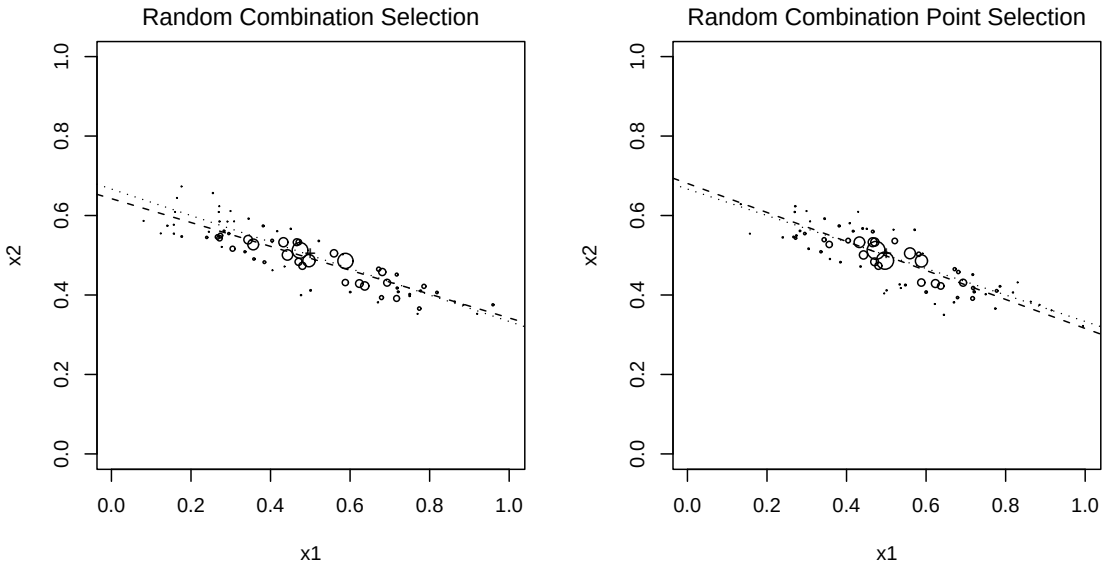

Figure 5: The voting points of random forests with the random linear combination selection (left) and the random linear combination point selection (right). The dotted line is the ideal line, the dashed line is the weighted principal component of the voting points.

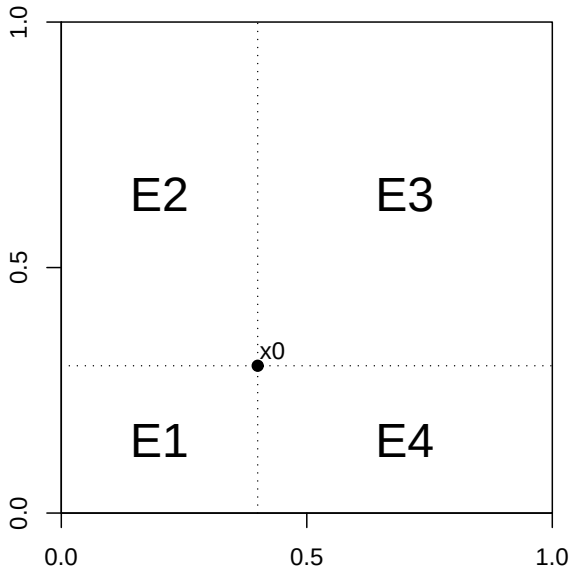

Figure 6: The illustration figure used in the proof of Lemma 1.