2009

Randomness In Tree Ensemble Methods

Joran Elias
The University of Montana

RANDOMNESS IN TREE ENSEMBLE METHODS

by

Joran Elias

B.A. Dartmouth College, USA 2001

M.A. University of Montana-Missoula, USA 2004

presented in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

The University of Montana

May 2009

Elias, Joran M. Ph.D., May 2009 Mathematics

Randomness in Tree Ensemble Methods

Committee Chair: Brian Steele, Ph.D.

Tree ensembles have proven to be a popular and powerful tool for predictive modeling tasks. The theory behind several of these methods (e.g. boosting) has received considerable attention. However, other tree ensemble techniques (e.g. bagging, random forests) have attracted limited theoretical treatment. Specifically, it has remained somewhat unclear as to why the simple act of randomizing the tree growing algorithm should lead to such dramatic improvements in performance. It has been suggested that a specific type of tree ensemble acts by forming a locally adaptive distance metric [Lin and Jeon, 2006]. We generalize this claim to include all tree ensembles methods and argue that this insight can help to explain the exceptional performance of tree ensemble methods. Finally, we illustrate the use of tree ensemble methods for an ecological niche modeling example involving the presence of malaria vectors in Africa.

Acknowledgements

I am grateful for the support from my advisor, Dr. Brian Steele, as well as for the many helpful comments and suggestions from my entire committee: Dr. Dave Patterson, Dr. Jon Graham, Dr. Solomon Harrar and Dr. Jesse Johnson. Finally, I am very thankful for the assistance I have received from the Department of Mathematical Sciences and the Montana-Ecology of Infectious Disease in the pursuit of my degree.

Contents

List of Tables

List of Figures

Chapter 1

Tree Ensemble Methods

1.1 Introduction

Tree ensembles[1] (bagging, random forests, etc.) have proven to be one of the most successful techniques in statistical learning. However, the theoretical understanding of tree ensembles has lagged behind the empirical evidence for their success. The result has been a diverse array of tree ensemble methods that perform very similarly to each other.

We will briefly review decision trees and tree ensemble methods, describe some of the different tree ensemble methods and review some recent attempts at theoretical explanations for their success. We conclude by suggesting that the motivation driving the creation of new tree ensemble techniques is unsatisfactory and by offering a promising alternative.

[1]By this we mean non-adaptive tree ensembles, so we are excluding methods like boosting [15, 28] and arcing [3] that have received more complete theoretical treatments.

1.2 Statistical Learning

In statistical learning we are presented with a data sample, or *training set*, $(y_i, \mathbf{x}_i)$ for $i = 1, \ldots, n$ where $(Y, \mathbf{X})$ arise from some joint distribution $f_{Y,X}(y, \mathbf{x})$. The values $\mathbf{x} = (x_1, \ldots, x_p)$ are referred to as feature vectors or covariates. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the domains of the random variables $\mathbf{X}$ and Y respectively. The goal is as follows: given some independently observed *test point* $\mathbf{x}_0$, accurately predict the value y_0. (There are many other aspects to modeling, of course, but here we will focus on predictive accuracy.)

This framework is divided further based on the nature of the values of y. When y is a continuous random variable, $y \in \mathbb{R}$, then we have a *regression* problem. When y is a discrete random variable taking the unordered values $y \in \{1, \ldots, G\}$, then it is called a *classification* problem. In either case, we model the conditional distribution of y given $\mathbf{x}$: $\mathrm{E}[y|\mathbf{X} = \mathbf{x}]$ for regression and $\Pr[y = g|\mathbf{X} = \mathbf{x}] = \Pr[g|\mathbf{X} = \mathbf{x}]$ for classification.

A *statistical learner* is a function $f : \mathcal{X} \to \mathcal{Y}$ that is intended to serve as an estimate of either $\mathrm{E}[y|\mathbf{X} = \mathbf{x}]$ or $\Pr[g|\mathbf{X} = \mathbf{x}]$. Let $\mathcal{L}(\hat{y}, y)$ denote a loss function that measures the discrepancy between our function's predictions and reality. This might be squared error loss or misclassification rate. The apparent or *empirical* error is the average error over our training sample:

$$\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\hat{y}_i, y_i)$$

The expected prediction error, or *generalization error* is the expected loss over the distribution of $(Y, \mathbf{X})$: $E_{Y,\mathbf{X}}\mathcal{L}(\hat{y}, y)$. The empirical loss is typically a biased estimate of the generalization error and so various methods have been developed for obtaining more accurate estimates (for example, cross-validation, the bootstrap)

1.3 Decision Trees

A decision tree is a piece-wise constant function, $f(\mathbf{x})$, on the feature space $\mathcal{X}$. Let $\Pi = \{R_1, \ldots, R_m\}$ be a partitioning of $\mathcal{X}$. A decision tree is constant on each region $R_i \in \Pi$. Specifically, if $\mathbf{x}_0 \in R_i$, and R_i also contains the training points $\{(y_1, \mathbf{x}_1), \ldots, (y_\ell, \mathbf{x}_\ell)\}$ then

$$f(\mathbf{x}_0) = \begin{cases} \frac{1}{\ell}\sum_i^\ell y_i & y \in \mathbb{R} \\ \arg\max \sum_i^\ell \mathcal{I}(y_i = g) & y \in \{1, 2, \ldots, G\} \end{cases}$$

where $\mathcal{I}$ denotes the indicator function. That is, f simply predicts the average response (or majority class) in each region $R_i \in \Pi$. When y is discrete we have written f as returning a class label, but we can easily modify f to estimate the class probabilities by the proportion of training observations of each class in a region.

Clearly, the challenge is in constructing an optimal partition Π. Searching for an optimal partition is generally a difficult combinatorial problem so we will simplify things considerably. This is done via a greedy algorithm and by restricting ourselves to a small class of partitions: those using only binary splits of the form: $x_i \leq c$.

This greedy algorithm is often called recursive binary partitioning, which leads to the characteristic "tree" structure. We begin with all of the training data in the *root node* and we perform an exhaustive search for the binary split of the form $x_i \leq c$ that minimizes a loss function $\mathcal{L}(\hat{y}, y)$. All of the training observations for which $x_i \leq c$ constitute the left *daughter node* (or child node) and the training observations for which $x_i > c$ constitute the right daughter node. We then repeat our search for a split on each daughter node. The process continues recursively until there is only one training observation in each node. These nodes represent the regions R_i in the partition Π and are called *leaf* or *terminal* nodes.

A partitioning with only one training observation in each terminal node is typically not optimal. Specifically, its generalization error will often be very high. Many methods have been developed to *prune* a maximally grown decision tree (cf. [4]). We will not address these methods here, as they are fairly involved and are not typically used in tree ensemble methods. Instead we will simply refer to trees that are "grown" until there are a maximum number of training observations in each terminal node. When this number is small we will split our training data many times and get a "large" tree, while when this number is large we will split our training data only a few times and get a "small" tree. This is called a *stopping criteria*: we stop splitting nodes when they contain $\leq k$ training observations.

1.4 Tree Ensembles

A tree ensemble is simply a statistical learner that is defined to be the average (or majority vote, in the classification case) over a collection of trees, $f_1, \ldots, f_B$.

The first tree ensemble method was bootstrap aggregation, or *bagging* [2]. Suppose we draw B bootstrap samples from the training set and fit a decision tree using each bootstrap replicate: $f_1^*, \ldots, f_B^*$. Each bootstrap sample will result in a slightly different partitioning and hence a slightly different decision tree. Then the bagged decision tree is

$$f^e(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_b^*(\mathbf{x})$$

for a continuous response y, and

$$f^e(\mathbf{x}) = \arg\max_g \sum_{b=1}^{B} \mathcal{I}(f_b^*(\mathbf{x}) = g)$$

when y is categorical. Hence when y is continuous we average the predictions $f_b^*(\mathbf{x})$ and when y is categorical we take the majority vote among the $f_b^*(\mathbf{x})$. Again, this expression is easily modified to estimate the class probabilities directly. Breiman [2] demonstrated that this simple procedure significantly outperforms single pruned decision trees.

The success of bagging spurred the development of other methods that employ increasing amounts of randomization in the creation of the individual decision trees. Breiman [5] introduced Random Forests (RFs) based on a technique developed by Amit and Geman [1]. In RFs, trees are built on bootstrap samples of the data, as in bagging. However, at each node of each tree, the algorithm only searches over a randomly selected subset of the covariates for the best split. Breiman also discussed randomizing the procedure further by searching for splits over random linear combinations of covariates (RF-RC): at each node select L variables and create F linear combinations of these L variables using coefficients generated randomly on the interval $[-1, 1]$.

Ho [20] suggested the random subspace method (RS), which is essentially identical to RFs but omits the bootstrap resampling. Cutler and Zhao [10] introduced perfect random tree ensembles (PERT) for classification tasks. In PERT, trees are constructed as follows: at each node two data points $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$, $\mathbf{x}_j = (x_{j1}, \ldots, x_{jp})$ are selected at random until they belong to different classes (if this is not possible, all values in this node are in the same class, and the node is terminal); randomly choose a feature, k, and split the data at $\alpha x_{ik} + (1 - \alpha)x_{jk}$ where $\alpha \sim U(0, 1)$. Cutler and Zhao note that this procedure can be performed with or without bootstrapping. They observe that PERT achieves accuracies similar (or better) than AdaBoost, RF, and RF-RC with running times that are considerably faster.

Geurts et al. [17] introduced extremely randomized trees (ERT). Here, at each node a random subset of L variables is selected, as in RFs. However, ERTs then select a split point completely at random on each of these L variables and split the node on the best of these L splits. This procedure is essentially identical to that suggested by Lin and Jeon [22], although they call it random point selection (RPS). They also note that this procedure can easily be incorporated into RF-RC: a split is randomly chosen on each linear combination and the best of these is selected to split the data. Lin and Jeon observe that this procedure maintains very high accuracies while running much faster than traditional bagging or RF procedures that search over all possible splits on each covariate.

Several authors have examined completely random trees (CRT) [13, 14, 23], where each split is chosen randomly, without evaluating any of the commonly used improvement criteria such as gini index, mean squared error (MSE) or variance reduction. Surprisingly, even this amount of randomness seems to work very well, although it appears to perform poorly when a large number of irrelevant features are present [24]. This problem can be ameliorated by varying the probability that one chooses a split randomly or according to some improvement criteria at each node [24].

In general, each author recommends growing large trees (i.e. a small number of training observations in each terminal node), frequently the largest trees possible, although Lin and Jeon [22] suggest that this may not always be optimal. Typically what is recommended is growing trees until there are between 1 and 5 observations per terminal node, depending on the particular algorithm and whether the problem is one of classification or regression.

1.4.1 Theoretical justifications for tree ensembles

We begin by reviewing Breiman's [2] justification for bagging. Suppose that y is continuous (a similar argument exists for when y is discrete, which we omit) and that we can collect multiple independent training sets $\mathcal{T}$ from a distribution and fit a learner (i.e. a decision tree) on each set, $f(\mathbf{x}, \mathcal{T})$. Then the aggregated learner is defined to be the expected prediction over all possible training sets,

$$f_A(\mathbf{x}) = E_{\mathcal{T}}(f(\mathbf{x}, \mathcal{T})).$$

Let $(y_0, \mathbf{x}_0)$ be an independent test point. Using squared-error loss we have,

$$E_{\mathcal{T}}(y_0 - f(\mathbf{x}_0, \mathcal{T}))^2 = y_0^2 - 2y_0 E_{\mathcal{T}} f(\mathbf{x}_0, \mathcal{T}) + E_{\mathcal{T}} f^2(\mathbf{x}_0, \mathcal{T})$$

Substituting $E_{\mathcal{T}} f(\mathbf{x}_0, \mathcal{T}) = f_A(\mathbf{x}_0)$ and applying the inequality $EZ^2 \geq (EZ)^2$ yields

$$E_{\mathcal{T}}(y_0 - f(\mathbf{x}_0))^2 \geq (y_0 - f_A(\mathbf{x}_0))^2.$$

Integrating over $(Y, \mathbf{X})$ on both sides shows that the mean squared error of the aggregated learner is never greater than the mean squared error of a learner fit to a single training set. Breiman asserts that how much improvement we see depends on how unequal the two sides of

$$(E_{\mathcal{T}} f(\mathbf{x}_0, \mathcal{T}))^2 \leq E_{\mathcal{T}} f^2(\mathbf{x}_0, \mathcal{T})$$

are. In essence, the more variable $f(\mathbf{x}, \mathcal{T})$ is with each individual training set $\mathcal{T}$, the greater the advantage of the aggregated predictor f_A. Hence Breiman claims the advantage of bagging is to reduce variance, and will be particularly effective for learners that are in some sense "unstable". By unstable, Breiman means that small perturbations in the training set $\mathcal{T}$ produce large changes in the function $f(\mathbf{x}, \mathcal{T})$.

Obviously, it is not possible to sample randomly with replacement from the distribution that generated the training set $\mathcal{T}$. Instead, the bagging procedure samples with replacement from the empirical distribution placing mass $1/n$ at each training point in $\mathcal{T}$. The hope is that sampling from the bootstrap distribution will be a good approximation to sampling from the distribution that generated the actual data. As has been noted elsewhere [31], this is not really a proof that bagging works so much as a justification for why it's reasonable to try. It is important to note that many of the subsequently developed tree ensemble methods omit bootstrapping entirely and still perform very well. This strongly suggests that data resampling is not a crucial element of the success of tree ensembles.

There have been other limited investigations of tree ensembles, mostly focusing on bagging. Domingos [12] suggested from a Bayesian perspective that bagging "shifts the priors to a more appropriate model space". Buja and Stuetzle [7] examine bagging in the limited case of U-statistics and find that bagging always increases bias and that the effect on variance and mean squared error (MSE) depend on the particular U-statistic and its distribution. Friedman and Hall [16] provide a heuristic argument that bagging can reduce variance, although their argument applies only to smooth estimators, whereas trees are non-smooth (the function is not continuous). Buja [8] also argues in a very general setting that bagged functionals are always "smooth" even if the original functional is not. Buhlmann and Yu [6] argue that bagging results in a smoother decision boundary which in turn reduces variance. Unlike most other investigations, Buhlmann and Yu address bagged decision trees directly, although the theoretical results become very technical and shed little light on the essential mechanism of tree ensembles. Grandvalet [18] observed that bagged decision trees tend to equalize the influence of each training point on the resulting learner. Few of the more randomized tree ensembles have received much theoretical examination. Cutler and Zhao [10] observe that PERT fits a "blockwise multilinear interpolating surface." They note that this can be calculated exactly using some extremely complicated recursive equations but that it is faster to use Monte Carlo estimation.

Some more promising results have been achieved for RFs. Breiman [5] proved that RFs generalization error converges (in probability) as the number of trees increases, meaning that

RFs will not overfit the training data. He also derives an upper bound on the generalization error for RFs for classification, which we review here.

Let $f(\mathbf{x}, \theta)$ denote a learner in an ensemble corresponding to the random vector θ and assume that y is categorical. In the specific case of RFs, $f(\mathbf{x}, \theta)$ is a single tree where θ denotes the bootstrap sample used and the variables selected at each node. We can think of θ as representing the randomness injected into the tree construction procedure. Hence different realizations of the random vector θ will yield slightly different trees. Define the margin function for a RF as

$$mr(\mathbf{x}, y) = P_\theta(f(\mathbf{x}, \theta) = y) - \max_{j \neq y} P_\theta(f(\mathbf{x}, \theta) = j)$$

as the difference between the proportion of ensemble members making a correct prediction at the point $(\mathbf{x}, y)$ and the proportion of ensemble members predicting the second most likely class. The margin function can be thought of as measuring the certainty of an ensemble's predictions. Define the strength of a set of classifiers as

$$s = E_{\mathbf{X},Y} mr(\mathbf{x}, y),$$

the expectation of the margin over all points $(\mathbf{x}, y)$. Note that $-1 \leq mr(\mathbf{x}, y), s \leq 1$. To ease notation, let

$$\hat{j}(\mathbf{x}, y) = \arg\max_{j \neq y} P_\theta(f(\mathbf{x}, \theta) = j)$$

denote the second most commonly predicted class at a point $(\mathbf{x}, y)$. Define the raw margin function as

$$rmg(\theta, \mathbf{x}, y) = \mathcal{I}(f(\mathbf{x}, \theta) = y) - \mathcal{I}(f(\mathbf{x}, \theta) = \hat{j}(\mathbf{x}, y))$$

so that $mr(\mathbf{x}, y)$ is the expectation of $rmg(\theta, \mathbf{x}, y)$ over the random vector θ. Let $\rho(\theta, \theta_1)$ denote the correlation between $rmg(\theta, \mathbf{x}, y)$ and $rmg(\theta_1, \mathbf{x}, y)$ where θ and θ_1 are fixed and the point $(\mathbf{x}, y)$ is allowed to vary. Finally, let $\bar{\rho}$ denote the mean value of this correlation over all pairs of realizations of θ. Breiman proves that the generalization error for RFs, PE^*, is bounded by

$$PE^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{1.1}$$

The correlation $\bar{\rho}$ measures the extent to which the same training points are correctly classified by different trees. For example, if each tree correctly classifies the same subset of training points, then $\bar{\rho} = 1$. A similar bound can be proven for the regression case, relating prediction error to the correlation between residuals and the overall strength of each tree.

Breiman concludes that the goal in constructing tree ensembles is to decrease the correlation between trees while maintaining their overall strength. This conclusion has driven much of the pursuit for random tree building procedures in order to reduce $\bar{\rho}$. Many (PERT, ERT, CRT etc.) are motivated by an attempt to make the individual trees as uncorrelated as possible. However, this path has had several drawbacks:

- The bound in (1.1) is likely to be loose. Indeed, in the unit square the median value of this upper bound is roughly 1.2. In [5] Breiman calculates estimates of $\bar{\rho}$ and s for three data sets. For the sonar data the bound exceeds 1; for the satellite data the bound is about 5 times higher than the actual accuracy achieved and it is fairly tight for the breast data set. The only other instance we have found where $\bar{\rho}$ and s are estimated is in [10] using PERT and CART. The results are summarized in Table 1.1. Note that the upper bound is always far higher than the best error rate achieved with PERT (and is once greater than one). Also observe that in two cases the upper bound for single CART trees is actually *lower* than for PERT, even though their error rates are higher.

Table 1.1: Values for $s, \bar{\rho}$, upper bound from (1.1) and error rates for PERT. Entries in parentheses are values for single trees (CART).

	s (CART)	$\bar{\rho}$ (CART)	Bound (CART)	Error
waveform	0.2 (0.37)	0.06 (0.22)	1.44 (1.39)	16.8
twonorm	0.5 (0.54)	0.07 (0.14)	0.21 (0.34)	3.0
threenorm	0.2 (0.31)	0.04 (0.13)	0.96 (1.22)	15.3
ringnorm	0.27 (0.48)	0.05 (0.14)	0.64 (0.47)	10.7

- Reducing $\bar{\rho}$ has diminishing returns. The most extreme randomization exists in PERT and CRT and the correlations achieved by PERT are remarkably small (Table 1.1). However, the improvements in accuracy over methods like bagging and RFs are relatively modest.

- This bound fails to account for the particular success of randomization in decision tree ensembles as opposed to other learners. In deriving this bound, $f(\mathbf{x}, \theta)$ could refer to any base learner, including those known to receive little benefit from bagging (i.e. k nearest neighbor). Why randomization techniques should be so successful with trees but not with other learners is unclear.

1.5 A Look Ahead

We have given a short introduction to decision trees, tree ensembles and some of the theoretical justifications for the success of these methods. Also, we have argued that the principal theoretical justification has some serious shortcomings. In the following chapter, we will provide an alternative framework for understanding the success of tree ensembles by investigating a connection between tree ensembles, weighted k-nearest neighbor methods and distance metrics.

Chapter 2

Metrics

2.1 Introduction

In this chapter we will expand upon an observation by Lin and Jeon [22] to connect all tree ensemble methods to weighted k-nearest neighbor, or kernel methods. We will discuss the concept of an optimal distance metric and see how this relates to the behavior of tree ensembles in general. We begin with the following definitions.

Definition 2.1.1. *A function* $d(\mathbf{x}, \mathbf{y})$ *is called a* metric *when it satisfies the following conditions,*

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
3. $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$
4. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$.

Definition 2.1.2. *A function* $d(\mathbf{x},\mathbf{y})$ *is called a* pseudometric *when it satisfies conditions 1, 2 and 4 from Definition 2.1.1.*

Clearly, every metric is a pseudometric, but not vice versa. However, any pseudometric d gives rise to a genuine metric in the following manner. Define an equivalence relation between the points $\mathbf{x}$ and $\mathbf{y}$ by

$$\mathbf{x} \sim \mathbf{y} \Leftrightarrow d(\mathbf{x},\mathbf{y}) = 0.$$

Then d is a metric on the resulting equivalence classes.

Definition 2.1.3. *A* kernel *is a non-negative real valued function that satisfies the following two conditions,*

- $\int_{-\infty}^{\infty} K(u)du = 1$
- $K(-u) = K(u)$.

We note here the important point that if K is a kernel, then so is $K_\lambda(u) = \lambda^{-1}K(\lambda^{-1}u)$ for $\lambda > 0$.

2.2 Kernel Methods

2.2.1 Introduction

Kernel methods are a general statistical learning method that estimates the conditional distribution of y locally. The local nature of these methods are obtained via a kernel function that identifies training observations that are in some sense "close" to the test point at which we would like to make a prediction. Almost any model for the conditional expectation of y can be made "local" by introducing weights corresponding to a particular kernel function. Examples include weighted versions of linear and non-linear parametric models. Here we consider only the simplest form, a locally constant model, commonly called the Nadaraya-Watson estimator,

$$\hat{E}(y|\mathbf{X} = \mathbf{x}_0) = \frac{\sum_i^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i) y_i}{\sum_i^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i)}$$

stated here for the case that y is continuous, where the kernel function acts by

$$K_\lambda(\mathbf{x}_0, \mathbf{x}) = K\left(\frac{d(\mathbf{x}_0, \mathbf{x})}{h_\lambda(\mathbf{x}_0)}\right) \tag{2.1}$$

where d is some distance metric and h_λ is a scaling factor, commonly called the *bandwidth*. The analogous version when y is discrete is simply

$$\widehat{\Pr}(g|\mathbf{X} = \mathbf{x}_0) = \frac{\sum_i^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i) \mathcal{I}(y_i = g)}{\sum_i^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i)}.$$

Note that scaling by $\sum_i^n K_\lambda(\mathbf{x}_0, \mathbf{x}_i)$ is somewhat redundant, since this scaling factor could be incorporated into the bandwidth function h_λ.

With the appropriate choice of h_λ and K this definition includes the well known k-nearest neighbor and weighted k nearest neighbor methods.

2.2.2 Optimal metrics

The notion of "closeness" that is embodied in a kernel is largely captured by the distance metric used in 2.1. It is well known in k-nearest neighbor modeling that the choice of distance metric can dramatically affect performance. This has motivated the identification of an optimal distance metric. The central result in this vein is by Cover and Hart [9], where they motivate and derive an optimal distance metric for the 1-nearest neighbor learner in the case of 2-class classification. We review this result here.

Let $\mathbf{x}_{nn}$ denote the closest training point to the independent test point $\mathbf{x}_0$. The *risk* of a 1-nearest neighbor classifier is the probability that we misclassify $\mathbf{x}_0$. We will shorten our notation slightly so that, for example, $\Pr(1|\mathbf{x}_0) = \Pr(Y_0 = 1|\mathbf{x}_0)$ and $\Pr(1|\mathbf{x}_{nn}) = \Pr(Y_{nn} = 1|\mathbf{x}_{nn})$. They begin by substituting the asymptotic risk of the 1-NN rule with its upper bound (see [29],[26] for a derivation of this bound):

$$r^\star(\mathbf{x}_0) = 2\Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0)$$

On the other hand, the *finite* sample risk is given by

$$\begin{aligned}
r(\mathbf{x}_0, \mathbf{x}_{nn}) &= \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_{nn}) + \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_{nn}) + \Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) - \Pr(1|\mathbf{x}_0) + \\
&\qquad \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) - \Pr(1|\mathbf{x}_0) + \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_{nn}) + \\
&\qquad \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(1|\mathbf{x}_0)\left[1 - \Pr(2|\mathbf{x}_0)\right] - \Pr(1|\mathbf{x}_0)\left[1 - \Pr(2|\mathbf{x}_{nn})\right] + \\
&\qquad \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(1|\mathbf{x}_0)^2 - \Pr(1|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) + \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= 2\Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(1|\mathbf{x}_0)^2 - \Pr(1|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) - \\
&\qquad \Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \Pr(2|\mathbf{x}_0)\Pr(1|\mathbf{x}_{nn}) \\
&= 2\Pr(1|\mathbf{x}_0)\Pr(2|\mathbf{x}_0) + \left[\Pr(1|\mathbf{x}_0) - \Pr(2|\mathbf{x}_0)\right] \cdot \left[\Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_{nn})\right]
\end{aligned}$$

In light of these two expressions for $r^\star(\mathbf{x}_0)$ and $r(\mathbf{x}_0, \mathbf{x}_{nn})$, they argue that it makes sense to choose a distance metric that minimizes the difference between the asymptotic and finite sample risk of the 1-nearest neighbor model. The difference is simply

$$r(\mathbf{x}_0, \mathbf{x}_{nn}) - r^\star(\mathbf{x}_0) = [\Pr(1|\mathbf{x}_0) - \Pr(2|\mathbf{x}_0)] \cdot [\Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_{nn})] \, .$$

So for a fixed $\mathbf{x}_0$, minimizing this is equivalent to minimizing $|\Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_{nn})|$ or $[\Pr(1|\mathbf{x}_0) - \Pr(1|\mathbf{x}_{nn})]^2$. Hence, for two class classification, when using the 1-NN learner, we should use as a distance metric the function

$$d_1(\mathbf{x}_1, \mathbf{x}_2) = |\Pr(1|\mathbf{x}_1) - \Pr(1|\mathbf{x}_2)|. \tag{2.2}$$

Arguing by analogy, there are several options for extending this metric to multi-class classification (cf. [26]), among them

$$\begin{aligned} d_1(\mathbf{x}_1, \mathbf{x}_2) &= \sum_{g=1}^{G} |\Pr(g|\mathbf{x}_1) - \Pr(g|\mathbf{x}_2)|, \text{ or} \\ d_1(\mathbf{x}_1, \mathbf{x}_2) &= \sum_{g=1}^{G} \Pr(g|\mathbf{x}_1) \cdot |\Pr(g|\mathbf{x}_1) - \Pr(g|\mathbf{x}_2)|. \end{aligned}$$

Similarly, the analogous version of d_1 for a continuous response would be $d_1(\mathbf{x}_1, \mathbf{x}_2) = |E[y|\mathbf{x}_1] - E[y|\mathbf{x}_2]|$.

The first thing to note is that 2.2 is not, in fact, a metric. Specifically, we may have two points $\mathbf{x}_1 \neq \mathbf{x}_2$ where $d_1(\mathbf{x}_1, \mathbf{x}_2) = 0$, so it fails condition (3) in Definition 2.1.1. Hence d_1 is in fact a pseudometric, which can be converted to a true metric by considering equivalence classes defined by those points where $\Pr(1|\mathbf{x}_1) = \Pr(1|\mathbf{x}_2)$. Additionally, the usefulness of this metric seems limited, since if we *knew* the conditional class probabilities, a simple application of Bayes rule would yield the optimal classifier. Specifically, if we knew $\Pr(g|\mathbf{x}_0)$ for $g = 1, \dots, G$, then we would simply set $\hat{y}_0 = \arg\max_g \Pr(g|\mathbf{x}_0)$.

Intuitively, this metric is telling us that observations that lie on similar *contours* of the conditional expectation of y should be considered “close”. Note that this may or may not correspond to our intuitive notion of distance in the space $\mathcal{X}$. Two points $\mathbf{x}_1$ and $\mathbf{x}_2$ might be very distant in terms of their Euclidean distance, but if the conditional expectation of y at these two points is very close, then in some sense they are very “similar”. Several authors ([11], [19]) have used this result to motivate the development of *local* distance metrics, that create a unique distance metric for each test point $\mathbf{x}_0$ that accounts for the local characteristics of the conditional distribution of y. Usually, this is done through some form of feature weighting. For example, if near $\mathbf{x}_0$ the conditional distribution of y changes rapidly in the x_i direction, but remains relatively constant in the x_j direction, then in the resulting metric the x_i coordinate will receive a large weight and x_j a small weight.

2.3 Tree Ensembles as Kernel Method

2.3.1 Introduction

Here we will characterize all tree ensemble methods as kernel models. Specifically, the predictions from tree ensembles are weighted averages of the training data, and the weights are based upon a distance metric created by the tree ensemble. This basic idea was first noticed by Lin and Jeon [22]; we extend this notion to include *all* tree ensembles rather than just a particular type of random forest model.

Next, we explain how the metric created by tree ensembles is related to the optimal 1-NN metric d_1 described above, and provide a general, heuristic argument that this type of metric is sensible in the context of kernel methods in general.

2.3.2 Partition Based Metrics

Let $\Pi_1, \ldots, \Pi_B$ be a finite number of partitions on the space $\mathcal{X}$, with $\Pi_i = \{R_{1_i}, \ldots, R_{m_i}\}$. Hence in each Π_i, $R_\ell \cap R_j = \emptyset$ for all $\ell \neq j$ and $\bigcup_\ell R_\ell = \mathcal{X}$. Let $A(\mathbf{x}_1, \mathbf{x}_2, \Pi)$ denote the event that $\mathbf{x}_1$ and $\mathbf{x}_2$ lie in the same region of the partition Π. In other words, there exists an $R_\ell \in \Pi$ such that $\mathbf{x}_1, \mathbf{x}_2 \in R_\ell$.

Further, let $Q(\mathbf{x}_1, \mathbf{x}_2)$ be defined as

$$Q(\mathbf{x}_1, \mathbf{x}_2) \quad = \quad \frac{1}{B} \sum_{i=1}^{B} \mathcal{I}(A(\mathbf{x}_1, \mathbf{x}_2, \Pi_i))$$

and let $d_2(\mathbf{x}_1, \mathbf{x}_2) = 1 - Q(\mathbf{x}_1, \mathbf{x}_2)$. So $Q(\mathbf{x}_1, \mathbf{x}_2)$ is simply the proportion of times that $\mathbf{x}_1$ and $\mathbf{x}_2$ lie in the same region over all partitions $\Pi_1, \ldots, \Pi_B$. First we prove that d_2 is a pseudometric.

Proposition 1. *The function $d_2(\mathbf{x}_1, \mathbf{x}_2) = 1 - Q(\mathbf{x}_1, \mathbf{x}_2)$ is a pseudometric.*

Proof. First we observe that $d_2(\mathbf{x}_1, \mathbf{x}_2) \geq 0$ simply by construction. By the properties of a partition and set inclusion, we have that the event $A(\mathbf{x}_1, \mathbf{x}_2, \Pi)$ occurs if and only if the event $A(\mathbf{x}_2, \mathbf{x}_1, \Pi)$ occurs, so we immediately have that $d_2(\mathbf{x}_1, \mathbf{x}_2) = d_2(\mathbf{x}_2, \mathbf{x}_1)$. To demonstrate that d_2 satisfies the triangle inequality, we must show that

$$d_2(\mathbf{x}_1, \mathbf{x}_2) \leq d_2(\mathbf{x}_1, \mathbf{x}_3) + d_2(\mathbf{x}_3, \mathbf{x}_2)$$

which is equivalent to showing that

$$1 - Q(\mathbf{x}_1, \mathbf{x}_2) \leq 1 - Q(\mathbf{x}_1, \mathbf{x}_3) + 1 - Q(\mathbf{x}_3, \mathbf{x}_2)$$

or finally that

$$Q(\mathbf{x}_1, \mathbf{x}_3) + Q(\mathbf{x}_3, \mathbf{x}_2) \leq 1 + Q(\mathbf{x}_1, \mathbf{x}_2).$$

The remaining part of the proof proceeds by induction on the number of partitions, B. First suppose there is only one partition, Π_1. Then the function Q can take only the values 1 or 0. The only way the inequality could be violated is if $Q(\mathbf{x}_1, \mathbf{x}_2) = 0$ but $Q(\mathbf{x}_3, \mathbf{x}_2) = Q(\mathbf{x}_1, \mathbf{x}_3) = 1$. But this would imply that there exists an $R_\ell \in \Pi_1$ such that $\mathbf{x}_3, \mathbf{x}_2 \in R_\ell$ and there exists an $R_j \in \Pi_1$ such that $\mathbf{x}_1, \mathbf{x}_3 \in R_j$. But by the properties of partitions, this would imply that $R_\ell = R_j$, and that $\mathbf{x}_1, \mathbf{x}_2 \in R_j = R_\ell$ which contradicts $Q(\mathbf{x}_1, \mathbf{x}_2) = 0$.

Now suppose that the inequality holds for B partitions, and let Π_{B+1} be another partition. To ease notation, define

$$\begin{aligned} a &= \sum_{i=1}^{B} \mathcal{I}(A(\mathbf{x}_1, \mathbf{x}_3, \Pi_i)) \\ b &= \sum_{i=1}^{B} \mathcal{I}(A(\mathbf{x}_3, \mathbf{x}_2, \Pi_i)) \\ c &= \sum_{i=1}^{B} \mathcal{I}(A(\mathbf{x}_1, \mathbf{x}_2, \Pi_i)) \end{aligned}$$

so that a, b, c simply count the number of times that each pair of points lies in the same region over the first B partitions. Then by our induction hypothesis we have that

$$\begin{aligned} \frac{a}{B} + \frac{b}{B} &\leq 1 + \frac{c}{B}, \\ a + b &\leq B + c. \end{aligned}$$

Now, when we add the partition Π_{B+1}, there are only three possibilities: (1) each of a, b, c is increased by one, (2) precisely one of a, b, c is increased by one and the others remain unchanged, or (3) they all remain unchanged. In the first case we would have

$$\begin{aligned}
\frac{a+1}{B+1}+\frac{b+1}{B+1} &\leq 1+\frac{c+1}{B+1} \\
a+1+b+1 &\leq B+1+c+1 \\
a+b &\leq B+c.
\end{aligned}$$

Similarly, if we assume that (without loss of generality) only a is increased by one we would have

$$\begin{aligned}
\frac{a+1}{B+1}+\frac{b}{B+1} &\leq 1+\frac{c}{B+1} \\
a+1+b &\leq B+1+c \\
a+b &\leq B+c
\end{aligned}$$

Finally, if none of a, b, c are increased by one, we are left with

$$a+b \leq B+1+c$$

and so the expression holds in each case. □

It is easy to note that d_2 may fail (3) from Definition 2.1.1; namely, two points $\mathbf{x}_1 \neq \mathbf{x}_2$ may lie in the same region in every partition Π_i and hence would have $d_2(\mathbf{x}_1, \mathbf{x}_2) = 0$. As noted above, we can create a true metric from d_2 simply by defining equivalence classes consisting of those points whose distance is zero and allowing d_2 to act on the equivalence classes.

Since a decision tree partitions the space $\mathcal{X}$, we have as an immediate corollary to Proposition

1.

Corollary 2.3.1. *For any tree ensemble method the function d_2, defined as above, is a pseudometric (and hence there exists an associated metric), on the feature space $\mathcal{X}$.*

We emphasize that this corollary applies to any tree ensemble, regardless of the depth of the trees, or the nature of their construction. All that is required is that each tree consists of a partition of $\mathcal{X}$.

2.3.3 Tree Ensembles as Kernel Methods

We begin by describing Lin and Jeon's original observation in this vein, in which they considered a restricted class of random forest tree ensembles. Specifically, consider a tree ensemble in which each tree is grown to maximal depth: each leaf of each tree contains only a single training point.

This is slightly awkward in a classification setting, since if a node contains multiple training points all from the same class (i.e. the node is *pure*) we would typically not continue to split that node. So we must make the assumption that pure nodes with more than one training observation will be split, albeit randomly. (How we split pure nodes is irrelevant to the resulting ensemble, since the predictions will remain unchanged.) We can handle the analogous situation for regression in the same manner.

As before, let $\Pi_1, \ldots, \Pi_B$ denote B trees (i.e. partitions of $\mathcal{X}$) grown in this fashion on the training set $(y_i, \mathbf{x}_i)_{i=1}^n$. Then it is easy to see that the ensemble's prediction at a point $\mathbf{x}_0$ is simply a weighted average of the y_i's:

$$\begin{aligned}
\hat{E}(y|\mathbf{x}_0) &= \sum_{i=1}^{n} Q(\mathbf{x}_i, \mathbf{x}_0) y_i \text{ if } y \text{ is a continuous response} && (2.3)\\
\widehat{\Pr}(g|\mathbf{x}_0) &= \sum_{i=1}^{n} Q(\mathbf{x}_i, \mathbf{x}_0) \mathcal{I}(y_i = g) \text{ if } y \text{ is a discrete response} && (2.4)
\end{aligned}$$

where the function Q has the same meaning as in equation 2.3; namely the proportion of times that $\mathbf{x}_i$ and $\mathbf{x}_0$ lie in the same region (terminal node) over all partitions (trees). In addition, we see that these tree ensembles are creating a unique distance metric and then fitting a distance weighted n-nearest neighbor model. In practice, many of the training points receive a weight of 0, so we might say that the ensemble is fitting a distance weighted k-nearest neighbor model, where k (and the metric!) is different for each test point $\mathbf{x}_0$.

Of course, in actual implementations of tree ensembles it is often the case that we do not construct each tree such that every terminal node contains only a single training point. Hence, we might ask what can be said about tree ensembles in general: what happens if we relax the requirement that each terminal node contains only a single training point? The answer will require a simple modification of our definition of Q.

Let us assume again that we have B partitions (trees), $\Pi_1, \ldots, \Pi_B$ and a training set $(y_i, \mathbf{x}_i)_{i=1}^{n}$. We need not have used the training set to create the partitions (trees), Π_i, but there must be at least one training point in every region of every partition. Let $R_i(\mathbf{x}_j)$ denote the region in partition (tree) Π_i that contains the point $\mathbf{x}_j$. Finally, let $|R_i(\mathbf{x}_j)|$ denote the number of *training points* contained in the region $R_i(\mathbf{x}_j)$.

Suppose for the moment that y is continuous and consider a point $\mathbf{x}_0$ not in the training set. We can write the prediction made by the tree ensemble at the point $\mathbf{x}_0$ using the notation above as

$$\hat{y}_0 = \frac{1}{B}\sum_{i=1}^{B} |R_i(\mathbf{x}_0)|^{-1} \left(\sum_{y_j \in R_i(\mathbf{x}_0)} y_j \right). \tag{2.5}$$

So this is simply an average of averages; we average the mean of the y_j's in each node containing $\mathbf{x}_0$. If we rearrange the terms in this sum we find that the weight that each training value y_j receives in the final prediction is simply the sum of the $|R_i(\mathbf{x}_0)|^{-1}$ over the y_j, the reciprocal of the number of training observations in each node containing y_j.

We have used the notation $\mathbf{z}_1, \mathbf{z}_2$ to emphasize that they *may not* be points in the training set. Then we can write equivalent expressions to Equations 2.3 and 2.4 by modifying our definition of Q. Specifically, we define $Q^\star$ as

$$Q^\star(\mathbf{z}_1, \mathbf{z}_2) = \frac{1}{B}\sum_{i=1}^{B} |R_i(\mathbf{z}_1)|^{-1} \mathcal{I}(A(\mathbf{z}_1, \mathbf{z}_2, \Pi_i)). \tag{2.6}$$

Note that $Q^\star$ includes Q as a special case, when $|R_i(\mathbf{z}_j)| = 1$ for all i, j. Additionally, we note that $|R_i(\mathbf{z}_j)|$ *only* counts the *training points* contained in that region of a partition (tree). With this modification, we can say that for *any* tree ensemble,

$$\begin{aligned} \hat{E}(y|\mathbf{x}_0) &= \sum_{i=1}^{n} Q^\star(\mathbf{x}_i, \mathbf{x}_0) y_i \text{ if } y \text{ is a continuous response} && (2.7) \\ \widehat{\Pr}(g|\mathbf{x}_0) &= \sum_{i=1}^{n} Q^\star(\mathbf{x}_i, \mathbf{x}_0) \mathcal{I}(y_i = g) \text{ if } y \text{ is a discrete response} && (2.8) \end{aligned}$$

There are some important differences between Q and $Q^\star$. First, our definition of Q required only a collection of partitions (trees) on the space $\mathcal{X}$; $Q^\star$ requires both a collection of partitions (trees) *and* a training set $(y_i, \mathbf{x}_i)_{i=1}^{n}$. Additionally, we require that each region in each partition contain at least one training point; there can be no "empty" regions. However, we again note that the training points need not have been used to create the partitions (trees).

Next, we can see that the function Q is bounded $0 \leq Q(\mathbf{x}_1, \mathbf{x}_2) \leq 1$, so that the resulting pseudometric is also bounded between 0 and 1. However, when we allow more than one training observation in each region (terminal node) of every partition (tree), the bounds on $Q^\star$ may be different. Specifically, if $\mathbf{z}_1$ and $\mathbf{z}_2$ never lie in the same region (terminal node), then $Q^\star(\mathbf{z}_1, \mathbf{z}_2) = 0$, so the lower bound is the same. However, suppose that $\mathbf{z}_1$ and $\mathbf{z}_2$ *always* lie in the same region (terminal node), but that one of those regions contains more than one training point: $|R_i(\mathbf{z}_1)| > 1$. Then $Q^\star(\mathbf{z}_1, \mathbf{z}_2) < 1$.

To make this observation more concrete, suppose that there are exactly $k > 1$ training observations in each region (terminal node). Then if $\mathbf{z}_1$ and $\mathbf{z}_2$ always lie in the same region (terminal node) we have that $Q^\star(\mathbf{z}_1, \mathbf{z}_2) = 1/k$ so the function $Q^\star$ is bounded $0 \leq Q^\star(\mathbf{z}_1, \mathbf{z}_2) \leq 1/k$. In general, there may be a different number of training points in each region of each partition, so determining universal bounds on $Q^\star$ is in practice unrealistic. However, we can observe that increasing the average number of observations per region (terminal node) will generally cause the values $Q^\star(\mathbf{x}_i, \mathbf{x}_0)$ in (2.8) to be more evenly distributed across the training points $\mathbf{x}_i$. Conversely, decreasing the average number of observations per region (terminal node) will cause the values $Q^\star(\mathbf{x}_i, \mathbf{x}_0)$ to depend on only a handful of training points.

The fact that there may be different numbers of training points in each region of every partition gives the impression that $Q^\star$ is considerably more complex than Q. However, this is not necessarily the case. Assume, as above, that every region of each partition Π contains exactly k training points. Then we can rewrite $Q^\star$ simply as

$$\begin{aligned}
Q^\star(\mathbf{z}_1,\mathbf{z}_2) &= \frac{1}{B}\sum_{i=1}^{B}\frac{1}{|R_i(\mathbf{z}_1)|}\mathcal{I}(A(\mathbf{z}_1,\mathbf{z}_2,\Pi_i)) \\
&= \frac{1}{B}\sum_{i=1}^{B}k^{-1}\mathcal{I}(A(\mathbf{z}_1,\mathbf{z}_2,\Pi_i)) \\
&= \frac{1}{kB}\sum_{i=1}^{B}\mathcal{I}(A(\mathbf{z}_1,\mathbf{z}_2,\Pi_i)) \\
&= k^{-1}Q(\mathbf{z}_1,\mathbf{z}_2)
\end{aligned}$$

and so in this case $Q^\star$ is simply a scaled version of Q. Of course, in practice tree ensembles will only achieve this property approximately: the number of training observations in each region of every partition will only be approximately equal.

If we define $d_2^\star(\mathbf{z}_1,\mathbf{z}_2) = 1 - Q^\star(\mathbf{z}_1,\mathbf{z}_2)$, we can use a similar argument as in Proposition 1 to show that $d_2^\star$ is a pseudometric.

Proposition 2.3.2. *The function $d_2^\star(\mathbf{z}_1,\mathbf{z}_2) = 1 - Q^\star(\mathbf{z}_1,\mathbf{z}_2)$ defined above is a pseudometric.*

Proof. The proof is essentially identical to that in Proposition 1. Properties (1) and (2) from Definition 2.1.1 follow directly from the definition of $d_2^\star$, so we turn to property (4), the triangle inequality. As in Proposition 1, this amounts to showing that

$$Q^\star(\mathbf{z}_1,\mathbf{z}_3) + Q^\star(\mathbf{z}_3,\mathbf{z}_2) \leq 1 + Q^\star(\mathbf{z}_1,\mathbf{z}_2) \tag{2.9}$$

As before, we proceed by induction on the number of partitions B. First suppose that $B = 1$, so there is only a single partition, Π. Then one of the following must be true: (1) none of the $\mathbf{z}_i$ lie in the same region of Π, (2) precisely two of the $\mathbf{z}_i$ lie in the same region of Π or (3) all of the $\mathbf{z}_i$ lie in the same region of Π. In the case of (1), then Equation 2.9 simply reduces to $0 \leq 1$. In the case of (3) we have that $|R(\mathbf{z}_1)| = |R(\mathbf{z}_2)| = |R(\mathbf{z}_3)| = R$, so Equation 2.9

reduces to

$$\frac{1}{R} + \frac{1}{R} \leq 1 + \frac{1}{R}$$

Finally, in the case that (2) holds we simply have (without loss of generality) that $|R(\mathbf{z}_1)|^{-1} \leq 1$. So equation 2.9 holds in all cases.

Now assume that 2.9 holds for B, and consider adding a new partition, Π_{B+1}. Proceeding as in Proposition 1 we set

$$\begin{aligned} a &= \sum_{i=1}^{B} \frac{1}{|R_i(\mathbf{z}_1)|} \mathcal{I}(A(\mathbf{z}_1, \mathbf{z}_3, \Pi_i)) \\ b &= \sum_{i=1}^{B} \frac{1}{|R_i(\mathbf{z}_3)|} \mathcal{I}(A(\mathbf{z}_3, \mathbf{z}_2, \Pi_i)) \\ c &= \sum_{i=1}^{B} \frac{1}{|R_i(\mathbf{z}_1)|} \mathcal{I}(A(\mathbf{z}_1, \mathbf{z}_2, \Pi_i)) \end{aligned}$$

so that our induction hypothesis amounts to assuming that

$$\frac{a}{B} + \frac{b}{B} \leq 1 + \frac{c}{B} \Rightarrow a + b \leq B + c \tag{2.10}$$

Now for the partition Π_{B+1} we again consider the three possible cases listed above. In case (1) equation 2.10 becomes

$$\frac{a}{B+1} + \frac{b}{B+1} \leq 1 + \frac{c}{B+1}$$

which clearly holds. Next, assume we are in case (2). Then we must increase one of a, b or c by $r = |R_{B+1}(\mathbf{z}_i)|^{-1}$, $i = 1, 2, 3$. For example, if $\mathbf{z}_1, \mathbf{z}_3$ lie in the same region then equation 2.10 becomes

$$\frac{a+r}{B+1} + \frac{b}{B+1} \leq 1 + \frac{c}{B+1} \Rightarrow a + b + r \leq B + c + 1$$

which holds since $r < 1$. The other two options in case (2) are similar. Finally, suppose that we are in case (3). Then setting $r = |R_{B+1}(\mathbf{z}_i)|^{-1}$, $i = 1, 2, 3$, since they all lie in the same region, equation 2.10 becomes

$$\frac{a+r}{B+1} + \frac{b+r}{B+1} \leq 1 + \frac{c+r}{B+1} \Rightarrow a + b + r \leq B + c + 1$$

which holds since $r < 1$. □

It is interesting to note that the ways in which $d_2^\star$ can fail the identifiability condition (3) for metrics distinguish it from d_2. We observed before that with d_2 we may easily have two points $\mathbf{z}_1 \neq \mathbf{z}_2$ that nevertheless fall in the same terminal node in every partition and hence have $d_2(\mathbf{z}_1, \mathbf{z}_2) = 0$. This can happen with $d_2^\star$ as well; however, we may be unable to achieve a distance of 0 at all, even when $\mathbf{z}_1 = \mathbf{z}_2$!

This will be the case in our previous example where each region contains exactly $k > 1$ training points. This means that if $\mathbf{z}_1 = \mathbf{z}_2$ that $Q^\star(\mathbf{z}_1, \mathbf{z}_2) = 1/k$ so $d_2^\star(\mathbf{z}_1, \mathbf{z}_2) = 1 - 1/k > 0$. This makes converting $d_2^\star$ into a true metric somewhat more complicated. Specifically, simply creating equivalence classes defined by $\mathbf{z}_1 \sim \mathbf{z}_2 \Leftrightarrow d_2^\star(\mathbf{z}_1, \mathbf{z}_2)$ will not work since it may be that *no* pairs of points have distance 0. In the case where each region has exactly $k > 1$ training points we can simply define $d_2^\star(\mathbf{z}_1, \mathbf{z}_2) = 1/k - Q^\star(\mathbf{z}_1, \mathbf{z}_2)$; however, there is not a similarly simple solution for the case when we allow different numbers of training points in each region.

2.3.4 d_1 is Optimal for Kernel Methods Generally

Here we provide a heuristic argument that the metric d_1 is in some sense the optimal notion of distance for kernel methods in general. First we examine the regression case, so y is continuous. Consider estimators of the form

$$\hat{E}(y|\mathbf{X} = \mathbf{x}_0) = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}$$

i.e. kernel method estimators. It is natural to ask how we should choose the values w_i to most efficiently estimate the conditional expectation of y. An obvious answer is that we should choose the w_i such that

$$w_i = \begin{cases} 1 & E(y|\mathbf{X} = \mathbf{x}_0) = E(y|\mathbf{X} = \mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

In other words, the "best" estimate of $E(y|\mathbf{X} = \mathbf{x}_0)$ would be to average only those training observations where the conditional expectation remains unchanged. Then as long as $1/n \sum_i w_i \to 1$ as $n \to \infty$ we will have that $\hat{E}(y|\mathbf{X} = \mathbf{x}_0) \to E(y|\mathbf{X} = \mathbf{x}_0)$. (All this means is that as $n \to \infty$, the number of training points actually included in the estimate at $\mathbf{x}_0$ must also grow but not at the same rate.) This is optimal in the sense that it is unbiased and has minimum variance, since the only variability we will see is that of the true conditional expectation at $\mathbf{x}_0$. An identical argument can be made in the classification case.

This argument is meant only to emphasize that for kernel type estimators, it is sensible to make the weights w_i *large* when the conditional distributions $E(y|\mathbf{x}_0)$ (or $\Pr(g|\mathbf{x}_0)$) and $E(y|\mathbf{x}_i)$ (or $\Pr(g|\mathbf{x}_i)$) are close and w_i should be *small* when these conditional expectations are distant.

2.3.5 Connection to Optimal Metric

All kernel methods begin with the implicit assumption that the conditional distribution of y can be well approximated by a locally constant function, which is a reasonable assumption if the underlying conditional distribution is sufficiently smooth. The optimality of metrics like d_1 for the 1-NN rule can be seen then as a natural consequence of this assumption, as it tells us to look for the nearest neighbor in those regions near $\mathbf{x}_0$ where the conditional distribution of y is constant.

The connection to tree ensemble methods should now be clear. Recall how the greedy algorithm acts in the construction of each individual tree: each split is part of a search for a partition that consists of regions where the conditional distribution of y is as close to being constant as possible. Hence when two points $\mathbf{x}_1$ and $\mathbf{x}_2$ lie in the same region of a partition, that is a rough indicator that the conditional distributions at those points are likely very similar. Hence, over the entire ensemble, the more often this happens, the “closer” these two points should be.

2.4 A Look Ahead

In this chapter we reviewed Lin and Jeon’s [22] observation that tree ensembles grown such that there is only one training observation in each terminal node (partition region) are actually a kernel method. Specifically they fit a weighted average of the training points where the weights are given by the function $Q(\mathbf{z}_1, \mathbf{z}_2)$. Next we generalized this observation to include arbitrary tree ensembles, where the terminal nodes of each tree (partition regions) can contain any number (≥ 1) of training observations and in this case the weights were given by the function $Q^\star(\mathbf{z}_1, \mathbf{z}_2)$.

We observed that both situations lead to pseudometrics (and hence metrics) of the form $1-Q$ or $1-Q^\star$, although converting $1-Q^\star$ to a true metric can be awkward due to the way in which it fails the identifiability condition for metrics. Of course, it is still possible to perform this conversion. We can simply define an equivalence relation by saying that $\mathbf{x}_1 \sim \mathbf{x}_2$ precisely when they lie in the same region of every partition and then use this equivalence relation to convert $d_2^\star$ into a metric. We just cannot use a convenient numerical condition on $d_2^\star$ to define this equivalence relation. However, in many circumstances we can consider $1-Q$ as a simple approximation of $1-Q^\star$. In subsequent chapters this observation will allow us to focus on the much simpler task of calculating $1-Q$. We have been careful thus far to refer to d_1 and d_2 as pseudometrics, with the understanding that they can (in general) be used to define a valid distance metric. In what follows we will drop this formality and refer to d_1 and d_2 simply as metrics.

In the next chapter we will examine the role that randomization plays in the metric generated by d_1.

Chapter 3

Variable Randomness in Stump Ensembles

3.1 Introduction

As discussed in Chapter 1, different types of tree ensembles employ different amounts of randomness. For example, bagging builds trees on distinct bootstrap samples; random forests add the additional step of randomly selecting a subset of covariates at each node to search over for potential splits; completely random decision trees perform no bootstrapping, but split each node completely at random. This raises the question of what all this randomness is accomplishing. Here we will examine this question in light of our discussion in Chapter 2, where we established that tree ensembles act as a kernel method by generating a distance metric. Specifically, we will ask what varying degrees of randomization do to the resulting distance metric.

Typically, tree ensembles are too complex to allow a direct analytical treatment, so we will examine the role of randomness in a simplified tree ensemble model. We will argue that the level of randomness used in the ensemble influences how closely the resulting contours adapt themselves to the local conditional distribution. Specifically, extreme randomness will generally cause the contours of the metric to spread evenly in all directions from a given point. Additionally, we will argue that complete randomness in tree ensembles essentially recreates a nearest neighbor method using the L_1 distance and hence that extreme randomness may not be helpful.

We will focus our attention on the function $Q(\mathbf{x}_1, \mathbf{x}_2)$ rather than $Q^\star(\mathbf{x}_1, \mathbf{x}_2)$. The reasons are twofold: first, since $Q(\mathbf{x}_1, \mathbf{x}_2)$ is easily interpreted as the probability that two points lie in the same region it is easier to understand and to calculate. Second, as we argued in Section 2.3.3, under many circumstances these functions differ (approximately) only by a constant factor, so there is little lost in considering the simpler of the two.

3.2 Stump Ensembles

The simplest type of tree ensemble employs the simplest partition: a single binary partition. A tree that consists of only a single split is called a *stump*, so we will refer to this method as an ensemble of stumps. In general, we can partition the feature space $\mathcal{X}$ however we please. However, in practice it is convenient to have the range of possible partitions depend in some way on the data. Therefore, let $(\mathbf{x}_i, y_i)_{i=1}^n$ denote a training sample of size n where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$. Let $\mathbf{x}_0$ denote an independent test point. A *stump* consists of a single binary partition of the training sample of the form,

$$f(\mathbf{x}_0) = \begin{cases} c_1 & x_{0j} < x_{ij} \\ c_2 & x_{0j} \geq x_{ij}. \end{cases}$$

where x_{0j} is the jth component of the vector $\mathbf{x}_0$ and c_1, c_2 are formed either by the average or majority vote of the training observations that satisfy the corresponding condition.

The value x_{ij} is called a *split point*. Each training point $\mathbf{x}_i$ is a p-vector and each coordinate of $\mathbf{x}_i$ yields a unique stump. Hence, given a training set there are $p(n-1)$ possible stumps. (We make two mild assumptions here: first, we assume that all of the x_{ij} are distinct, so that there really are $p(n-1)$ unique split points. Second, we require that at least one data point fall in each half of every stump.)

Different ensemble creation techniques will lead us to select different combinations of stumps. For example, a completely random stump ensemble would select stumps randomly with replacement from the $p(n-1)$ possible stumps. Other techniques will lead us to select some stumps more than others.

A *stump ensemble* is formed by generating multiple stumps and then combining them either by averaging their predictions (continuous y) or by majority vote (discrete y). We will focus not on the performance of stump ensembles as a statistical learner (which is undoubtedly poor) but on the characteristics of the resulting metric via the function $Q(\mathbf{x}_1, \mathbf{x}_2)$.

3.3 Completely Random Stump Ensembles

Let $f(\mathbf{x})$ be a distribution function on the p-dimensional unit box $[-0.5, 0.5]^p$, let $\{\mathbf{x}_i\}_{i=1}^n$ be a random sample from the distribution f and let $\mathbf{z}_1, \mathbf{z}_2$ be two additional points arising from the distribution f.

In asking what role randomness is playing in tree ensembles, a convenient place to start is the extreme example of total randomness. Therefore, let $\Pi_1, \ldots, \Pi_B$ be stumps (as defined above) chosen randomly, with replacement, from among the $n(p-1)$ available given our training sample.

The function $Q(\mathbf{z}_1, \mathbf{z}_2)$ represents the proportion of times (out of B) that these points lie in the same region of a stump, Π_i. Hence $1 - Q(\mathbf{z}_1, \mathbf{z}_2)$ is simply the proportion of times these points are *separated* over all partitions Π_i. For the stump defined by the split point x_{ij} to separate the points $\mathbf{z}_1, \mathbf{z}_2$ we must have that $z_{1j} < x_{ij} < z_{2j}$ (assuming without loss of generality that $z_{1j} < z_{2j}$). The probability that such a stump exists to be chosen in the ensemble depends in a simple fashion on the distribution f that generated the data. Indeed, the probability that some point in our training sample has its jth component falling between z_{1j} and z_{2j} is simply $\int_{z_{1j}}^{z_{2j}} f_j(x_j)dx_j$, where f_j is simply the marginal distribution of the jth component of $\mathbf{x}$. This leads naturally to the following expression for $1 - Q(\mathbf{z}_1, \mathbf{z}_2)$, which holds as $B \to \infty$ and $n \to \infty$, the number of training points and the number of stumps (partitions) grows,

$$\begin{aligned} 1 - Q(\mathbf{z}_1, \mathbf{z}_2) &= \frac{1}{p}\sum_{j=1}^{p}\int_{z_{1j}}^{z_{2j}} f_j(x)dx_j \\ &= \frac{1}{p}\sum_{j=1}^{p} |F_j(z_{1j}) - F_j(z_{2j})|. \end{aligned}$$

Note that the region in the hyperrectangle defined by the points $\mathbf{z}_1, \mathbf{z}_2$ is in fact being counted p times. This is due to the fact that a training point in that hyperrectangle can contribute p potential partitions that can split these points, one for each coordinate.

If we impose a particular distribution on f we can see the potential limitations of complete randomness in tree ensembles. If we assume that f is uniform over $[-0.5, 0.5]^p$ then the above expression reduces to $1 - Q(\mathbf{z}_1, \mathbf{z}_2) = \frac{1}{p}\sum_{j=1}^{p} |z_{1j} - z_{2j}|$ which is simply the L_1 Euclidean distance (scaled by p to lie between 0 and 1). Hence completely random stump ensembles simply mimic a kernel model based on the L_1 metric.

More generally, this metric essentially computes distances between points $\mathbf{z}_1, \mathbf{z}_2$ that are proportional to the value of the cumulative distribution function along each coordinate direction between these points. Intuitively, the more training data points that are likely to fall "between" $\mathbf{z}_1, \mathbf{z}_2$, the farther apart they are.

3.4 Variable Randomness Stump Ensembles

One possible way to decrease the level of randomness in our stump ensemble model we must introduce a way to evaluate the "goodness" of each potential stump using a score function. A stump ensemble completely lacking in randomness (a deterministic stump ensemble) would simply choose the "best" stump at every turn, and hence would essentially consist of only one stump. In between these two extremes we can tie the selection of stumps in the ensemble to their scores to varying degrees.

First we introduce some additional notation for our notion of stump ensembles discussed above. Let $X = (x_{ij})$ be the $n \times p$ covariate matrix and let $X^{'} = (x_{(i)j})$ denote the $(n-1) \times p$ matrix obtained by sorting the columns of X and removing the final row. Then the elements of $X^{'}$ are the $(n-1)p$ possible split points for the stump ensemble. In particular, we will use $x_{(i)j}$ to refer to specific stumps: $f_{x_{(i)j}}$ is the stump obtained by splitting on the value $x_{(i)j}$. Now define the function $\delta_{ij}(\mathbf{x}_0)$ as follows:

$$\delta_{ij}(\mathbf{x}_0) = \begin{cases} 1 & x_{0j} > x_{(i)j} \\ -1 & x_{0j} \leq x_{(i)j} \end{cases}$$

for $i = 1, \ldots, n-1$ and $j = 1, \ldots, p$. Finally, let w_{ij} denote the probability that the stump $f_{x_{(i)j}}$ is selected for inclusion in the ensemble when sampling with replacement from all the possible at stumps. Then the probability that two independent test points $\mathbf{z}_1$ and $\mathbf{z}_2$ lie in the same region across the entire ensemble is

$$\Pr(\mathbf{z}_1, \mathbf{z}_2 \text{ lie in the same region}) = \sum_i \sum_j w_{ij} \mathcal{I}(\delta_{ij}(\mathbf{z}_1) = \delta_{ij}(\mathbf{z}_2)).$$

We will vary the levels of randomness by assigning values to the w_{ij}. Let $G(x_{(i)j})$ be a score function that assigns a value to each possible stump. For example, for regression data G might be mean squared error and for classification data G might be the Gini index. Let $g^* = \max_{i,j} G(x_{(i)j})$ and let $g_{ij} = G(x_{(i)j})$. Set $w_{ij} = \phi(g_{ij}|g^*, \sigma)$ where ϕ is the pdf of a normal distribution with mean g^* and standard deviation σ. The fixed values g_{ij} are calculated first and then the values w_{ij} are obtained by evaluating ϕ at the values g_{ij} as described above. The values w_{ij} are then scaled to ensure they lie between 0 and 1 to ensure that they are probabilities.

By choosing σ to be very small we approach the completely deterministic case by focusing more heavily on stumps with high scores; by choosing σ to be large we spread the probability of selection into the ensemble more evenly across all possible stumps, approaching the completely random case. Our interest is in how the metric $1 - Q(\mathbf{z}_1, \mathbf{z}_2)$ changes as we vary σ. To do this we will examine the contours of this metric empirically. Specifically, we will look at the contours defined by $C = \{\mathbf{z} | c = Q(\mathbf{z}_0, \mathbf{z})\}$ for a fixed point $\mathbf{z}_0$. This is the set of points that are the same "distance" from $\mathbf{z}_0$ as defined by the proportion of times they are separated by the stumps.

Our discussion above suggests that when σ is large (and the data are generated uniformly) we should expect these contours to resemble those of the L_1 metric, namely to be diamond shaped around the point $\mathbf{z}_0$. As σ becomes smaller, the contours will reflect an increasing emphasis on the "good" stumps (as indicated by their scores). Between these two extremes, the metric should more closely adapt itself to the local class conditional distribution near $\mathbf{z}_0$.

Figure 3.1 contains a training set ($n = 500$) from the XOR data in the *mlbench* package in R. We constructed four stump ensembles on these data using $\sigma = 0.05, 0.5, 1, 10$. Next we generated a test set that consisted of a grid of points over the same domain as the XOR data. We fixed one point in this grid to be $\mathbf{z}_0$ and calculated $Q(\mathbf{z}_0, \mathbf{z}_i)$ for each of the remaining $\mathbf{z}_i$ in the grid. We used these values to construct contour plots of $Q(\mathbf{z}_0, \mathbf{z})$ relative to the point $\mathbf{z}_0$. These are shown in Figure 3.2.

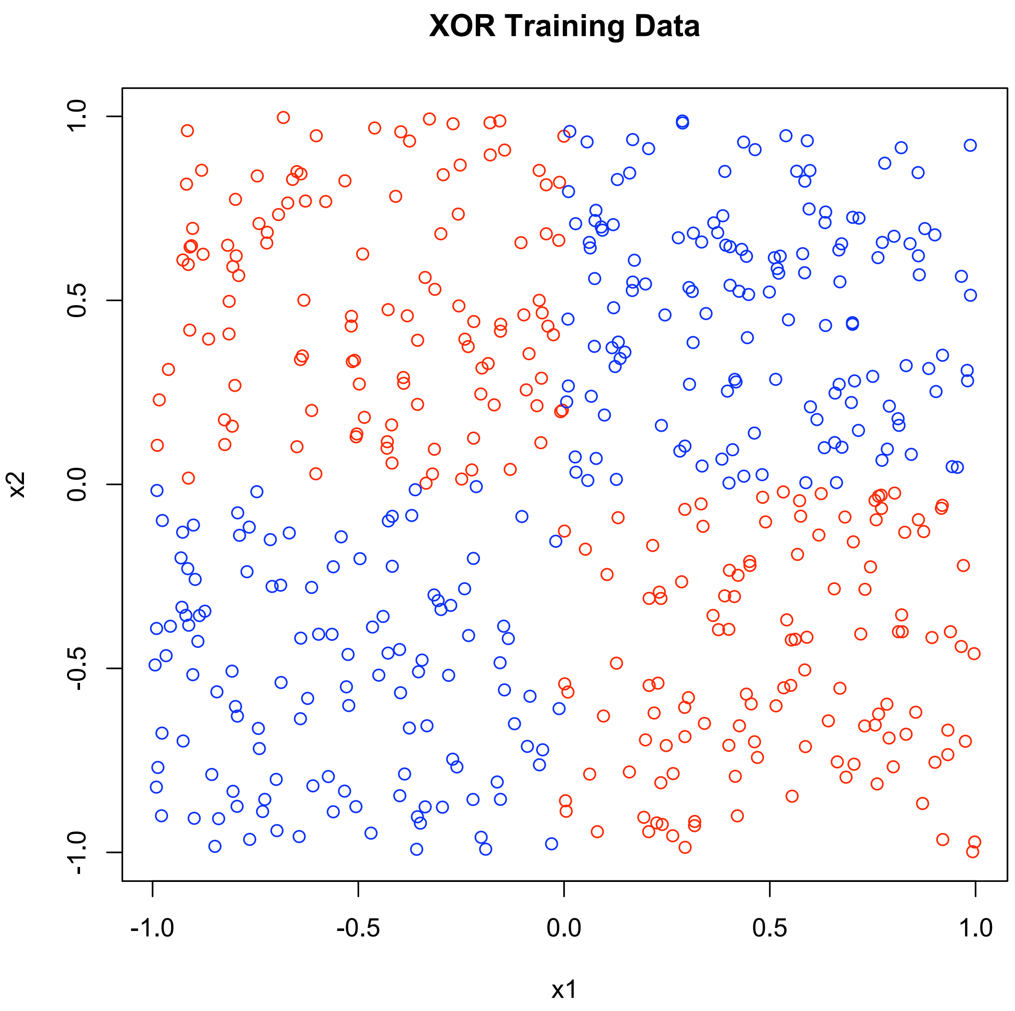


Figure 3.1: Training set ($n = 500$) used to construct stump ensembles.

The contours in Figure 3.2 represent the proportion of times each point lies in the same half of a stump in each ensemble as the point $\mathbf{z}_0$. Hence, the 0.9 contour represents points that are in the same half of the stumps of an ensemble 90% of the time, so the 0.9 contour constitutes

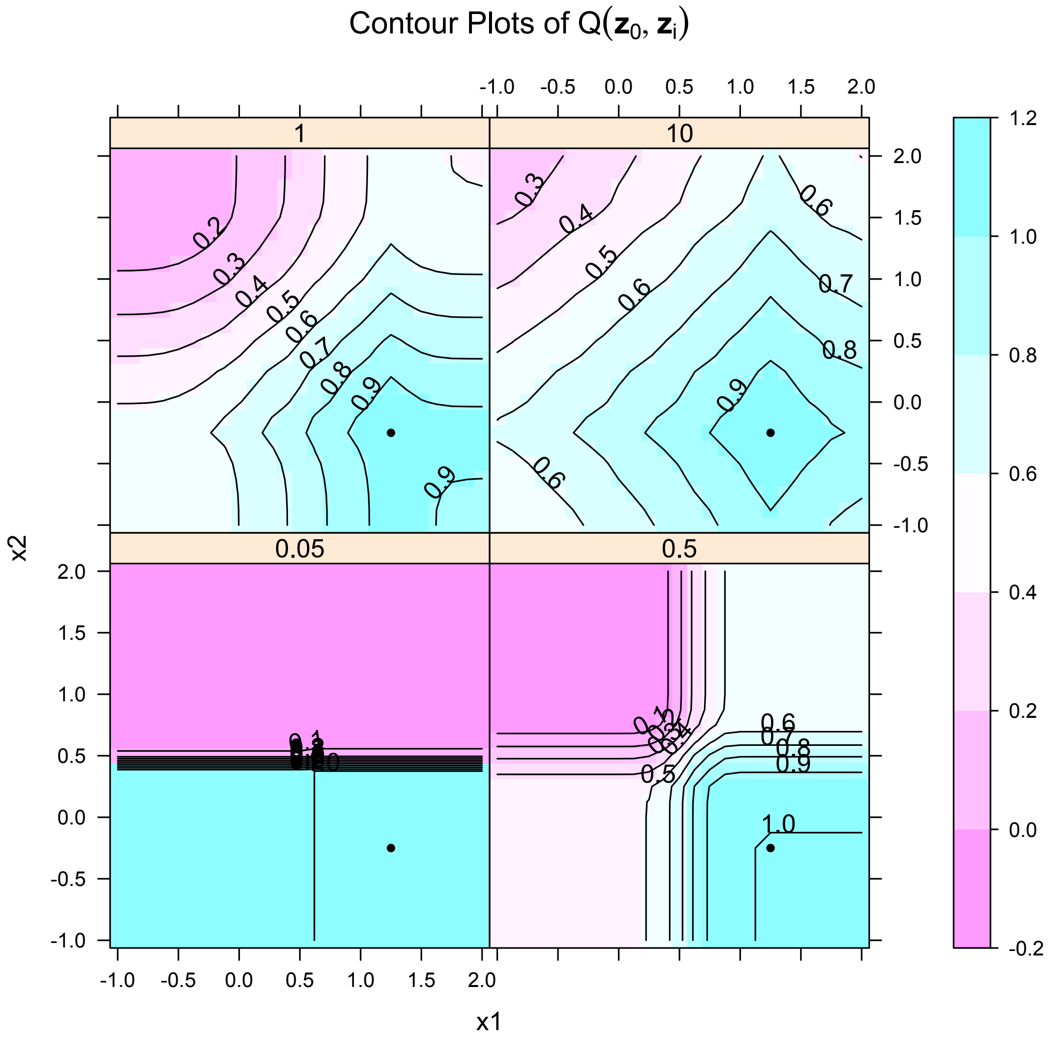


Figure 3.2: Contour plots of $Q(\mathbf{z}_0, \mathbf{z})$. Each panel is labeled according to the value of σ used in the ensemble. The point $\mathbf{z}_0$ is indicated as the black dot in the lower right of each panel.

points that are close to $\mathbf{z}_0$ and the 0.1 contour represents points that are far from $\mathbf{z}_0$.

Note that for $\sigma = 10$ in Figure 3.2 the contours indeed resemble those we would obtain from the L_1 metric, as they are roughly diamond shaped and even spaced away from the point $\mathbf{z}_0$. When $\sigma = 0.05$, the stump ensemble consists almost entirely of stumps that split the data on x_2 near 0.5. In this sense the ensemble is essentially acting as a single stump. With intermediate amounts of randomness ($\sigma = 0.5, 1$) the ensemble is more adaptive to the local structure of the two classes by identifying only the points in the lower right as being close to $\mathbf{z}_0$.

3.5 Conclusion

In this chapter we have investigated the effect of varying amounts of randomness in tree ensembles. Since full tree ensembles pose significant obstacles to direct analytical treatment, we examined the simpler method of stump ensembles. We argued that extreme randomness is not necessarily beneficial, as in the case of stump ensembles complete randomness simply recreates a kernel method using the L_1 metric. Subsequently, we argued empirically that moderate amounts of randomness aids performance by allowing the distance metric $1 - Q(\mathbf{z}_i, \mathbf{z}_j)$ to adapt itself to the local structure of the conditional density of the response y.

An important question is whether these metrics must be estimated via a randomized tree ensemble, or whether they can be calculated directly. We have considered this question in some depth but could not achieve any meaningful results. In general, the precise metric one obtains will be different depending on the particular randomization strategy used. Additionally, the extreme non-linearity inherent in decision trees makes a theoretical analysis challenging. Thus, it remains an open question whether there is a deterministic route to calculating the values of the functions d_2 or $d_2^{\star}$.

Chapter 4

Local Models Using Tree Ensemble Weights

4.1 Introduction

The previous chapters developed the idea that tree ensembles are fitting weighted averages of the training points, where the weights are determined by a particular locally adaptive distance metric. This means that the predicted value of a tree ensemble at $\mathbf{x}_0$ (for a continuous response variable y) is

$$\hat{y}_0 = \sum_{i=1}^{n} Q(\mathbf{x}_0, \mathbf{x}_i) y_i.$$

The form of this model is simply that of a locally constant model. But this naturally leads us to ask if we might use these weights in some other fashion. Given the weights (or distances, if you prefer) $Q(\mathbf{x}_0, \mathbf{x}_i)$, we might apply them to any model that accepts weights.

In this chapter we briefly explore the possible benefits to using the distance metric $Q(\mathbf{x}_0, \mathbf{x}_i)$ as weights in locally linear models. By this we simply mean fitting weighted linear regression models using the values $Q(\mathbf{x}_0, \mathbf{x}_i)$ as weights.

4.2 Fitting Local Models

Consider training data that conform to a traditional regression setting, $(\mathbf{y}, \mathbf{X})$ and a corresponding test point $(y_0, \mathbf{x}_0)$. A standard linear model using these data has the form $\mathbf{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$ where the errors are typically assumed to be independent and normally distributed. The coefficients are found using least squares with the familiar formula $\boldsymbol{\hat{\beta} = (X'X)^{-1}X'y}$. This model can be altered to become a local mode by adding weights to each of the training points. In particular, given a diagonal weight matrix $\mathbf{W}$, the coefficient vector is now found via $\boldsymbol{\hat{\beta} = (X'WX)^{-1}X'Wy}$.

We propose using the tree ensemble distances $Q(\mathbf{x}_0, \mathbf{x}_i)$ as the diagonal elements of the matrix $\mathbf{W}$. To accomplish this we use the R implementation of Leo Breiman's RandomForest software. The R function `randomForest`[1] allows us to estimate the values $Q(\mathbf{x}_0, \mathbf{x}_i)$ using only the out-of-bag samples. This means that the function returns a matrix of values that represent the proportion of times that the two observations land in the same terminal node, and that these proportions are estimated using only those trees for which this pair of observations are both "out-of-bag", that is not included in the bootstrap sample for that tree. Once we have the values $Q(\mathbf{x}_0, \mathbf{x}_i)$ to be used in the matrix $\mathbf{W}$ we estimate the coefficients $\boldsymbol{\hat{\beta}}$ and use them to predict the value y_0 at $\mathbf{x}_0$. This process is repeated for each test point, so a distinct set of coefficients $\boldsymbol{\hat{\beta}}$ is estimated for each test point.

One issue that arises in implementing this procedure is that tree ensembles are often used in situations where the number of covariates is very large. While the tree ensemble easily

[1]`http://cran.r-project.org/web/packages/randomForest/index.html`

handles large numbers of covariates, traditional linear models can have difficulties, specifically in inverting the matrix $\mathbf{X}'\mathbf{X}$. Generally, the solution to this problem is some form of regularization. Instead, we will utilize another feature of the `randomForest` that measures the local variable importance for each training point. What this means is that for each training point we examine the trees in the ensemble for which that observation is out-of-bag. Next we pick a variable, say the mth, and permute its values (i.e. permute that column in the matrix $\mathbf{X}$). Then we run our training observation down each of the trees and record the number of votes for the correct class (classification) or the mean squared error (regression). This is repeated over several permutations of the mth variable and the results averaged. The difference between this and the corresponding value for the un-permuted version of $\mathbf{X}$ is the importance of variable m on this training observation.

We can estimate the local variable importance for our independent test point $\mathbf{x}_0$ by combining the local variable importance values of the training points with their proximities to the test point $\mathbf{x}_0$. Let $\mathbf{p}$ be the vector of proximities of the test point to each of the training points as estimated by the tree ensemble and let $\mathbf{L}$ be the matrix of local variable importance scores for the training observations. The columns of $\mathbf{L}$ correspond to the training observations and the rows to variables. Take the point-wise product of each row of $\mathbf{L}$ with the vector $\mathbf{p}$ and sum along the rows of the resulting matrix. This vector is the local variable importance for the independent test point. The rationale here is that the local variable importance at our test point is estimated by the (weighted) average of the local variable importance at training points *close* to our test point. We can now use this vector to select only the m most relevant covariates to use when we fit our local linear model. Specifically, we would only use the m columns of $\mathbf{X}$ corresponding to the m highest local variable importance values for the test point.

This procedure actually allows us to significantly enlarge the number of covariates we can consider. Since the tree ensemble method does not suffer from over-fitting problems when we include large numbers of variables, we can add additional transformations of our original variables to the tree ensemble model, and only those that are locally relevant at a particular test point will be selected for inclusion in our linear model. (We must be still be cautious regarding collinearity issues though; this procedure may very well select both $\mathbf{x}$ and $\log \mathbf{x}_i$ as locally relevant which will likely cause problems when fitting the linear model.) For simplicity, we will only consider adding the corresponding quadratic terms here, although in principle we might consider including interactions and other more complicated transformations. The only modification to our procedure needed is that we included the square of each of the original covariates when constructing our tree ensemble model. From there everything proceeds normally. This model fitting procedure can be summarized as follows,

- Generate a tree ensemble using the training data.
- For each test point requiring a prediction, use the (out-of-bag) proximities and local variable importances from the tree ensemble to fit a locally linear/quadratic regression model.
- Use this local model to make a prediction at our test point.

Next we demonstrate the potential benefits of adopting such locally linear or quadratic models using several simulated and real datasets. Finally, we will demonstrate how fitting local models can serve as a tool for visualizing models of multivariate data.

4.3 Numerical Study

For the simulated data sets we selected three classification and three regression from the `mlbench` package in R. Additionally, we selected four real data sets, two regression problems and two classification problems. These are all summarized in Table 4.1.

For the simulated data sets, training and test samples of size 200 were generated. A random forest model was fit to the training data using the `randomForest` function in R. Then using the resulting proximity matrix and local variable importance measures for the test set, we fit a locally linear/quadratic regression model to each test point in turn. This process was repeated 30 times and the results averaged.

For the real data sets, we used a leave-one-out cross-validation strategy rather than an independent test set. We first fit a random forest model to the entire data set, yielding proximity and local variable importance matrices based upon the out-of-bag data. Then for each observation we fit a locally linear/quadratic regression model to the remaining observations and use this model to make a prediction at the omitted point. Since there is considerable randomness in this process (forming the tree ensemble, which provides us the proximity and local variable importance) we also repeat this procedure 30 times and average the results. (Our implementation in R generally took several minutes to run for a single data set.)

Table 4.1: Dataset summaries. Twonorm, Threenorm, Circle and Friedman 1, 2 and 3 are the simulated data sets, the remaining are real data sets.

Dataset	Sample Size	Dimension	Classes
Twonorm	200	20	2
Threenorm	200	20	2
Circle	200	20	2
Ionosphere	351	34	2
Sonar	208	60	2
Friedman 1	200	10	
Friedman 2	200	4	
Friedman 3	200	4	
Boston Housing	506	14	
Concrete	1030	9	

4.3.1 Results

The results are summarized in Table 4.2. Table 4.2 suggests that the local regression models using the RF proximities as weights do provide some improvement in accuracy, although not always. There appears to be greater advantage to using this method with regression problems rather than with classification problems. Figures 4.1, 4.2, 4.3 and 4.4 display boxplots for a selection of the simulation results from Table 4.2.

With the exception of the Twonorm data, the locally linear (or quadratic) models all perform better than the standard RF model. There is no clear pattern to recommend locally linear over locally quadratic models, as their performance differs between data sets. In general, the local models are improving performance in regression problems more than classification, and seem to perform better on the real data sets than on the simulated ones.

Table 4.2: Mean out-of-bag error rates for random forests (RF) using its default settings and weighted linear/quadratic models using weights derived from the RF model.

Dataset	Local Linear	Local Quadratic	RF
Twonorm	0.1108	-	0.0473
Threenorm	0.1762	-	0.1710
Circle	0.1678	0.1379	0.1680
Ionosphere	-	0.0595	0.0751
Sonar	-	0.1196	0.1531
Friedman 1	4.64	-	7.08
Friedman 2	19134	17725	21625
Friedman 3	0.0187	0.0241	0.0204
BostonHousing	7.31	-	9.72
Concrete	-	21.07	26.42

4.4 Local Models As Visualization Tool

Finally we consider how these local models can be used as data exploration and visualization tools. First we will examine a simple simulated example and then a real data set. The simulated data we begin with is used simply to illustrate that our method is correctly identifying features of the data when we know the true data generation mechanism. Generally, the method we are introducing would be more useful in situations where we are presented with

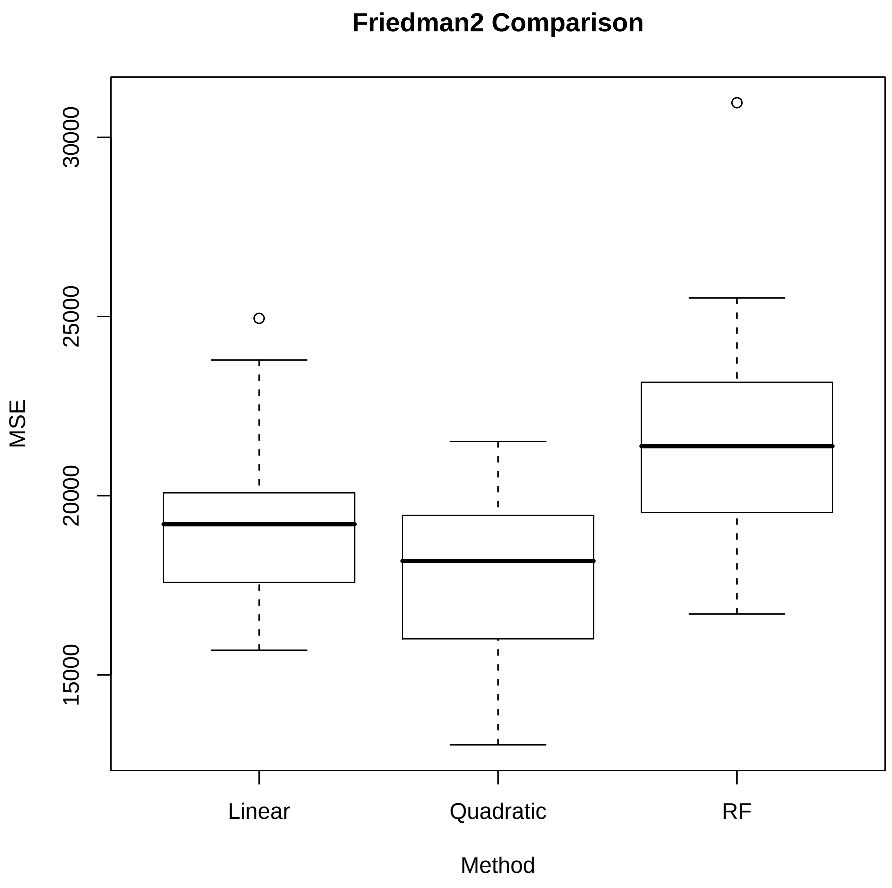


Figure 4.1: Boxplots of errors for simulation on the Friedman 2 dataset.

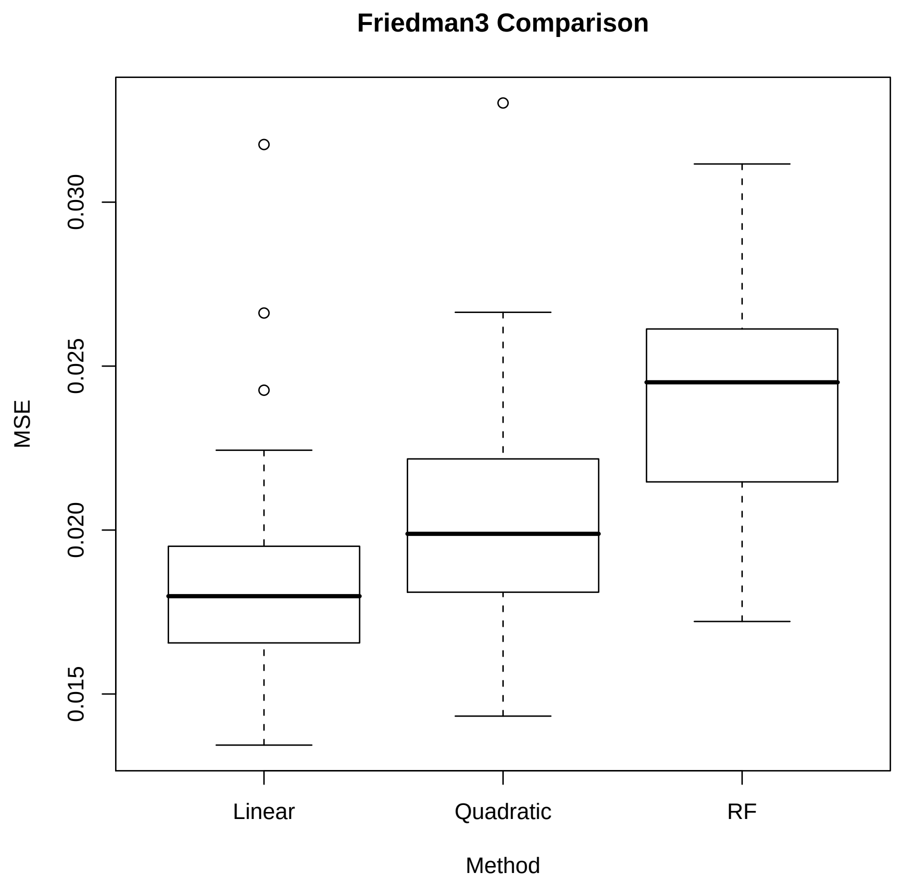


Figure 4.2: Boxplots of errors for simluation on the Friedman 3 dataset.

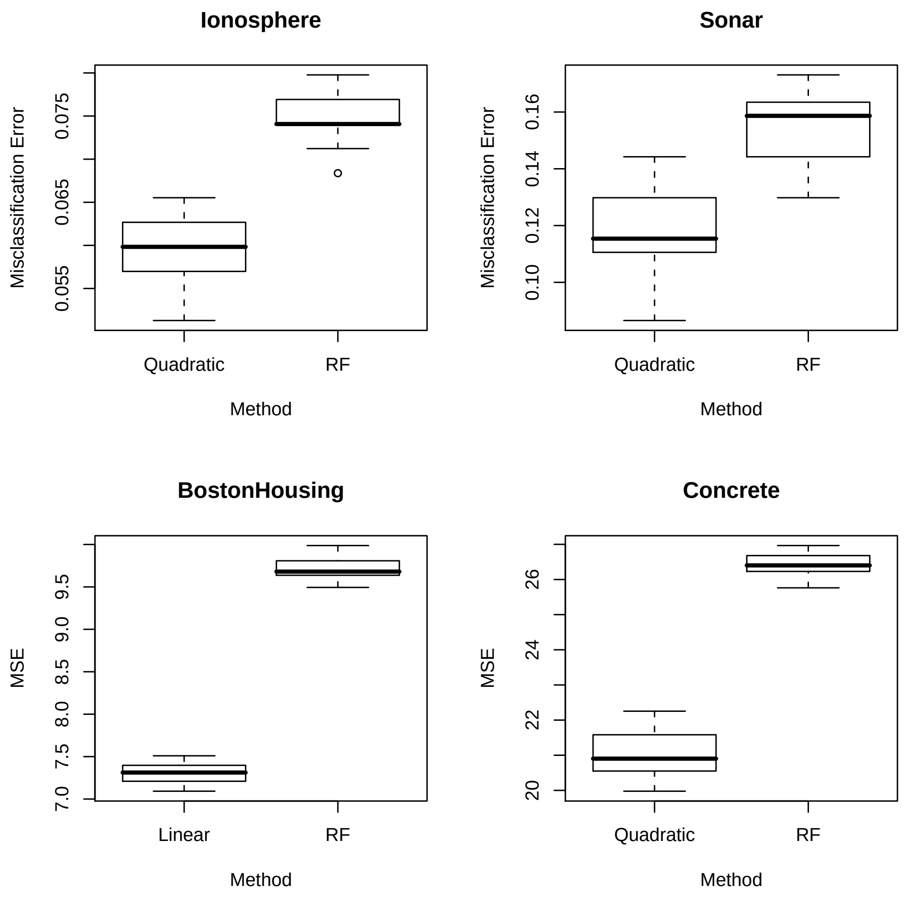


Figure 4.3: Boxplots of errors for the real data sets.

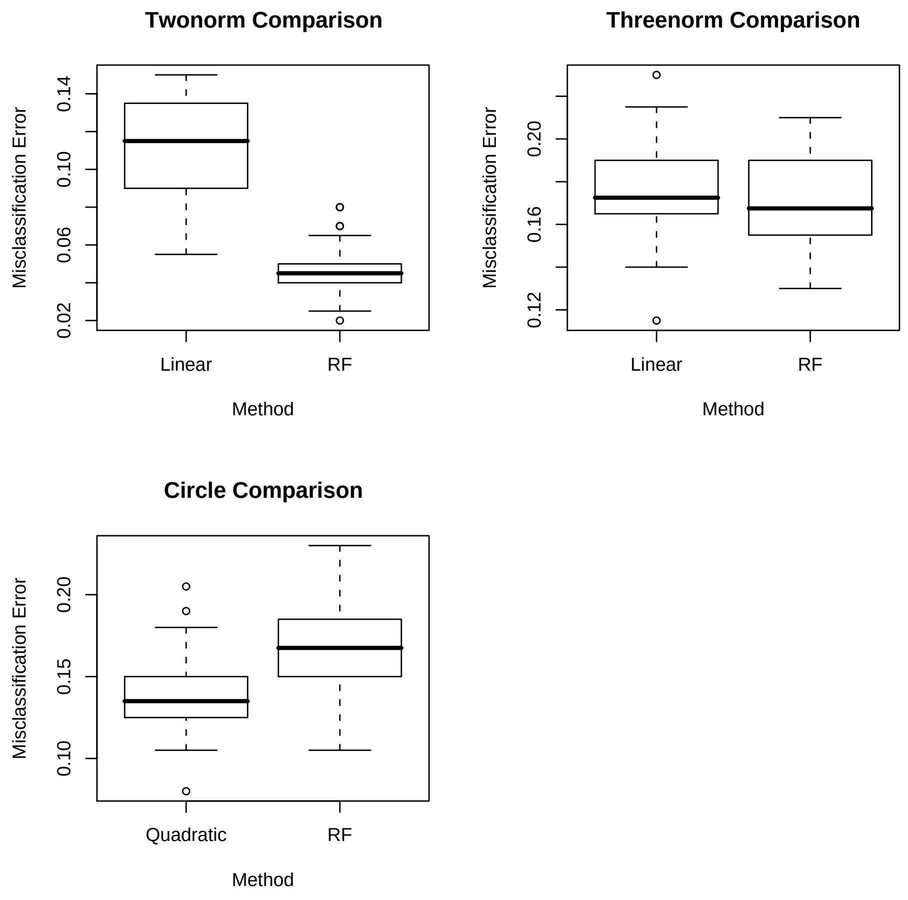


Figure 4.4: Boxplots of the errors for the simulated data sets.

large numbers of covariates and little *a priori* information regarding the functional relationship generating the data. The examples in this section are meant for illustrative purposes only; similar results and conclusions could likely be derived using standard methods (i.e. linear regression models). In Chapter 5 we will examine a case where the dimensionality and complexity of the data make standard approaches more difficult.

We generated 1000 observations from some unknown model (to be revealed below) that we divide into a training and test set of 500 observations each. The data consists of a single continuous response variable, y and 10 covariates $x_i, i = 1, \ldots, 10$.

4.4.1 Local Model

Here we fit a locally linear model to the same data for comparison using RF proximities as weights. The MSE for this model is 180.2. Since we have, in the process, estimated a unique set of coefficients for each test point, this means we end up with a matrix of coefficient values, allowing us to plot how each coefficient changes across the feature space. This allows a flexible collection of models to be explored.

Specifically, we are allowing the coefficients in our linear model to be *functions* of the data:

$$y = \beta_0 + \beta_1(\mathbf{x})x_1 + \cdots + \beta_{10}(\mathbf{x})x_{10} + \epsilon.$$

So if we plot β_1 versus x_1 we are visualizing how the coefficient of x_1 changes as a function of x_1. Of course, with ten variables there are many such two dimensional plots we could examine. Here we will present only a few, chosen for relevance. First consider Figure 4.5. The blue lines are *loess* smoothers applied to the scatterplots as visual aids. Notice that in (a) β_1 grows roughly linearly with x_1. This might suggest that $\beta_1 \propto x_1$, which would imply that x_1

is in fact influencing the response, y, in a quadratic fashion.

Continuing, we note that in (b) β_2 is roughly constant with respect to x_2, remaining near the value -2. This suggests that $\beta_2 \propto -2$. Similarly, we note that in (c) $\beta_3 \propto x_3^2$, suggesting that x_3 influences y in a cubic fashion. Finally, in (d) we note that $\beta_4 \propto -x_5$, which suggests a negative interaction between the variables x_4 and x_5 (a plot of β_5 vs. x_4 would show the same pattern).

In Figure 4.6 we show four more plots where each range of coefficients appears to be roughly constant and near zero, suggesting that these variables (and interactions) are less important.

The actual model used to generate the data was as follows:

$$\begin{gathered} y = 5x_1^2 - 2x_2 + 3x_3^3 - 4x_4x_5 + \epsilon \\ x_i \sim U(-3,3), i = 1, \ldots, 10 \\ \epsilon \sim N(0,2) \end{gathered}$$

Note that the variables $x_6, \ldots, x_{10}$ were irrelevant to the response y. Our local linear model correctly implied that x_1, x_2 and x_3 should be quadratic, linear and cubic terms, respectively, and that there should be a negative interaction between variables x_4 and x_5.

Next we demonstrate this visualization method on a real data set, the Boston Housing data. This is a regression problem where the response variable is the median value of owner-occupied homes (in thousands) together with 13 predictor variables. We can gain some insight into the data by examining Figure 4.7.

The first three variables, *crim* (per capita crime rate by town), *zn* (proportion of residential land zoned for lots over 25,000sq ft) and *indus* (proportion of non-retail business acres per town) each seem to have only a very mild impact on home values. The binary variable

Figure 4.5: Plots of coefficients from local linear models versus variable values.

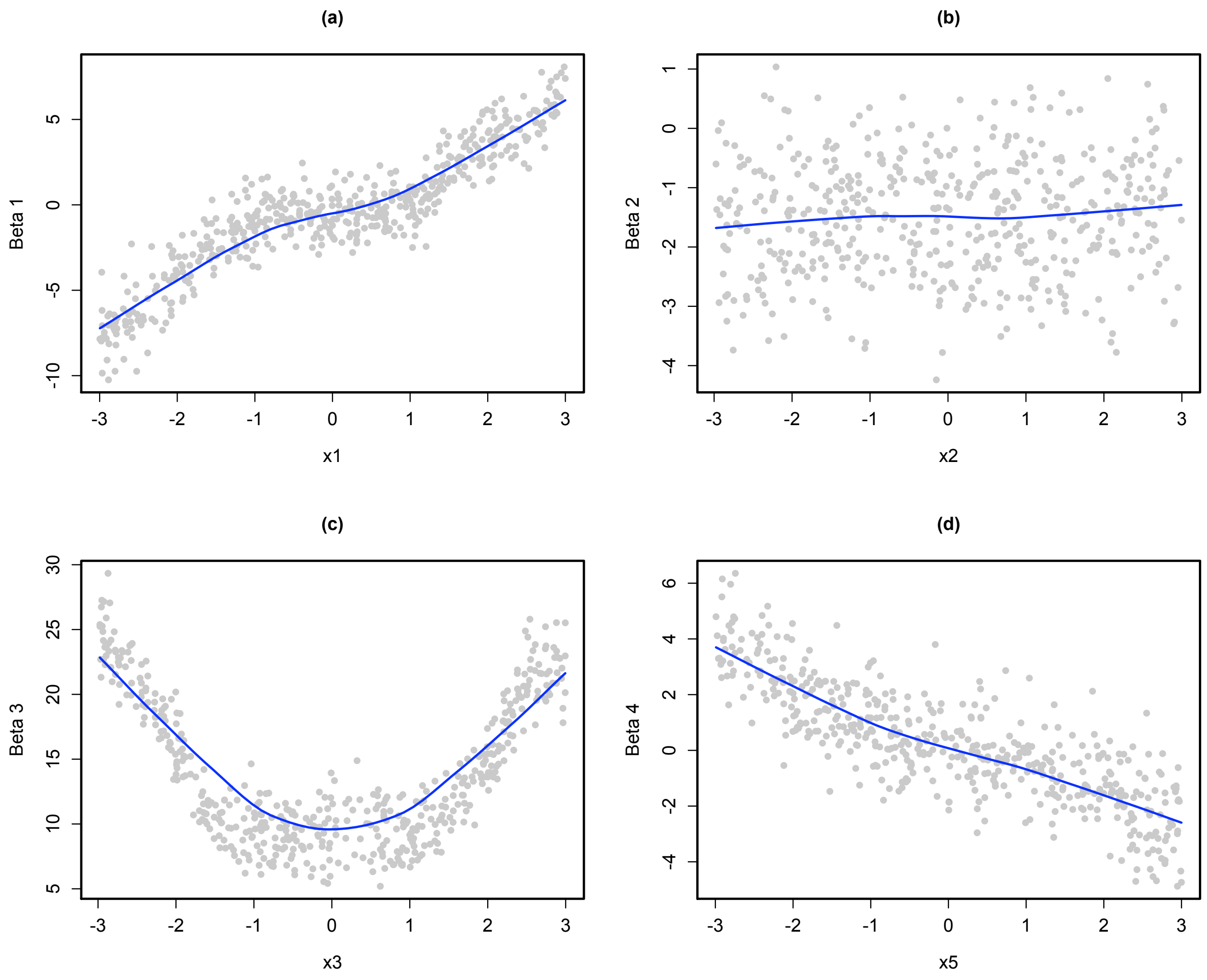
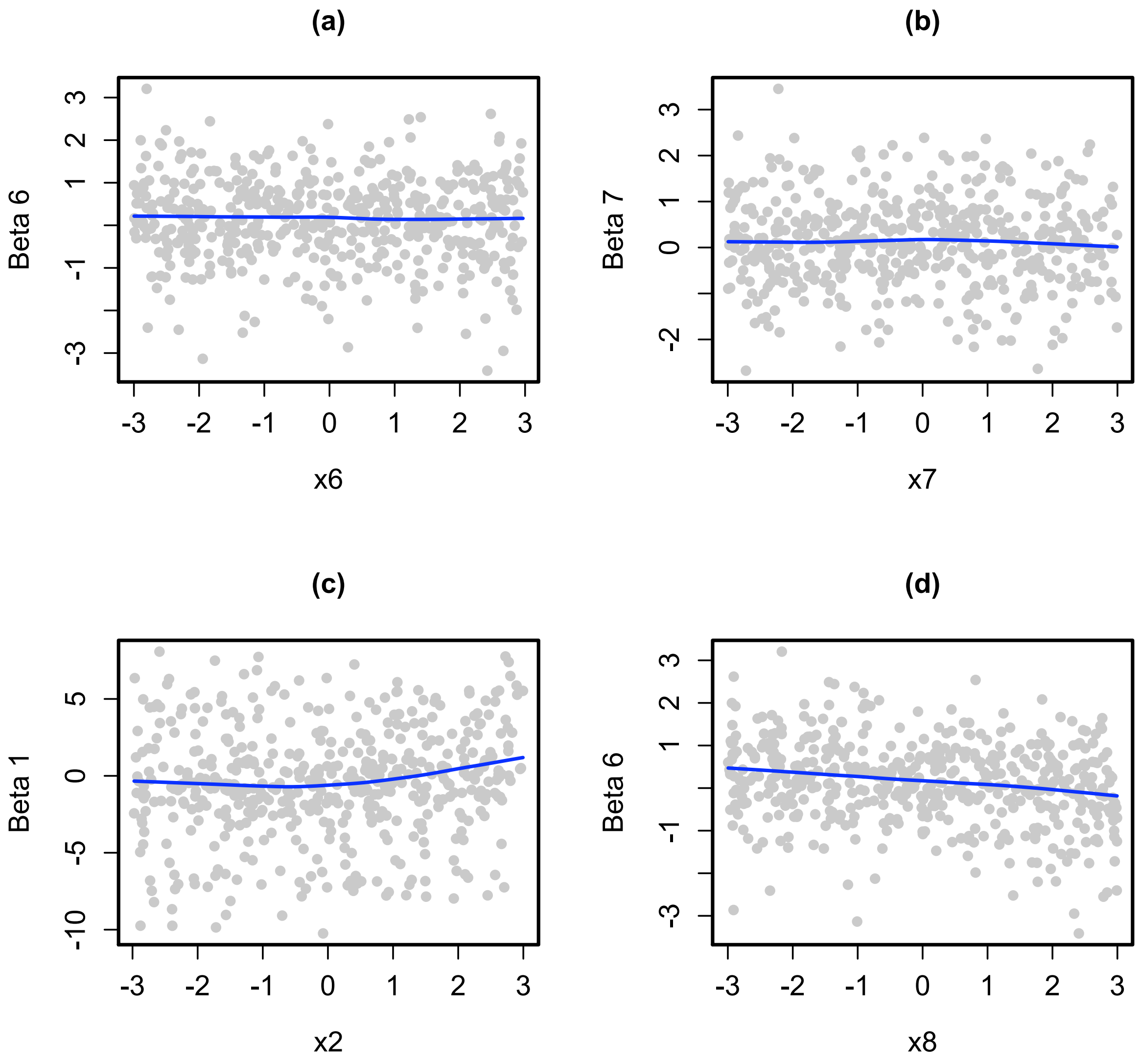


Figure 4.6: Plot of coefficients from local linear models versus variable values.



chas (= 1 if tract bounds Charles River, = 0 otherwise) is noteworthy in that it indicates (unsurprisingly) that lying on the shore of the Charles River is unlikely to *reduce* ones home value, relative to other locations. The behavior of *rm* (average number of rooms/dwelling) is also interesting. This probably acts as a decent proxy for average home size. Interestingly, average home size seems to have little impact on home values until the average number of rooms rises above about 6. Finally, we note the impact of *age* (proportion of owner-occupied units built prior to 1940). This appears to have a very weak negative effect on home values, until we reach the oldest neighborhoods (≥ 80) when it switches to having a small positive effect on home values.

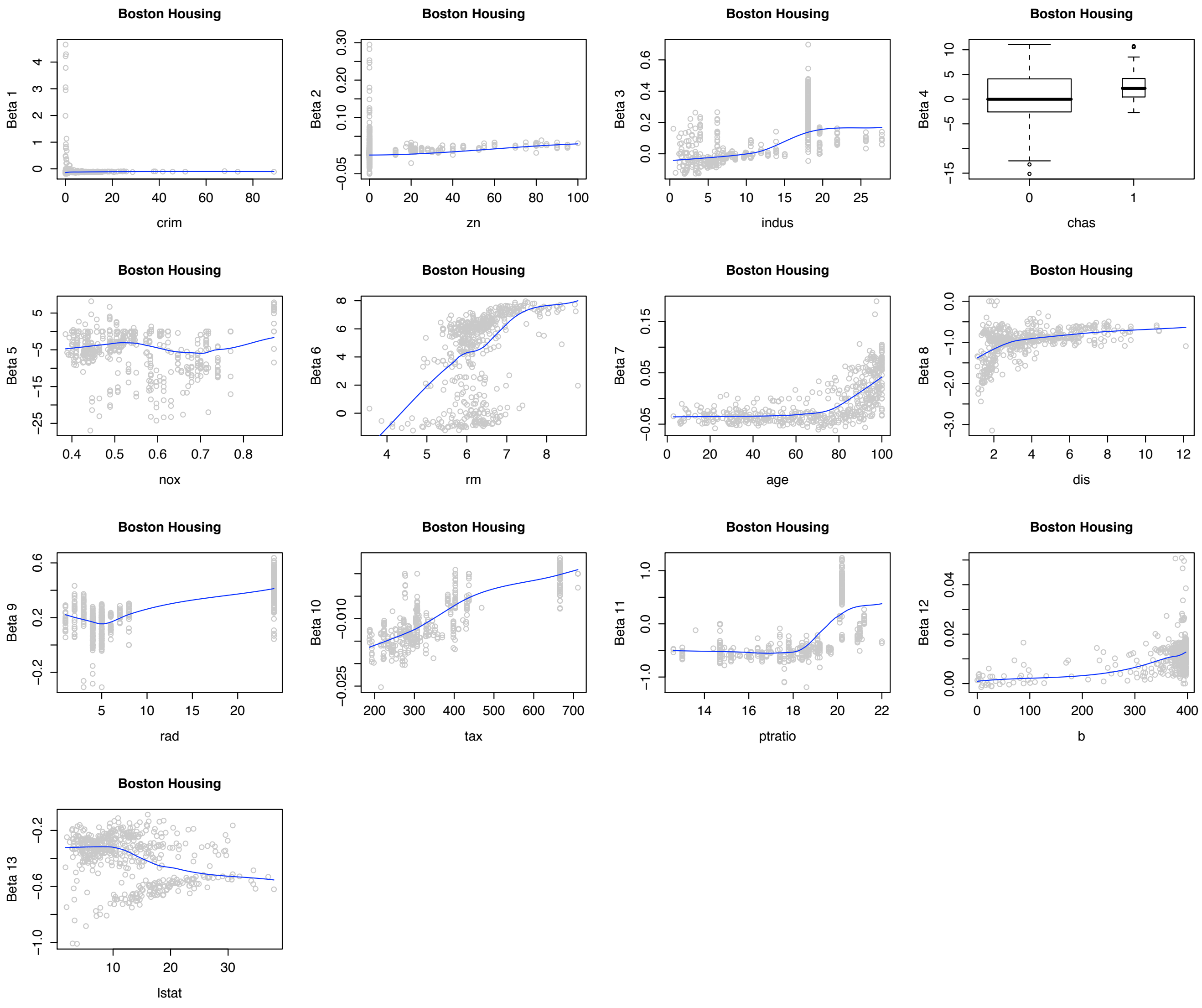


Figure 4.7: Scatterplots (with loess smooths in blue) of each coefficient with respect to their corresponding variable.

Chapter 5

Case Study: Prediction of Malaria Presence from Environmental and Climatic Data

5.1 Introduction

In this chapter we demonstrate how tree ensembles may be useful in the analysis of certain types of disease data. Specifically, we consider the case of *niche modeling* for malaria vectors in Africa. The modeling goal is to accurately predict the presence or absence of a certain species using only environmental or climatic data. These data first appeared in [25] and were kindly supplied by the authors.

5.2 Data

Malaria is a vector borne infectious disease occurring in primarily tropical regions. Human infection is caused by protozoan parasites that are spread by mosquitos (vectors). Transmission typically occurs when a female mosquito takes a blood meal from an infected human, ingesting the parasites. The mosquito then passes the parasites on while taking a blood meal from an uninfected human. These data concern malaria vectors, rather than infected humans.

The data consist of 977 locations for malaria vectors, along with values for 21 environmental variables at each location. (These are summarized in Table 5.1) This constitutes our *presence* data. Thirty-seven of these presences were subsequently omitted due to suspected geo-referencing errors causing them to appear on large bodies of water, leaving us with 940 presence observations. [1]

Additionally, we randomly selected 1553[2] locations on the African continent to serve as *absence* data. Clearly, we have no reason to be sure that malaria vectors are truly not present at these locations. This is a common, and oft discussed ([27],[21],[30],[32]), difficulty in niche modeling; namely that it is often impossible (or even nonsensical) to directly observe a species' absence. For this reason, our randomly selected locations are often referred to as *pseudo-absences*. We will not enter into the extensive debate about the use of pseudo-absence data here, except to say that this type of modeling is common in the niche modeling literature and that the difficulties it raises are beyond the scope of this work.

Along with these 1553 pseudo-absence locations, we collected the analogous environmental data for each location on the 21 variables mentioned above. Hence our complete data set can be organized into a matrix consisting of $940 + 1553 = 2493$ rows and 22 columns: one column

[1]For a more complete discussion of how these data were compiled using ArcMap GIS see [25].

[2]We desired a "large" number of background points, but there is no particular significance to this number; extracting a large random set of points from ArcGIS is an involved process that is not compatible with specifying the exact number of points ahead of time.

Table 5.1: Environmental parameters used in niche modeling.

Parameter
Annual Mean Temperature
Mean Diurnal Range
Isothermality
Temperature Seasonality
Maximum Temperature of Warmest Month
Minimum Temperature of Warmest Month
Temperature Annual Range
Mean Temperature of Wettest Quarter
Mean Temperature of Driest Quarter
Mean Temperature of Warmest Quarter
Mean Temperature of Coldest Quarter
Annual Precipitation
Precipitation of Wettest Month
Precipitation of Driest Month
Precipitation Seasonality
Precipitation of Wettest Quarter
Precipitation of Driest Quarter
Precipitation of Warmest Quarter
Precipitation of Coldest Quarter
Altitude
Land Cover

for the binary response variable of 1's (presences) and 0's (pseudo-absences) and 21 columns for each of the environmental variables. The goal is to accurately predict the response variable using the 21 environmental variables. Additionally, we would like to infer some information on the relative importance of the various environmental variables on the presence of malaria vectors.

5.3 Random Forests

Random Forests (RFs) were briefly outlined in Chapter 1. They are a particular form of tree ensemble with the following characteristics: B bootstrapped copies of the data are generated via sampling with replacement. A single decision tree is constructed on each of the B replicated data sets. At each node of each tree, M of the variables are randomly selected (without replacement) and these M variables are scanned for the best binary split at that node. Splitting continues until a node contains a single observation or until the node is "pure", meaning that there are only observations from one class in that node.

The bootstrap resampling allows for a convenient method for obtaining accurate assessments of prediction accuracy. Specifically, each observation will only be used in the construction of approximately 2/3 of the trees in the ensemble. We drop each observation down only those trees for which it was not included in that bootstrap sample to obtain a prediction; these predictions are aggregated to obtain a final prediction. The resulting error rate is called the *out-of-bag error*.

The resulting tree ensemble provides two ways to graphically investigate the importance of each variable on the response. First, we can compare each variable to each other to get a ranking of the most influential variables. Second, we can plot the partial dependence of the response on each variable. The variable importance measure is obtained using a permutation test. The values of each variable are permuted (while holding all other variables fixed). The degree to which this permutation increases the out-of-bag error rate is our measure of variable importance (the larger the increase, to more important the variable). The partial dependence plots are obtained by plotting the following function:

$$\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} f(x, x_{iC})$$

where n is the number of observations in the data, x is the variable for which the partial dependence is sought, x_{iC} is the covariate vector for the ith observation omitting the variable x and $f(x, x_{iC})$ is the log odds of belonging to class 1 (presence) for the (artificial) data vector (x, x_{iC}).

5.4 Analysis

The RF algorithm was run on our malaria data; we constructed an ensemble of 2000 trees, selecting 4 variables at each node. The resulting out-of-bag error estimate and confusion matrix is given in Table 5.2.

Table 5.2: Confusion matrix for malaria random forest niche model. The overall out-of-bag error rate was 0.039. (Rows are true values, columns are prediction by the RF model.)

	Pseudo-absence	Presence	Error Rate
Pseudo-absence	1510	43	0.028
Presence	54	886	0.057

As we can see, the RF model did an admirable job of identifying malaria vector presence observations. Figure 5.1 shows the environmental variables ranked in order of importance, as measure by the mean decease in out-of-bag accuracy upon permuting that variable. Altitude is clearly the most important factor, followed by precipitation of the wettest month and quarter, landcover and the minimum temperature of the coldest month. Landcover is a categorical (unordered) variable with 14 levels that is difficult to interpret as we were not provided a key for the specific types of landcover classes it indicates. The remaining variables are of lesser importance, by this measure.

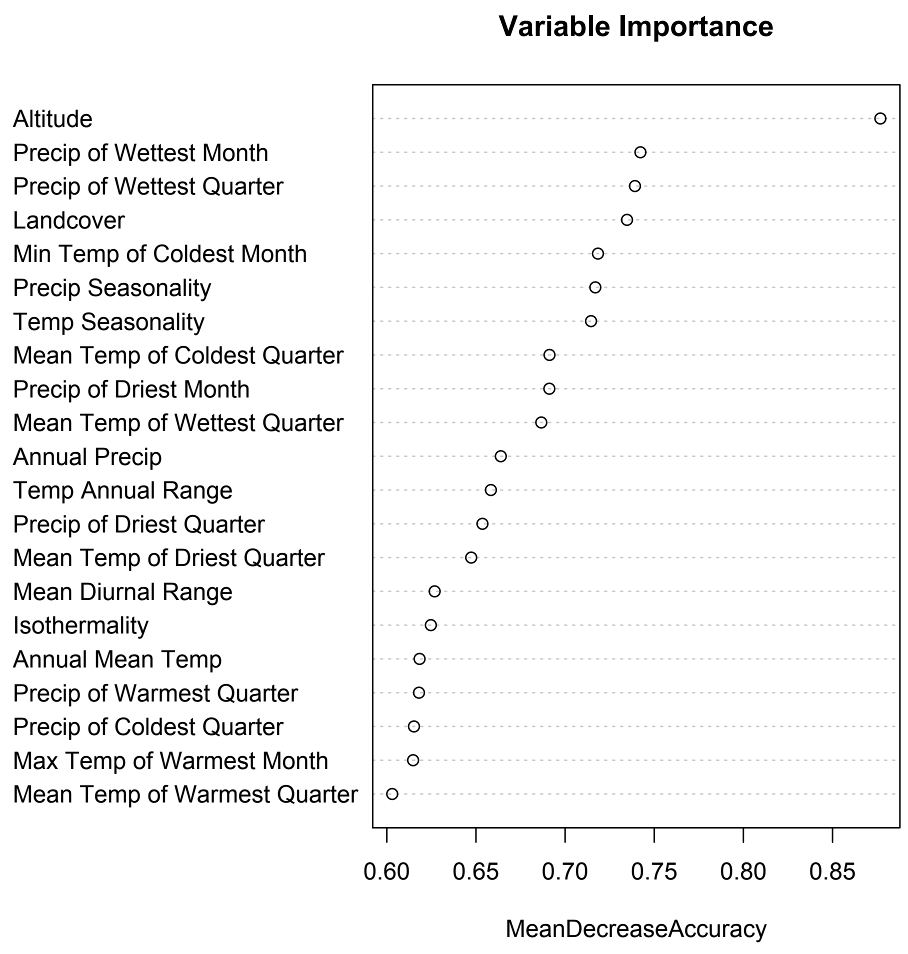


Figure 5.1: Variable importance plot for malaria vector niche model.

Figures 5.2, 5.3, 5.4, 5.5, and 5.6 show the partial dependence plots for the five most important variables on the response. We can interpret the vertical axis of these plots as the approximate log odds of class 1 (presence). Hence positive values indicate an increased likelihood of malaria vector presence and negative values indicate a decreased likelihood. The horizontal axis represents the range of values for the indicated variable that occur in the data set; the rug plot indicates the deciles of that variable.

We can conclude from these graphs that malaria vectors are most likely to be present at relatively low elevations (< 500 meters), in climates that are relatively wet, and that are relatively warm all year (i.e. mild winters).

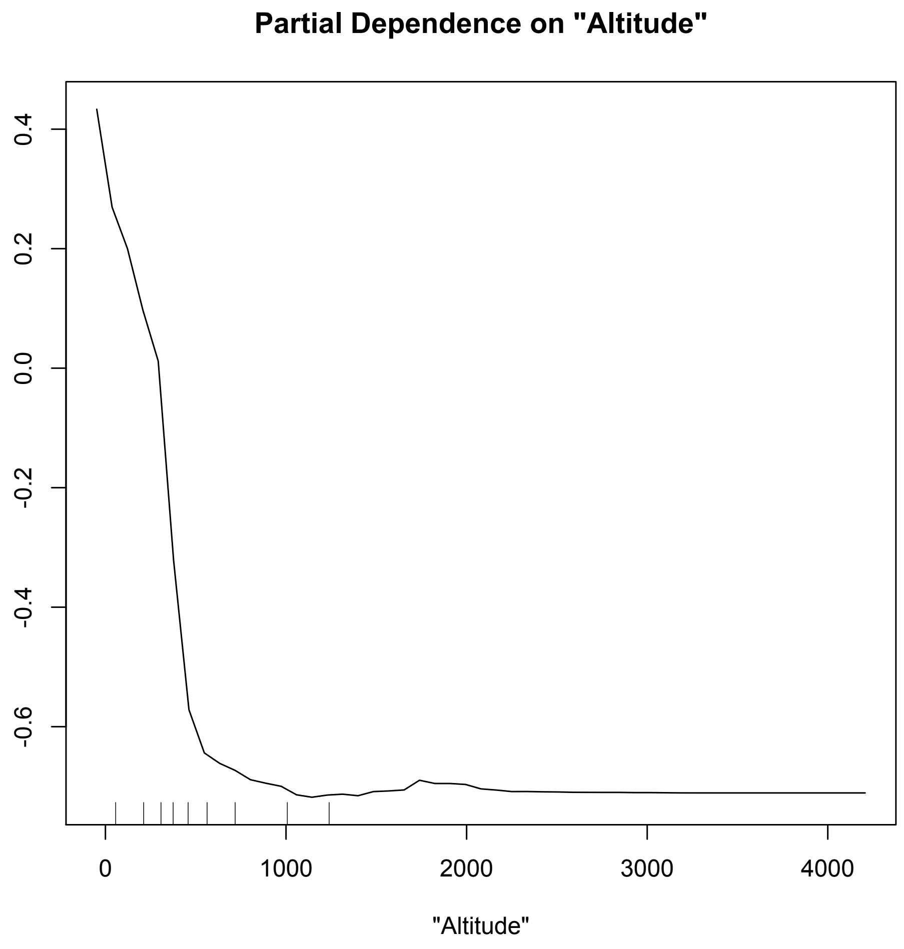


Figure 5.2: Partial dependence plot of altitude (in meters) on the log odds of malaria vector presence.

We can also apply the methods from Chapter 4 to use the proximities from the RF model to fit a locally linear model. We found that centering and scaling the covariate values helped stabilize the linear models, so the values here will not directly correspond to the partial

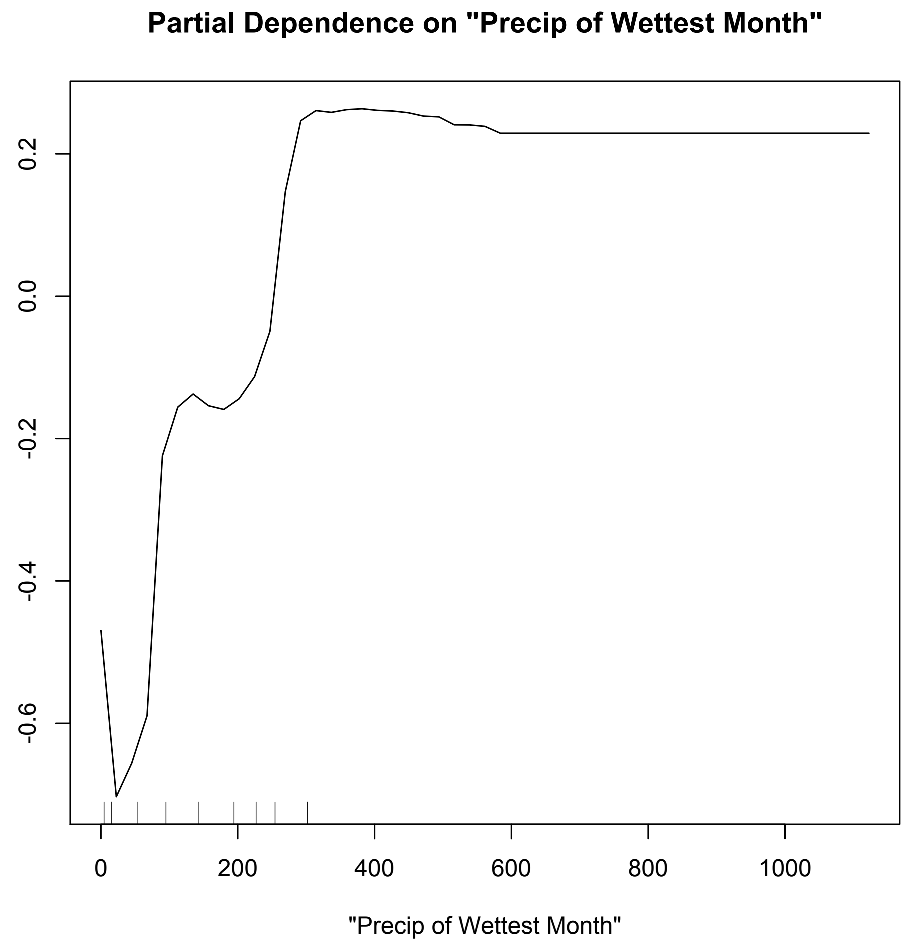


Figure 5.3: Partial dependence plot of precipitation of the wettest month (units unknown) on the log odds of malaria vector presence.

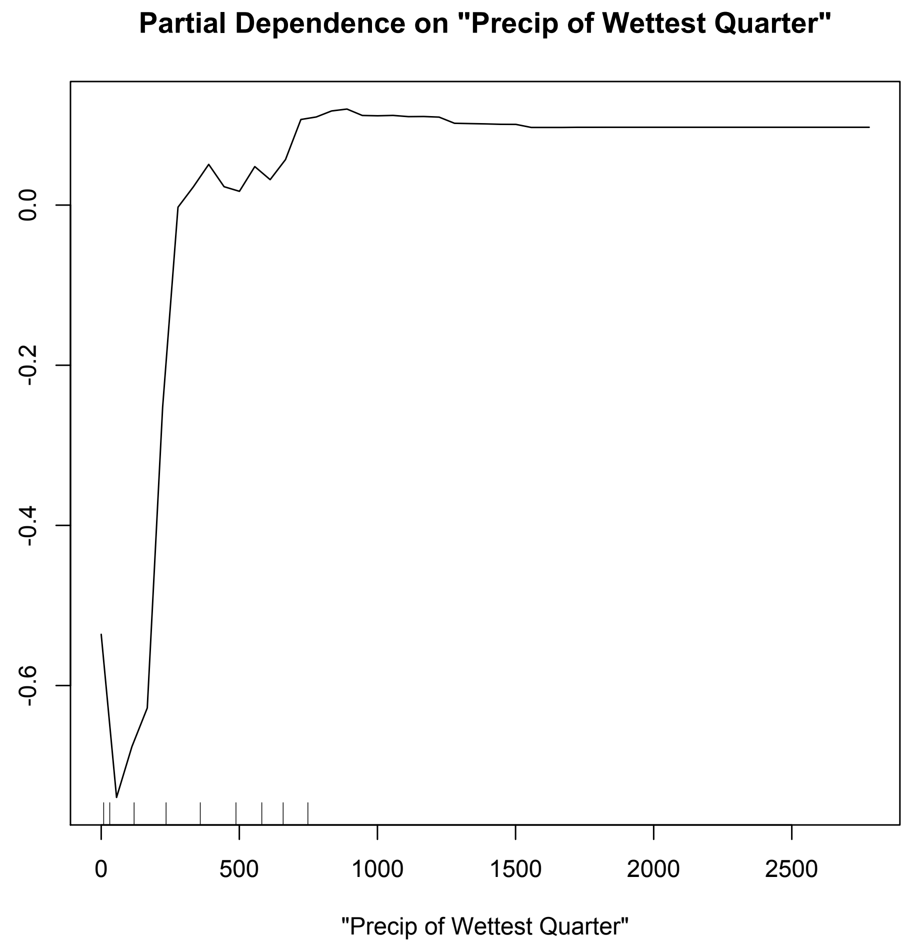


Figure 5.4: Partial dependence plot of precipitation of the wettest quarter (units unknown) on the log odds of malaria vector presence.

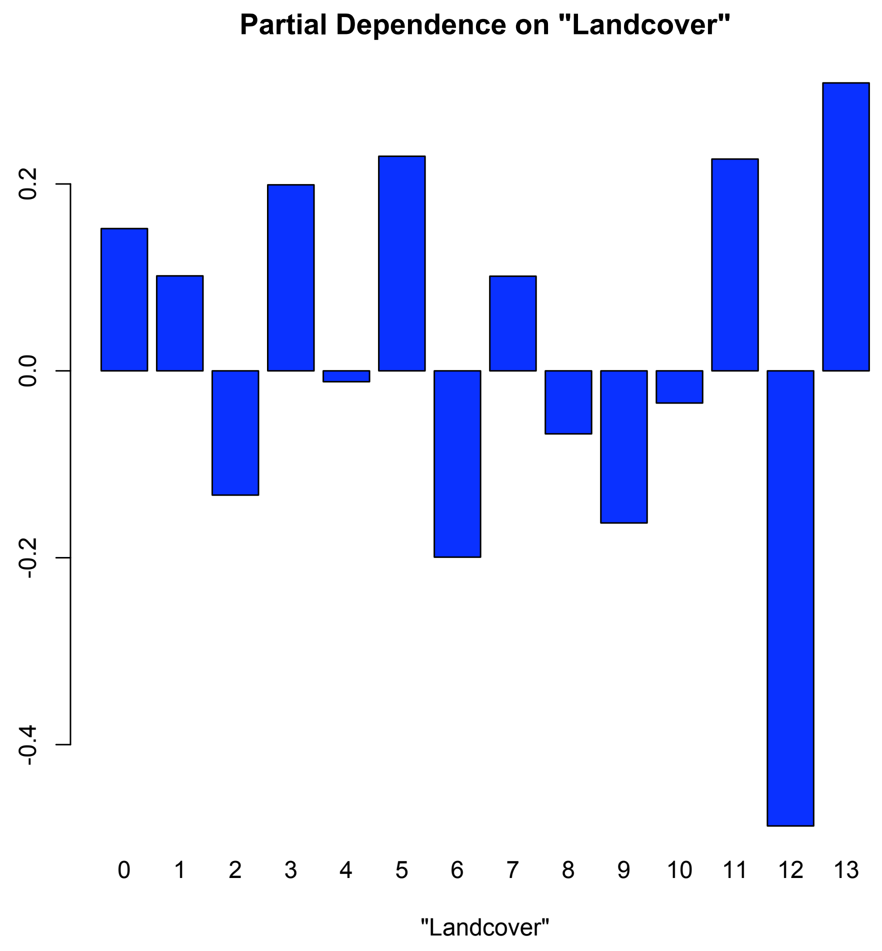


Figure 5.5: Partial dependence plot of landcover type (categories unknown) on the log odds of malaria vector presence. While the landcover classes associated with these categories is unknown, one might surmise that landcover 12 corresponds to something analogous to “desert”.

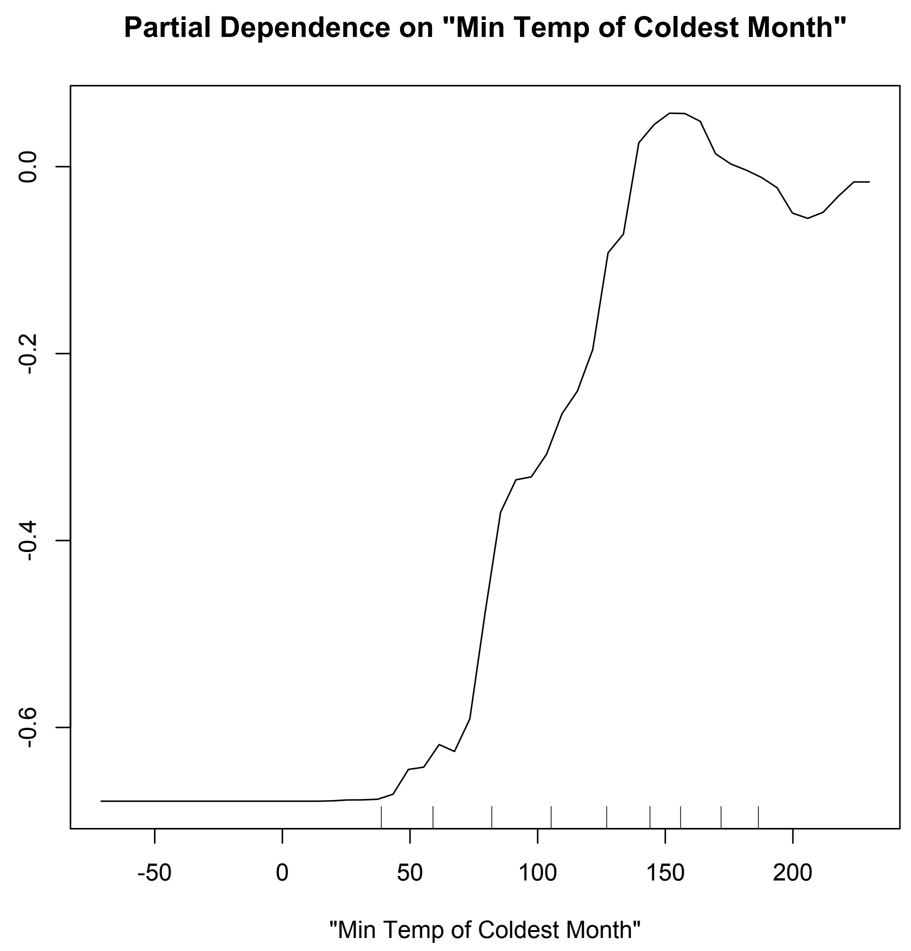


Figure 5.6: Partial dependence plot of the minimum temperature of the coldest month (units unknown) on the log odds of malaria vector presence.

dependance plots shown before. The locally linear model yields a slightly lower error rate of 0.026, although we should be careful making comparisons as these error rates were estimated differently (out-of-bag vs. leave one out cross validation). Additionally, visualizations of the estimated coefficients yields similar conclusions about the relative importance of the variables. For example, Figures 5.7 and 5.8 show the estimated coefficients for Altitude and Precipitation of the Wettest Month.

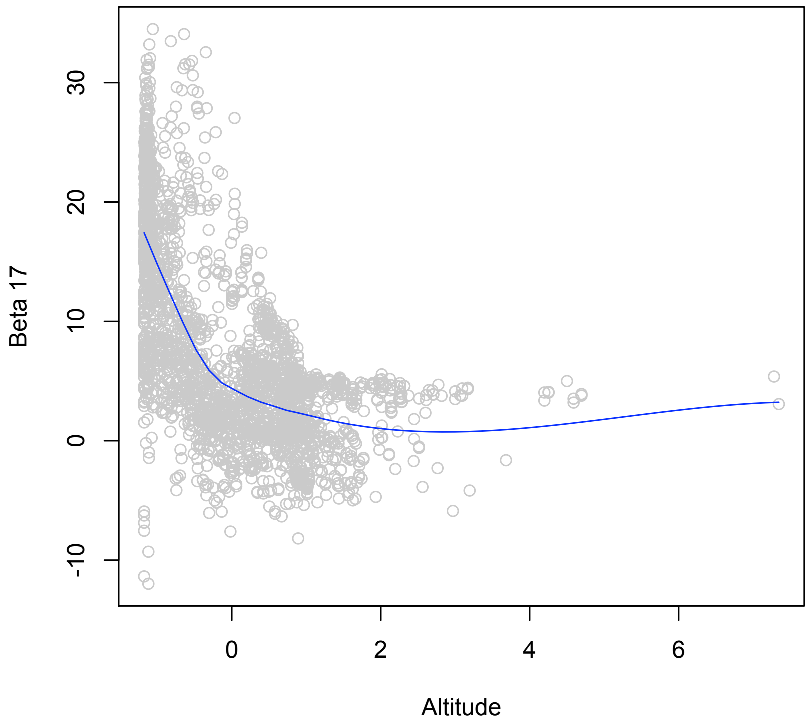


Figure 5.7: Estimated coefficient versus Altitude for the locally linear model.

For comparison, we analyze these malaria data using logistic regression. First, we fit the entire data set using logistic regression and chose the best model using backwards stepwise selection using AIC as the selection criteria. This procedure removed only three variables: temperature annual range, mean temperature of driest quarter and precipitation of the wettest quarter. The remaining variables have all been retained in the model. The resulting coefficient

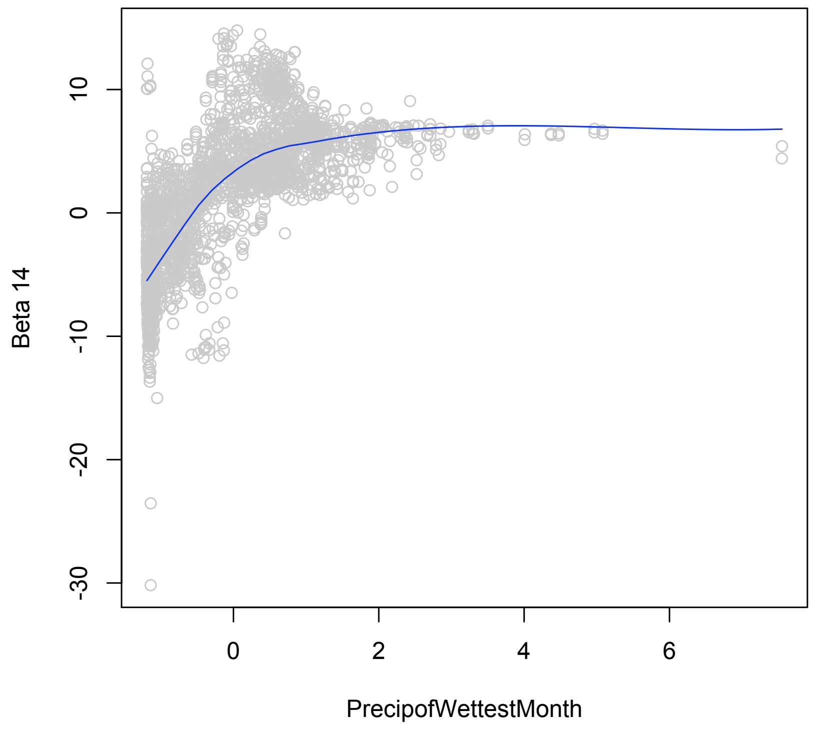


Figure 5.8: Estimated coefficient versus Precipitation of the Wettest Month for the locally linear model.

estimates are shown in Table 5.3. The 10-fold cross validation error estimate for this model is 0.117. Backward stepwise selection via AIC is a fairly liberal procedure, meaning that it will tend to allow the inclusion of many variables.

Next we attempted to find a reduced model by hand. We began by scaling the (continuous) covariates and checking them for collinearity. Many of these 22 variables are very highly correlated (Pearson correlation coefficient > 0.9). In order to maintain a useful comparison with the RF models, we elected to retain variables that the RF model identified as important, whenever possible. Five variables (Temperature Annual Range, Mean Temperature of the Warmest Quarter, Mean Temperature of the Coldest Quarter, Precipitation of the Wettest Quarter and Precipitation of the Driest Quarter) were omitted for the remainder of this analysis.

We fit a linear logistic model using all of the remaining variables. Next we performed a backwards stepwise elimination procedure (by hand) using χ^2 tests. This process removed another five variables. Once we'd obtained a reduced model using only linear terms, we examined the addition of quadratic terms. Ultimately, three were deemed significant and added to the model. A summary of the resulting final model is shown in Table 5.4 (note that we have scaled the covariates, so the coefficient estimates differ in scale from those in Table 5.3). The 10-fold cross validated error estimate of this model was 0.101. There are too many interaction terms to check by hand and have too little intuition about the physical system to propose likely variable interactions that may exist, so we opted not to pursue adding interaction terms.

Selecting a reduced model by hand did improve the error rate of our model slightly. It is possible that further improvements in accuracy are possible by refining this logistic model, although the time and complexity required may become prohibitive. However, none of the logistic models came close to matching the tree ensembles in terms of pure accuracy. That said, both approaches generally identified altitude and precipitation of the wettest month as being particularly important, so both methods are in a crude sense both identifying altitude and precipitation as the most important variables.

5.5 Conclusion

Tree ensembles, and RFs in particular, can be a powerful technique for niche modeling. They are a fast, easy to use and extremely accurate modeling technique that easily accommodates large numbers of covariates. Additionally, RFs provide convenient tools to graphically investigate the importance and influence that each covariate has on the response. These features are particular useful in cases where such visualizations using standard methods are more challenging (high dimensionality, classification tasks). For these reasons, tree ensembles should be considered a strong candidate for modeling tasks like species niche modeling.

Table 5.3: Logistic regression coefficients of best model resulting from backwards stepwise selection using AIC.

	Estimate	Std. Error	z value	Pr(>|z|)
Intercept	2.15	1.26	1.71	0.09
Altitude	-0.002	0.00	-8.27	0.00
Landcover1	3.08	1.16	2.65	0.01
Landcover2	-1.97	0.59	-3.33	0.00
Landcover3	-0.01	0.82	-0.02	0.99
Landcover4	-0.50	0.82	-0.61	0.54
Landcover5	2.54	0.94	2.70	0.01
Landcover6	0.32	0.55	0.59	0.55
Landcover7	1.19	0.53	2.25	0.02
Landcover8	1.03	0.56	1.86	0.06
Landcover9	0.65	0.55	1.17	0.24
Landcover10	0.89	0.57	1.56	0.12
Landcover11	2.63	0.61	4.30	0.00
Landcover12	0.09	0.59	0.16	0.88
Landcover13	17.86	381.88	0.05	0.96
AnnualMeanTemp	0.09	0.03	2.73	0.01
MeanDiurnalRange	1.82	1.29	1.41	0.16
Isothermality	-1.76	1.29	-1.36	0.17
TempSeasonality	-0.01	0.00	-4.63	0.00
MaxTempofWarmestMonth	-0.12	0.02	-7.31	0.00
MinTempofColdestMonth	0.04	0.01	3.27	0.00
MeanTempofWettestQuarter	0.01	0.00	3.58	0.00
MeanTempofWarmestQuarter	0.26	0.05	5.42	0.00
MeanTempofColdestQuarter	-0.27	0.04	-5.92	0.00
AnnualPrecip	-0.01	0.00	-9.73	0.00
PrecipofWettestMonth	0.05	0.00	15.84	0.00
PrecipofDriestMonth	-0.04	0.03	-1.43	0.15
PrecipSeasonality	-0.03	0.00	-8.22	0.00
PrecipofDriestQuarter	0.02	0.01	2.40	0.02
PrecipofWarmestQuarter	-0.00	0.00	-1.67	0.10
PrecipofColdestQuarter	-0.00	0.00	-3.11	0.00

Table 5.4: Logistic regression coefficients of best model resulting from manual backwards stepwise selection followed by forward addition of quadratic terms.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.00	0.60	0.01	1.00
Altitude	-1.58	0.16	-10.12	0.00
Landcover1	3.69	0.98	3.78	0.00
Landcover2	-2.34	0.67	-3.51	0.00
Landcover3	-0.17	0.88	-0.19	0.85
Landcover4	-0.02	0.84	-0.02	0.98
Landcover5	2.37	0.97	2.44	0.01
Landcover6	-0.04	0.62	-0.07	0.94
Landcover7	1.02	0.60	1.70	0.09
Landcover8	1.22	0.63	1.93	0.05
Landcover9	0.99	0.62	1.58	0.11
Landcover10	1.27	0.65	1.96	0.05
Landcover11	2.17	0.67	3.22	0.00
Landcover12	-0.59	0.68	-0.88	0.38
Landcover13	18.38	389.08	0.05	0.96
AnnualMeanTemp	2.78	0.59	4.74	0.00
MeanDiurnalRange	1.50	0.35	4.33	0.00
TempSeasonality	4.22	0.62	6.78	0.00
MaxTempofWarmestMonth	-5.10	0.68	-7.54	0.00
MinTempofColdestMonth	2.98	0.69	4.30	0.00
AnnualPrecip	-3.75	0.38	-9.79	0.00
PrecipofWettestMonth	5.59	0.39	14.26	0.00
PrecipofDriestMonth	2.45	0.29	8.55	0.00
PrecipSeasonality	-1.11	0.15	-7.25	0.00
PrecipofColdestQuarter	-0.34	0.10	-3.43	0.00
I(Altitude^2)	0.37	0.04	9.43	0.00
I(MinTempofColdestMonth^2)	-1.46	0.16	-8.95	0.00
I(PrecipofDriestMonth^2)	-0.60	0.07	-8.84	0.00

Bibliography

[1] Y. Amit, D. Geman, and K. Wilder, *Joint induction of shape features and tree classifiers*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997), no. 11, 1300–1305.

[2] L. Breiman, *Bagging predictors*, Machine Learning **24** (1996), no. 2, 123–140.

[3] ______, *Bias, variance, and arcing classifiers*, Statistics (1996).

[4] ______, *Classification and regerssion trees*, Chapman & Hall/CRC, 1998.

[5] ______, *Random forests*, Machine Learning **45** (2001), no. 1, 5–32.

[6] P. Bühlmann and B. Yu, *Analyzing bagging*, The Annals of Statistics **30** (2002), no. 4, 927–961.

[7] A. Buja and W. Stuetzle, *Bagging does not always decrease mean square error*, Preprint (2000).

[8] ______, *Smoothing effects of bagging* (2000).

[9] T. Cover and P. Hart, *Nearest neighbor pattern classification*, Information Theory, IEEE Transactions on **13** (1967), no. 1, 21–27.

[10] A. Cutler and Z. Guohua, *Pert - perfect random tree ensembles*, Computing Science and Statistics **33** (2001).

[11] C. Domeniconi, J. Peng, and D. Gunopulos, *Locally adaptive metric nearest-neighbor classification*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002), no. 9, 1281–1285.

[12] P. Domingos, *Why does bagging work? a bayesian account and its implications*, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (1997), 155–158.

[13] W. Fan, H. Wang, PS Yu, and S. Ma, *Is random model better? on its accuracy and efficiency*, Data Mining, 2003. ICDM 2003. Third IEEE International Conference on Data Mining (2003), 51–58.

[14] Wei Fan, Joe McCloskey, and Philip S. Yu, *A general framework for accurate and fast regression by data summarization in random decision trees*, Kdd '06: Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining, 2006, pp. 136–146.

[15] Y. Freund and R.E. Schapire, *Experiments with a new boosting algorithm*, Machine Learning: Proceedings of the Thirteenth International Conference **148** (1996), 156.

[16] J.H. Friedman and P. Hall, *On bagging and nonlinear estimation*, Journal of statistical planning and inference **137** (2007), no. 3, 669–683.

[17] P. Geurts, D. Ernst, and L. Wehenkel, *Extremely randomized trees*, Machine Learning **63** (2006), no. 1, 3–42.

[18] Y. Grandvalet, *Bagging equalizes influence*, Machine Learning **55** (2004), no. 3, 251–270.

[19] T. Hastie and R. Tibshirani, *Discriminant adaptive nearest neighbor classification*, IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996), no. 6, 607–616.

[20] T.K. Ho, *The random subspace method for constructing decision forests*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 8, 832–844.

[21] K.I.M.A. Keating and S. Cherry, *Use and Interpretation of Logistic Regression in Habitat-Selection Studies*, Journal of Wildlife Management **68** (2004), no. 4, 774–789.

[22] Y. Lin and Y. Jeon, *Random forests and adaptive nearest neighbors*, Journal of the American Statistical Association **101** (2006), no. 474, 578–590.

[23] F.T. Liu, K.M. Ting, and W. Fan, *Maximizing tree diversity by building complete-random decision trees*, Proceedings of Nineth Pacific Asian Knowledge Discovery and Data Mining Conference (PAKDD05) (2005).

[24] F.T. Liu and K.M. Ting, *Variable randomness in decision tree ensembles* (2006).

[25] A. Moffett, N. Shackelford, and S. Sarkar, *Malaria in africa: Vector species' niche models and relative risk maps*, PLoS ONE **2** (2007), no. 9.

[26] JP Myles and DJ Hand, *The multi-class metric problem in nearest neighbour discrimination rules*, Pattern Recognition **23** (1990), no. 11, 1291–1297.

[27] J.L. PEARCE and M.S. BOYCE, *Modelling distribution and abundance with presence-only data*, Journal of Applied Ecology **43** (2006), no. 3, 405–412.

[28] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, *Boosting the margin: A new explanation for the effectiveness of voting methods*, The Annals of Statistics **26** (1998), no. 5, 1651–1686.

[29] RD Short and K. Fukunaga, *Optimal distance measure for nearest neighbor classification*, IEEE TRANS. INFO. THEORY **27** (1981), no. 5, 622–627.

[30] G. Ward, T. Hastie, S. Barry, J. Elith, and J.R. Leathwick, *Presence-only data and the em algorithm* (2007).

[31] D. Wolpert and W. Macready, *Combining stacking with bagging to improve a learning algorithm*, Machine Learning (1998).

[32] A.E. Zaniewski, A. Lehmann, and J.M.C. Overton, *Predicting species spatial distributions using presence-only data: a case study of native new zealand ferns*, Ecological Modelling **157** (2002), no. 2-3, 261–280.