

Flexible Metric Nearest Neighbor Classification

Jerome H. Friedman*

November 11, 1994

Abstract

The K -nearest-neighbor decision rule assigns an object of unknown class to the plurality class among the K labeled “training” objects that are closest to it. Closeness is usually defined in terms of a metric distance on the Euclidean space with the input measurement variables as axes. The metric chosen to define this distance can strongly effect performance. An optimal choice depends on the problem at hand as characterized by the respective class distributions on the input measurement space, and within a given problem, on the location of the unknown object in that space. In this paper new types of K -nearest-neighbor procedures are described that estimate the local relevance of each input variable, or their linear combinations, for each individual point to be classified. This information is then used to separately customize the metric used to define distance from that object in finding its nearest neighbors. These procedures are a hybrid between regular K -nearest-neighbor methods and tree-structured recursive partitioning techniques popular in statistics and machine learning.

1 Introduction

Nearest-neighbor methods are among the most popular for classification. They represent the earliest general (nonparametric) methods proposed for this problem and were heavily investigated in the fields of statistics and (especially) pattern recognition. Recently renewed interest in them has emerged in the connectionist literature (“memory” methods) and also in machine learning (“instance-based” methods). Despite their basic simplicity and the fact that many more sophisticated alternative techniques have been developed since their introduction, nearest-neighbor methods still remain among the most successful for many classification problems.

This paper presents extensions to the basic nearest-neighbor approach with the goal of enhancing its performance in certain situations. These are characterized by the fact that the measured variables input into the classification procedure may not all be equally relevant for classifying a new object. Moreover, this differential relevance may depend on the location of the object in the input measurement space. The relevance of a particular input variable for classifying the object may depend on the particular object being classified within the same classification problem. It is well known that input variables of low relevance can degrade the performance of nearest-neighbor procedures if they are allowed to be equally influential with those of high relevance in defining the (near-neighbor) distance from the point to be classified.

*Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94305 (jhf@playfair.stanford.edu). Work supported in part by the Department of Energy under contract number DE-AC03-76SF00515, and by the National Science Foundation under grant number DMS-9403804.

If the relative (local) relevance of each input variable were known, this information could be used to advantage by constructing a metric to define nearest-neighbor distance that provides differential weighting for the inputs; variables of higher relevance receive more weight in defining the distance. Unfortunately, such information is seldom available in advance so that it is either ignored, or attempts must be made to estimate it from the training data at hand. In this paper new types of nearest-neighbor procedures are described that estimate the local relevance of each input variable for each individual object to be classified. This information is then used to separately customize the metric used to define the distance from that object in finding its nearest-neighbors.

The paper is organized as follows. The next section defines the (generic) classification problem establishing notation for what follows. Section 3 focuses on the K -nearest-neighbor approach characterizing its strengths and weaknesses. Two established methods for attempting to mitigate these weaknesses are then described: input variable subset selection, and the recursive partitioning approach basic to the tree-structured induction methods of machine learning. Possible limitations of these methods are described so as to motivate the new procedures described in Section 4. Sections 4.1 - 4.3 describe measures of local input variable relevance and how they can be estimated from the training data at hand. Section 4.4 describes one way to use these estimates (the “machete”) to form a local metric for classifying a new object. Section 4.5 discusses generalizations of the basic machete through including additional (“derived”) variables that are functions of the original inputs. These derived variables are individually customized to have high local relevance for the particular object being classified. Section 4.6 presents a further generalization (the “scythe”) which defines a whole class of classification procedures including the machete and ordinary K -nearest-neighbor methods as special cases. The relative performances of several versions of these procedures, as well as the ordinary K -nearest-neighbor method and tree-based recursive partitioning methods are examined in Section 5. Section 5.1 compares the methods through a set of artificially simulated examples while Section 5.2 uses real data examples. A concluding summary is presented in Section 6.

2 Classification

In the classification problem an object, characterized by a set of measurements $\mathbf{x} = (x_1, \dots, x_p) \in R^p$, is presumed to be a member of one of J groups (“classes”) $\mathbf{x} \in \{G_j\}_1^J$. The particular group is unknown, and the goal is to assign the object to the correct group using its measured values \mathbf{x} . More formally, let L_{jk} be a loss (cost) associated with assigning it to the j th group $\mathbf{x} \rightarrow G_j$, when it is actually a member of the k th group $\mathbf{x} \in G_k$. Then the expected loss (“misclassification risk”) is

$$R_j = \sum_{k=1}^J L_{jk} \Pr(k | \mathbf{x}) \quad (1)$$

where $\Pr(k | \mathbf{x})$ is the probability that \mathbf{x} is a member of the k th group given the particular set of measurement values \mathbf{x} . The risk (1) is minimized by the assignment $\mathbf{x} \rightarrow G_{j^*}$ with

$$j^* = \arg \min_{1 \leq j \leq J} R_j(\mathbf{x}) \quad (2)$$

which reduces to

$$j^* = \arg \max_{1 \leq j \leq J} \Pr(j | \mathbf{x}) \quad (3)$$

if all misclassifications are deemed equally costly

$$L_{jk} = 1 - \delta_{jk}, \quad \delta_{jk} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases} \quad (4)$$

so that the risk reduces to the misclassification (error) rate. These (2) (3) are known as the “Bayes” decision rules (for a given problem and loss matrix L_{jk}) and their associated misclassification risk represents the minimum achievable.

In order to apply the Bayes rule the (true) conditional probabilities $\{\Pr(j | \mathbf{x})\}_1^J$ must be known for every point $\mathbf{x} \in R^p$ at which assignment predictions are to be made. This is seldom possible. With the “supervised learning” paradigm one obtains a (“training”) sample of correctly labeled objects

$$\{\mathbf{x}_n, g_n\}_1^N, \quad g_n = j \Rightarrow \mathbf{x}_n \in G_j \quad (5)$$

presumed to be a random sample drawn with respective probabilities $\{\Pr(j | \mathbf{x}_n)\}_1^J$. The distribution of the training sample $\{\mathbf{x}_n\}_1^N$ over the measurement space is also presumed to be (at least somewhat) representative of the corresponding distribution of future “test” objects to be classified. The training data (5) is used to obtain estimates

$$\{\widehat{\Pr}(j | \mathbf{x})\}_1^J \quad (6)$$

which are then used in (1) (2)

$$\hat{j}(\mathbf{x}) = \arg \min_{1 \leq j \leq J} \sum_{k=1}^J L_{jk} \widehat{\Pr}(k | \mathbf{x}) \quad (7)$$

or the corresponding analog of (3) to estimate class assignment.

Strategies for supervised learning fall into two basic categories. One approach is based on density estimation via Bayes theorem

$$\Pr(j | \mathbf{x}) = \Pr(\mathbf{x} | j) \Pr(j) \bigg/ \sum_{k=1}^J \Pr(\mathbf{x} | k) \Pr(k). \quad (8)$$

Here $\Pr(\mathbf{x} | j)$ is the probability density distribution of the j th class over the measurement space and $\Pr(j)$ is the “prior” probability of observing a class j object in the absence of a set of measurement values \mathbf{x} . With this approach the training sample is partitioned into J groups according to class label, and the data in each group is used separately to estimate the respective class conditional probability densities $\Pr(\mathbf{x} | j)$ over the measurement space. The prior probabilities are either known in advance, or are estimated as the proportions of each class in the training sample. These estimates are then used to derive estimates of the location conditional probabilities (6) through (8). An example of this density estimation approach is “discriminant analysis” [see McLachlan (1992)] in which the class conditional densities are approximated by normal distributions and the training data for each class is used to estimate the respective parameters (mean vector and covariance matrix) for its class. Methods based on (normal) mixtures [Chow and Chen (1992)] and learning vector quantization techniques [Kohonen (1990)] are also examples of this approach.

The density estimation approach to classification estimates probabilities conditioned on the value of the class label. The second category of supervised learning techniques, based on regression, condition on the prediction point \mathbf{x} and directly estimate (6) the probabilities $\{\Pr(j | \mathbf{x})\}_1^J$. At a given prediction point \mathbf{x} , the class label g is assumed to be a random variable from a multinomial distribution with probabilities $\{\Pr(j | \mathbf{x})\}_1^J$. Each potential value for g (at \mathbf{x}) is characterized by a separate (“dummy”) output variable

$$y_j | \mathbf{x} = \begin{cases} 1 & g | \mathbf{x} = j \\ 0 & g | \mathbf{x} \neq j \end{cases}. \quad (9)$$

Clearly

$$f_j(\mathbf{x}) \doteq \Pr(j | \mathbf{x}) = \Pr(y_j = 1 | \mathbf{x}) = E(y_j | \mathbf{x}) \quad (10)$$

and

$$f_j(\mathbf{x}) = \arg \min_f E[(y_j - f)^2 | \mathbf{x}], \quad j = 1, J, \quad (11)$$

so that the location conditional probabilities $\{f_j(\mathbf{x}) = \Pr(j | \mathbf{x})\}_1^J$ are the solutions to a set of (conditional) least-squares problems. The regression approach assumes that the training data (5), and future data to be predicted, are random realizations from this process and applies standard regression methodology to estimate each of the separate “target” functions $f_j(\mathbf{x})$ from the corresponding training samples

$$\{\mathbf{x}_n, y_j\}_{n=1}^N, \quad j = 1, J. \quad (12)$$

Because they represent probabilities the target functions $\{f_j(\mathbf{x})\}_1^J$ satisfy the constraints

$$0 \leq f_j(\mathbf{x}) \leq 1, \text{ and } \sum_{j=1}^J f_j(\mathbf{x}) = 1 \quad (13)$$

for all values of \mathbf{x} . CART [Breiman, Friedman, Olshen and Stone (1984)], projection pursuit [Friedman (1985)], neural networks [Lippmann (1989)], nearest-neighbor kernel methods, and many of the techniques developed in machine learning and pattern recognition either directly or indirectly apply this regression paradigm to the classification problem.

3 Nearest-neighbor kernel methods

Kernel methods are among the earliest proposed methods for nonparametric regression [Parzen(1962)] Their basic (and only) assumption is smoothness of the target functions $\{f_j(\mathbf{x})\}_1^J$ (10) to be estimated. That is, $f_j(\mathbf{x} + \delta\mathbf{x}) \simeq f_j(\mathbf{x})$ for $\|\delta\mathbf{x}\|$ small enough. Therefore,

$$f_j(\mathbf{x}) \simeq \text{ave} [f_j(\mathbf{x}') | \mathbf{x}' \in R(\mathbf{x})] \quad (14)$$

where $R(\mathbf{x})$ is a subregion of R^p constrained to contain points \mathbf{x}' “close” to \mathbf{x} . This motivates the estimates

$$\hat{f}_j(\mathbf{x}) = \text{ave} [y_{nj} | \mathbf{x}_n \in R(\mathbf{x})] \quad (15)$$

on the training data (12). For the dummy response outputs (9) this reduces to

$$\hat{f}_j(\mathbf{x}) = \#\{g_n = j \in R(\mathbf{x})\} / \#R(\mathbf{x}) \quad (16)$$

where the numerator is the number of class j labels among the training observations (5) in $R(\mathbf{x})$, and the denominator is the total number of observations in the region. A particular kernel method is defined by how the region $R(\mathbf{x})$ is specified.

K -nearest-neighbor kernel methods (“ K -NN”) define a metric q -norm distance on the Euclidean space of input measurement variables

$$D_q(\mathbf{x}, \mathbf{x}_n) = \left\{ \sum_{i=1}^p \|[\mathbf{W}(\mathbf{x})(\mathbf{x} - \mathbf{x}_n)]_i\|^q \right\}^{1/q} \quad (17)$$

and define the region (at \mathbf{x}) to be one that contains exactly the K closest (smallest distance) training observations to \mathbf{x} . The region is therefore specified by a value for K

and choice of a distance measure, which in turn is determined by a norm $q \succ 0$ and a metric defined by the matrix $\mathbf{W}(\mathbf{x}) \in R^{p \times p}$.

The initial popularity of the K -nearest-neighbor method stemmed from its asymptotic properties. As the training sample size becomes arbitrarily large one has

$$\lim_{N \rightarrow \infty} \hat{f}(\mathbf{x}) = f_j(\mathbf{x}) \quad (18)$$

provided the value for K is chosen (as a function of N) so that

$$\lim_{N \rightarrow \infty} K = \infty, \quad \lim_{N \rightarrow \infty} K/N = 0 \quad (19)$$

[Fix and Hodges (1951)]. These results imply that asymptotically ($N \rightarrow \infty$) the K -NN rule achieves the minimum risk of the Bayes rule (1) (2), provided the value of K is chosen appropriately, for any choice of norm q or metric $\mathbf{W}(\mathbf{x})$. This suggests that for finite sample sizes ($N \prec \infty$) these choices may not be too crucial. Furthermore, another asymptotic result [Cover(1968)] suggests that the choice of a value for K may not matter too much either. Let E_1 be the classification error rate (4) of the 1-NN rule and E_∞ that of the Bayes rule. Then $\lim_{N \rightarrow \infty} E_1 = e$, where $E_\infty \leq e \leq E_\infty(2 - e) \leq 2E_\infty$, so that in the asymptotic limit no decision rule is more than twice as accurate as the 1-NN rule. This includes a K -NN rule for any value of K . These asymptotic results together suggest that a 1-NN rule based on simple Euclidean distance ($q = 2$, $\mathbf{W}(\mathbf{x}) = \mathbf{I}_p$)

$$D_2(\mathbf{x}, \mathbf{x}_n) = \left[\sum_{i=1}^p (\mathbf{x}_i - \mathbf{x}_{ni})^2 \right]^{1/2} \quad (20)$$

might perform well provided the training sample is not too small.

3.1 Curse-of-dimensionality

These asymptotic results rest on the fact that the bias in the estimates of each $f_j(\mathbf{x})$

$$\text{bias } \hat{f}_j(\mathbf{x}) = f_j(\mathbf{x}) - E \hat{f}_j(\mathbf{x}) \quad (21)$$

becomes vanishingly small. This will be the case in the asymptotic limit (provided $f_j(\mathbf{x})$ is continuous at \mathbf{x} and $K/N \rightarrow 0$) because the region $R(\mathbf{x})$ will only contain training points \mathbf{x}_n arbitrarily close to \mathbf{x} . If the training sample N is large and the number of input variables p is small then these asymptotic results may be valid. However, for a moderate to large number of inputs, the sample sizes required for their validity are usually beyond feasibility.

This is illustrated by examining the expected diameter of a K -nearest-neighborhood in the input measurement space. Consider a random sample of size N drawn from a uniform distribution in the p -dimensional unit hypercube. The expected diameter of a $K = 1$ neighborhood using Euclidean distance (20) is

$$d_1(p, N) = 2 \left[\frac{p\Gamma(p/2)}{2\pi^{p/2}N} \right]^{1/p}. \quad (22)$$

This shows that for a given p , the diameter of the neighborhood containing the closest training point shrinks as $N^{-1/p}$ for increasing N . Table 1 shows the value of (22) for various values of p and N .

Table 1

p	N	$d_1(p, n)$
4	100	0.42
4	1000	0.23
6	100	0.71
6	1000	0.48
10	1000	0.91
10	10000	0.72
20	10000	1.51
20	10^6	1.20
20	10^{10}	0.76

Keeping in mind that the entire range of each variable is 1.0, one sees that for even moderate number of input variables very large training sample sizes are required to make even a $K = 1$ nearest-neighborhood relatively small. This effect is often referred to as the “curse-of-dimensionality” and has as a consequence that the bias (21) can be quite large even for the smallest possible value of K and operationally very large data sets. Therefore, asymptotic results need not be valid and can often be misleading. For example it is often the case that a 1-NN rule is far more than twice as inaccurate as the Bayes rule for a given problem. Although derived here for the $q = 2$ norm (20) this curse-of-dimensionality (diameter $\sim N^{-1/p}$) inflicts all $q > 0$ norms.

One immediate consequence of having to deal with finite sized data sets is that the choice of metric $\mathbf{W}(\mathbf{x})$ can effect performance. It was recognized early that the influence of each input measurement variable x_i on the distance (20) is proportional to the dispersion of its values over the training data. Changing the scale of a variable by for instance expressing it in different units, changes the contribution of that variable to the distance calculation, and thereby its influence on the classification assignment. For example, a variable characterizing distance would have much more influence when expressed in angstroms rather than in light years. This artifact is generally considered undesirable and in the absence of any other information is usually treated by rescaling each input variable x_i by a factor s_i so that the rescaled dispersions of all variables are the same. Commonly used scale factors are based on the standard deviation

$$s_i = \left[\frac{1}{N} \sum_{n=1}^N (x_{ni} - \bar{x}_i)^2 \right]^{-1/2} \quad (23)$$

the extreme range

$$s_i = [x_{(N)i} - x_{(1)i}]^{-1} \quad (24)$$

or a more robust scale measure such as the interquartile range

$$s_i = [x_{(3N/4)i} - x_{(1/4)i}]^{-1}. \quad (25)$$

Scaling the variables in this manner is equivalent to using the (constant) metric

$$\mathbf{W} = \text{diag}\{s_1, \dots, s_p\} \quad (26)$$

in (17) for all prediction points \mathbf{x} . Although this choice does not mitigate the curse-of-dimensionality it at least renders the rule independent of the particular measurement units employed for each variable, and causes them to contribute equally to the classification decision.

Although the effect of the curse-of-dimensionality would seem to rule out nearest-neighbor methods as being competitive with other classification techniques, this turns out not to be the case in practice. In many benchmark studies simple nearest neighbor methods are often among the best performers. There may be many reasons for this but two can be readily identified. First, the number of input variables represents only an upper bound on the intrinsic dimensionality of the data. Often there is a high degree of association among the inputs so that the data lie near an (often much) lower dimensional manifold within the p -dimensional measurement space. Provided the curvature of the manifold is not too large, distance computed on the full dimensional space is (at least roughly) monotone in distance within the manifold, and it is the (intrinsic) dimensionality of the manifold that governs the curse-of-dimensionality. To the extent that this intrinsic dimensionality is much smaller than the number of inputs the curse is thereby mitigated.

A second (perhaps more important) reason for the resistance of nearest-neighbor methods to the curse-of-dimensionality has to do with the nature of the classification problem. Accurate estimates (6) of the conditional probabilities (10), though sufficient, are not necessary for accurate classification (7). For example, with the simple loss matrix (4) all that is required to obtain the optimal decision (3) is that the largest estimated class probability correspond to the one that is actually the largest

$$\max_{1 \leq j \leq J} \hat{f}_j(\mathbf{x}) = \max_{1 \leq j \leq J} f_j(\mathbf{x}) \quad (27)$$

irrespective of their actual values, or the values (or order) of the estimated probabilities for the other classes, so long as (27) holds. It can often be the case that the bias (21), although very large, effects each of the $\hat{f}_j(\mathbf{x})$ in roughly the same way so that their order relation is similar enough to that of the true underlying probabilities $\{f_j(\mathbf{x})\}_1^J$ to realize an optimal class assignment. This effect is also at least partially responsible for the fact that classification techniques that are very different in concept tend to give rise to very similar error rates.

3.2 Variable subset selection

Although K -NN methods are more resistant to the curse-of-dimensionality than might be expected, they are not immune to it. The requirement of large neighborhoods (22) can effect the bias (21) differentially for the respective class probability estimates enough to cause nonoptimal assignment, thereby increasing misclassification risk. Bias can be reduced by taking advantage of the fact that (at least locally) the class probability functions (10) may not vary with equal strength or in the same manner in all directions in the measurement space emanating from the prediction point \mathbf{x} . This is accomplished by choosing a metric $\mathbf{W}(\mathbf{x})$ (17) that gives the most influence to those directions on which the dependence of the class probability functions are most different and correspondingly less influence to other directions. The resulting shape of the K -NN neighborhood will be thereby elongated along the deemphasized directions and constricted along the influential ones.

One (simple) example of this type of situation is where there exists a subset S of the input variables upon which the class probability functions (10) all have a very similar dependence (for example, no dependence at all). In this case differential bias can be reduced by using a metric that gives no influence to the “irrelevant” variables $x_i \in S$ by extending the range of the K -NN neighborhood to that of the entire training set along those variables. This removes the irrelevant variables from the problem, thereby reducing the dimensionality. This in turn reduces the diameter (22) of K -NN neighborhood in the subspace of relevant variables lowering the bias. This is the motivation for variable subset selection techniques.

Nearest-neighbor methods with variable subset selection consider constant (independent of \mathbf{x}) metrics of the form

$$\mathbf{W} = \text{diag} \{s_i I(x_i \in S)\}_1^p \quad (28)$$

where $\{s_i\}_1^p$ are the (predetermined) scale factors (23 - 25) and $I(\cdot)$ has the value one if its argument is true and zero otherwise. The goal is to choose the subset S that best improves misclassification risk. This is a combinatorial optimization problem, that for a large number of variables p , represents a formidable computation. However heuristic search techniques have been developed for achieving good (suboptimal) solutions that produce lower misclassification risk than K -NN methods using all of the variables in some situations [see Langley (1994)].

Although variable subset selection (28) is often worthwhile its flexibility to adapt to particular situations is clearly limited. It only considers metrics based on diagonal matrices so that only axis oriented directions are eligible for increased (decreased) influence. Furthermore only two levels of influence are considered: either no influence at all (“irrelevant”) or equal influence with the others that have been selected (“relevant”). Finally, the metric (relevant subset) is taken to be the same for all prediction points \mathbf{x} in the input measurement space. It is easy to imagine a wide variety of situations where one or more of these restrictions can limit the ability to exploit special characteristics of a problem to improve misclassification risk.

3.3 Recursive partitioning

Recursive partitioning methods [Morgan and Sonquist (1963)], [Friedman (1977)], [Breiman, Friedman, Olshen and Stone (1984)], [Quinlan (1993)] can be viewed as attempts to overcome the limitations of simple variable subset selection with kernel methods. Like kernel methods they employ local averaging (15) (16) to estimate the class probabilities (10) for the decision rule (7). However, the region $R(\mathbf{x})$ over which the averaging takes place is constructed in a highly adaptive manner. One starts with the entire measurement space R_0 and divides (“splits”) it into two subregions R_l and R_r on one of the input variables x_i at some point s :

if $\mathbf{x} \in R_0$ then
 if $x_i \leq s$ then $\mathbf{x} \in R_l$
 else $\mathbf{x} \in R_r$

The particular variable i and location s used to define the split are those that jointly minimize a criterion closely related to the average misclassification risk associated with using R_l and R_r as the regions $R(\mathbf{x}_n)$ (7) (16) to classify all training observations that respectively lie within them ($\mathbf{x}_n \in R_l$ or $\mathbf{x}_n \in R_r$). This procedure is then recursively applied to separately split R_l and R_r (by replacing R_0 with R_l and R_r respectively) creating four regions. Each of these four is similarly split, and the splitting continues until all regions contain training observations of the same class. After the splitting is completed a “pruning” procedure is usually applied that recursively recombines adjacent regions in a bottom-up manner using cross-validated misclassification risk to determine when to stop the pruning. The resulting set of regions represent a disjoint partition of the input measurement space. The region (16) used for estimating the class probabilities and making the assignment (7) for any prediction point \mathbf{x} is the resulting region in which the point lies.

Recursive partitioning methods inherently perform variable subset selection. At each step in the recursive subdivision the (estimated) best variable is used for splitting thereby

reducing the extent of the neighborhood along the chosen axis for all successive regions that are descendants of the one being split. Irrelevant input variables that have little or no classification information are less likely to be chosen for splitting. Moreover, this selection process is local. As the partitioning proceeds successive regions span smaller and smaller volumes of the input measurement space. The decision as to which variable to choose for splitting is based solely on the (local) data contained in each such region. Thus, recursive partitioning methods can exploit “local relevance” of input variables at different prediction points \mathbf{x} in the measurement space. Although there may be many relevant variables globally, a (different) smaller subset may be most relevant locally at different locations \mathbf{x} . Some recursive partitioning implementations add locally (within region) derived variables that are functions of the original input variables (such as linear combinations) [Breiman, et. al. (1984)]. These are included along with the original inputs as candidates for splitting, allowing regions with non-axis oriented boundaries to be produced. Finally, if pruning is employed recursive partitioning methods also estimate the best region size (K) locally for different prediction points \mathbf{x} .

With all of this additional flexibility one might expect recursive partitioning methods to outperform K -NN methods (with or without variable subset selection) in nearly all situations. As noted above this is not the case. In benchmark studies even the simplest K -NN rules often outperform recursive partitioning methods. Among the possible reasons for this, several can be readily identified. First, the training sample size N restricts the number of input variables (splits) that can be used to limit the extent of a subregion. As the partitioning proceeds the number of training observations in successive regions becomes smaller and smaller. After a relatively small number of splits the data remaining is insufficient to provide meaningful estimates. For example, if splits on average divide the data in half each time then the number of splits (and thus the maximum number of variables) used to define each region is on the order of $\log_2 N$. If the number of locally relevant variables is larger than this, recursive partitioning is at a disadvantage. By contrast, K -NN methods have the ability to involve many or all of the input variables in defining (restricting) the extent of the region centered at the prediction point \mathbf{x} .

A second limitation of the recursive partitioning technique is the piecewise-constant nature of the probability estimates (6) that it produces. Although different subregions are used in different parts of the measurements space, those regions are fixed by the training algorithm and (at best) are partially customized for each individual (future) prediction point \mathbf{x} . The partial customizing involves selecting the subregion in which the prediction point lies. All points that lie in the same region have the same probability estimates and thus are assigned to the same class irrespective of their location within the region. The goal is to choose regions that are good as averaged over all points in the region. Since the training sample size prevents the subregions from becoming arbitrary small, the shape (and size) of each resulting region represents a compromise choice to be applied to all points within it, and may not represent the best choice for all such points individually, especially those far from the center near the region boundaries. Furthermore, the top down (greedy) nature of the splitting process forces early splits (which help define boundaries for later regions) to be compromises based on global information only, without access to the local information available later when the final subregions are being formed. These early splits help form the boundaries of several to many final subregions and thus cannot be simultaneously optimal for all of them.

By contrast K -NN methods produce (nearly) continuous rather than piecewise-constant probability estimates (15) (16). They use a different region for each individual prediction point \mathbf{x} (centered at \mathbf{x}) so that, to the extent possible, all training observations in the region are close to the point being predicted. These positive aspects of the K -NN technique help mitigate its lack of flexibility to customize the size and shape of the region,

often resulting in comparable or better prediction performance than that achieved by (the more flexible) recursive partitioning methods.

4 Flexible metric K -nearest-neighbor rules

The goal of this approach is to produce classification methods that include the attractive properties of both K -NN and recursive partitioning methods, while trying to avoid their respective limitations. The resulting methods are a hybrid between the two that combine the flexibility of recursive partitioning to adapt the shape of the region $R(\mathbf{x})$ (16) with the ability of nearest-neighbors to keep the points within the region close to the point being predicted. The result produces (nearly) continuous probability estimates (6) (16) with the region $R(\mathbf{x})$ centered at \mathbf{x} and the shape of the region separately customized for each such individual prediction point.

4.1 Local relevance

In order to customize the shape of the region $R(\mathbf{x})$ (16) to exploit differential relevance of the input measurement variables to class assignment, a way to estimate such relevance is needed. Consider an arbitrary function $f(\mathbf{x})$ of p arguments $\mathbf{x} = (x_1, \dots, x_p)$. In the absence values for any of the argument variables the least-squares estimate for predicting $f(\mathbf{x})$ is just its expected (average) value

$$E f = \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \quad (29)$$

over the joint probability density $p(\mathbf{x})$ of its arguments. Suppose the value of just one of the argument variables x_i were known, say $x_i = z$. The least-squares prediction for $f(\mathbf{x})$ in this case would be the expected value of $f(\mathbf{x})$, under the restriction that x_i assumes the known value z

$$E[f | x_i = z] = \int f(\mathbf{x}) p(\mathbf{x} | x_i = z) d\mathbf{x}. \quad (30)$$

Here $p(\mathbf{x} | x_i = z)$ is the probability density distribution of the other input variables $\{x_k\}_{k \neq i}$

$$p(\mathbf{x} | x_i = z) = p(\mathbf{x}) \delta(x_i - z) / \int p(\mathbf{x}') \delta(x'_i - z) d\mathbf{x} \quad (31)$$

where $\delta(x_i - z)$ is the Dirac “delta” function defined by

$$\delta(x_i - z) = \begin{cases} \infty & x_i = z \\ 0 & x_i \neq z \end{cases} \quad (32)$$

and

$$\int_{-\infty}^{\infty} \delta(x_j - z) dx_j = 1. \quad (33)$$

The improvement in squared prediction error $I_i^2(z)$ associated with knowing the value z of the i th input variable $x_i = z$ is therefore

$$I_i^2(z) = E \left[(f(\mathbf{x}) - E f)^2 | x_i = z \right] - E \left[(f(\mathbf{x}) - E[f(\mathbf{x}) | x_i = z])^2 | x_i = z \right] \quad (34)$$

which reduces to

$$I_i^2(z) = (E f - E[f | x_i = z])^2. \quad (35)$$

In some simple situations (35) can be easily calculated. For example, suppose the input variables are distributed independently of each other

$$p(\mathbf{x}) = \prod_{i=1}^p p_i(x_i) \quad (36)$$

and $f(\mathbf{x})$ is a linear function of its arguments

$$f(\mathbf{x}) = a_0 + \sum_{i=1}^p a_i x_i. \quad (37)$$

Then one has

$$I_i^2(z) = a_i^2 (z - \bar{x}_i)^2 \quad (38)$$

with

$$\bar{x}_i = \int x_i p(x_i) dx_i \quad (39)$$

being the average value of the i th input variable. A bit more generally, suppose (36) and $f(\mathbf{x})$ is an additive function of its arguments

$$f(\mathbf{x}) = \sum_{i=1}^p f_i(x_i). \quad (40)$$

In this case

$$I_i^2(z) = [f_i(z) - E f_i]^2. \quad (41)$$

Although these situations (36) (37) (40) are far from general, they do suggest that (35) reflects the influence of the i th input variable on the variation of $f(\mathbf{x})$ at the particular point $x_i = z$.

Consider an arbitrary point $\mathbf{z} = (z_1, \dots, z_p)$ in the p -dimensional input measurement space. A measure of the relative influence (“relevance”) of the i th input variable x_i to the variation of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{z}$ is given by

$$r_i^2(\mathbf{z}) = I_i^2(z_i) \Big/ \sum_{k=1}^p I_k^2(z_k). \quad (42)$$

It has the value $r_i^2(\mathbf{z}) = 0$ when $f(\mathbf{x})$ is independent of x_i (at \mathbf{z}) and the value $r_i^2(\mathbf{z}) = 1$ when $f(\mathbf{x})$ depends only on x_i there. Values in between these extremes reflect varying degrees of relevance. It (42) can be viewed as a measure of “local” relevance in the sense that its value depends on the particular value $\mathbf{x} = \mathbf{z}$ at which it is evaluated. There may be some points \mathbf{z} for which the i th input variable is relatively more informative concerning the value of $f(\mathbf{x})$ than at other points in the input measurement space. However this measure still has a global flavor in that the other inputs $\{x_k\}_{k \neq i}$ are unrestricted. The measure (35) (42) can be made more local by causing it to reference a subregion $R(\mathbf{z})$ of the input measurement space, that includes the point \mathbf{z} , by restricting all expected values leading to its definition to be taken only over that subregion. That is $p(\mathbf{x})$ is replaced by

$$p(\mathbf{x} | \mathbf{x} \in R(\mathbf{z})) = p(\mathbf{x}) 1(\mathbf{x} \in R(\mathbf{z})) \Big/ \int p(\mathbf{x}') 1(\mathbf{x}' \in R(\mathbf{z})) d\mathbf{x}' \quad (43)$$

in all definitions, where $1(\cdot)$ takes the value one when its argument is true and zero otherwise. In this case the corresponding quantity $r_i^2(\mathbf{z} | R(\mathbf{z}))$ reflects the relevance of the i th input variable to the variation of $f(\mathbf{x})$ at the point \mathbf{z} within the region $R(\mathbf{z})$.

4.2 Estimation

Since the target function $f(\mathbf{x})$ is unknown a way must be found to estimate (35) from the training data (12), if (42) is to be of use in practice. This is accomplished by observing that $f(\mathbf{x}) = \mathbb{E}(y | \mathbf{x})$ (10). Therefore (35) can be equivalently expressed as

$$I_i^2(z) = (\mathbb{E}y - \mathbb{E}[y | x_i = z])^2 \quad (44)$$

either for the entire measurement space or any subregion $R(\mathbf{z})$ (43) contained therein. The quantity $\mathbb{E}y$ is estimated by its average

$$\hat{\mathbb{E}}y = \bar{y} = \frac{1}{N} \sum_{n=1}^N y_n \quad (45)$$

over the whole measurement space, or within a subregion $R(\mathbf{z})$ by

$$\hat{\mathbb{E}}[y | R(\mathbf{z})] = \frac{\sum_{n=1}^N y_n 1(\mathbf{x}_n \in R(\mathbf{z}))}{\sum_{n=1}^N 1(\mathbf{x}_n \in R(\mathbf{z}))}. \quad (46)$$

The conditional expectation in (44) cannot be directly estimated in this manner since there may be no (or at most a few) training observations for which $x_{ni} = z$. For this estimate one can use the K -nearest-neighbor technique (15) (16) with $R(\mathbf{x})$ defined to be a small interval on the i th input variable containing the value $x_i = z$. Specifically, the estimate is taken to be

$$\hat{\mathbb{E}}[y | x_i = z] = \frac{\sum_{n=1}^N y_n 1(|x_{ni} - z| \leq \Delta_i)}{\sum_{n=1}^N 1(|x_{ni} - z| \leq \Delta_i)} \quad (47)$$

where the value of Δ_i is chosen so that the interval contains a fixed (small) number L of points

$$\sum_{n=1}^N 1(|x_{ni} - z| \leq \Delta_i) = L. \quad (48)$$

(Choice of a value for L is discussed in Section 4.4.) Note that since only one variable x_i is involved in obtaining the estimate (47) there is no issue of the curse-of-dimensionality here. This estimate (47) can be made to reference a subregion $R(\mathbf{z})$ by replacing the indicator function in (46) (47) (48) with

$$1(|x_{ni} - z| \leq \Delta_i) 1(x_{ni} \in R(\mathbf{z})). \quad (49)$$

Using these estimates (45 - 49) in (44) produces an empirical measure of the relevance (42) of each input variable x_i at the point $\mathbf{x} = \mathbf{z}$.

4.3 Multiple outputs

In the classification problem there are J output variables $\{y_j\}_1^J$ (9), reflecting the J probability functions $\{f_j(\mathbf{x})\}_1^J$ (10), to be simultaneously estimated using the same neighborhood $R(\mathbf{x})$ (15) (16). The relevance measure (42) must therefore be suitably generalized for this situation. One method would be to replace $I_i^2(z)$ (44) by its sum over the J outputs

$$I_i^2(z) = \sum_{j=1}^J (\mathbb{E}y_j - \mathbb{E}[y_j | x_i = z])^2 \quad (50)$$

in the relevance measure (42), reflecting the influence of each input x_i (at $\mathbf{x} = \mathbf{z}$) on the combined variation of all of the target functions. A problem with this direct approach is that each output target function does not influence the composite measure (50) to the same degree. As with all Euclidean type distance measures, the influence of each variable y_j will be proportional to its variance over the measurement space or within a prespecified subregion $R(\mathbf{z})$. One can compensate for this effect by transforming the problem so that all output variables have the same variance as a result of the transformation. In the context of the “dummy” outputs (9) this is accomplished by rescaling their corresponding probability functions (10) to have the same average value

$$\int f_j(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = 1/J, \quad j = 1, J, \quad (51)$$

either globally, or within a particular subregion $R(\mathbf{z})$ under consideration. This causes the respective outputs to y_j to have equal variance as well as expected value over the region

$$\mathbb{E} y_j = 1/J, \quad \text{var } y_j = \frac{1}{J} \left(1 - \frac{1}{J} \right). \quad (52)$$

The empirical (5) (12) analog of (51) is accomplished by giving each training observation a weight

$$w_n = N/(JN_{g_n}) \quad (53)$$

where

$$N_j = \sum_{n=1}^N 1(g_n = j) 1(\mathbf{x}_n \in R(\mathbf{z})) \quad (54)$$

is the number of observations of its class in the region. This weight is used in the computation of all averages (45 - 49) leading to the estimate of each $I_i^2(z)$ in (50).

With the transformation (51), the influence measure (50) reduces to

$$I_i^2(z) = \sum_{j=1}^J \left(\frac{1}{J} - \mathbb{E}[y_j | x_i = z] \right)^2 \quad (55)$$

which is a measure of “purity” (negative “diversity”) of the class labels, conditioned on $x_i = z$. It achieves its minimum value when the conditional expectations (probabilities) are all equal

$$\mathbb{E}[y_j | x_i = z] = \Pr(j | x_i = z) = 1/J, \quad j = 1, J \quad (56)$$

and maximum value when a particular class label is certain (conditioned on $x_i = z$)

$$\mathbb{E}[y_j | x_i = z] = 1, \quad \{\mathbb{E}[y_k | x_i = z] = 0\}_{k \neq j}. \quad (57)$$

Note that the definitions leading to (55) insure

$$\sum_{j=1}^J \mathbb{E}[y_j | x_i = z] = 1. \quad (58)$$

This purity measure (55) is related to the well known “Gini index-of-diversity” [Breiman, et. al. (1984)]

$$G_i(z) = 1 - \sum_{j=1}^J \mathbb{E}^2[y_j | x_i = z] \quad (59)$$

by

$$I_i^2(z) = (1 - 1/J) - G_i(z). \quad (60)$$

The first term on the right side of (60) represents the maximum value for the Gini index so that our influence measure (55) (60) is just the reduction in the value of the Gini index obtained by conditioning on $x_i = z$, from its largest possible value, which from (51 - 53) is its value over the whole region $R(\mathbf{z})$ under consideration.

4.4 The Machete

Given a prediction point $\mathbf{x} = \mathbf{z}$ the K -nearest-neighbor method estimates each $f_j(\mathbf{z})$ (10) by its average over a region $R(\mathbf{z})$ containing \mathbf{z} (15) (16). The combined mean-squared error of these estimates is

$$\sum_{j=1}^J \mathbb{E} \left[f_j(\mathbf{z}) - \hat{f}_j(\mathbf{z}) \right]^2 = \sum_{j=1}^J \left(\mathbb{E} f_j^2(\mathbf{z}) - \mathbb{E}^2 \hat{f}_j(\mathbf{z}) \right) + \sum_{j=1}^J \left[f_j(\mathbf{z}) - \mathbb{E} \hat{f}_j(\mathbf{z}) \right]^2. \quad (61)$$

The first term on the right side of (61) is the variance of the estimates which depends on the number of counts K in the region and is independent of its shape. The second term (bias-squared) on the other hand depends on both K and the shape of the region. Thus, for a given value of K the goal is to choose the shape of the region to minimize the (total) bias-squared, by restricting its extent along those directions in which the $\{f_j(\mathbf{x})\}_1^J$ (10) exhibit the most (differential) variation. This is the purpose of the recursive partitioning strategy outlined in Section 3.3. At each step of its successive refinement procedure the (locally) most relevant input variable is chosen for splitting, where relevance is measured by the average-squared difference between the respective class means (9) (10) (increase in purity) on each side of the split, as estimated from the training data contained in each successive (smaller) region. To the extent that this strategy is successful, the result will be a final set of regions whose width along each axis (inversely) reflects the (differential) strength of the function (10) variation along that axis within the region. As noted in Section 3.3 one limitation with this approach is that the regions so produced are static. They are fixed at training time and not customized to each individual point \mathbf{z} to be predicted. Additionally, the region used for each prediction is not centered at the prediction point. The ‘‘machete’’ is an alternative successive splitting procedure that attempts to overcome these limitations.

As with recursive partitioning, the machete begins with the entire input measurement space R_0 and divides it into two regions by a split on one of the input variables. However, the manner in which the splitting variable is selected, and the nature of the split itself, are quite different. The input variable used for splitting is the one that maximizes the estimated relevance (42) (50) (60) as evaluated at the point \mathbf{z} to be predicted

$$i^*(\mathbf{z}) = \arg \max_{1 \leq i \leq p} \hat{r}_i^2(\mathbf{z}). \quad (62)$$

Thus, for the same training data (5) (12) different input variables can be selected for this (first) split at different prediction points \mathbf{z} , depending on how the relevance of each input variable changes with location in the input measurement space. The space is then split on the i^* th (62) input variable so that (to the extent possible) the i^* th component of \mathbf{z} , z_{i^*} , is centered within the resulting subinterval that contains it. In particular, the training data (5) are sorted in increasing order on $|z_{i^*} - x_{ni^*}|$ and a new region $R_1(\mathbf{z})$ (defined by the data contained within it) is

$$R_1(\mathbf{z}) = \{\mathbf{x}_n \mid |z_{i^*} - x_{ni^*}| \leq d(M_1)\} \quad (63)$$

where $d(M_1)$ is the M_1 th order statistic of $\{|z_{i^*} - x_{ni^*}|\}_1^N$, so that $R_1(\mathbf{z})$ contains $M_1 \prec N$ training observations.

This split divides the input measurement space into two regions. One is $R_1(\mathbf{z})$ (63) that contains the prediction point \mathbf{z} and the M_1 training observations that are closest to it on the chosen (i^* th) input variable. The other (complement) region $\bar{R}_1(\mathbf{z})$ contains the $N - M_1$ remaining observations that are farthest from \mathbf{z} on that variable. This complement region is removed from consideration (for classifying \mathbf{z}) by discarding the data contained within it. Thus the result (output) of this splitting procedure is just the single region $R_1(\mathbf{z})$ (63).

As with all recursive methods, the entire machete procedure is defined by successively applying its splitting procedure to the result of the previous split. In this case a second region $R_2(\mathbf{z})$ (nested within $R_1(\mathbf{z})$ and containing $M_2 \prec M_1$ training points) is obtained by substituting $R_1(\mathbf{z})$ in place of R_0 in the above description. The variable selected for splitting (62) in this second step can, but need not, be the same as that chosen for the first split since $\{\hat{r}_i^2(\mathbf{z})\}_1^p$ (42) are now evaluated only using data contained in $R_1(\mathbf{z})$ (63). This second region $R_2(\mathbf{z})$ is similarly split producing $R_3(\mathbf{z})$, and so on, until there are K training observations left in the region under consideration. This final region is then used to estimate the (K -NN) probabilities (16) at \mathbf{z} . These estimates are in turn inserted into (7) to predict the class assignment.

In our implementation the number of counts M_k (63) in each successive region $R_k(\mathbf{z})$ is taken to be a fixed fraction α of the number M_{k-1} in the previous (parent) region $R_{k-1}(\mathbf{z})$ whose split gave rise to $R_k(\mathbf{z})$

$$M_k = \alpha M_{k-1}, \quad 0 \prec \alpha \prec 1. \quad (64)$$

(Choice of a value for α is discussed below.)

At each (k th) step of the machete procedure a current region $R_k(\mathbf{z})$ is split on the input variable (62) that is estimated to be the most relevant in terms of capturing the variation of the target functions (10) within that region. The expectation is that this choice will cause the greatest reduction in bias when $\{f_j(\mathbf{z})\}_1^J$ are estimated in the next (smaller) region $R_{k+1}(\mathbf{z})$ produced by the split, and thereby in all subsequent regions $R_l(\mathbf{z})$, $l > k$, that are its descendants nested within it. This includes the final descendant region (containing the K points) used for classification. Another consequence of splitting $R_k(\mathbf{z})$ on the i^* th input variable is that the relevance of that variable will tend to be reduced in all of its subsequent descendant regions. This will encourage the choice of other variables for splitting the descendants to the extent that their relevance is greater than the deflated relevance of x_{i^*} . The intervals on each variable defining the boundaries of the final region (used for classification) are determined by the last split on that particular variable, in the sequence of splits producing the final region. These intervals will (to the extent possible) be centered at the prediction point \mathbf{z} and their respective widths will (inversely) reflect the estimated relative strength of the variation of the target functions along the respective axis directions emanating from \mathbf{z} .

The final classification region produced by the machete is an axis-oriented hyperrectangle in the input measurement space. This is the same region that would be produced by a K -NN method based on the $q = \infty$ norm (17) using the metric

$$\mathbf{W}(\mathbf{z}) = \text{diag} \{1/l_1(\mathbf{z}), \dots, 1/l_p(\mathbf{z})\} \quad (65)$$

where $l_i(\mathbf{z})$ is the length of the interval on the i th input variable defining the final region. In this sense the machete can be viewed as a variable metric K -NN method, where the metric is customized for each prediction point \mathbf{z} to reflect the different degrees of (local) target function variation along the axis directions emanating from the prediction point.

Using the $q = \infty$ norm is not a requirement and other norms such as $q = 2$ (Euclidean) can be used in association with the derived metric $W(\mathbf{z})$ (65).

Besides the number of nearest neighbors K used to define the size of the classification region, there are two additional parameters whose values must be specified to apply the machete. One is the number of counts L (48) that define the intervals $\{\Delta_i\}_1^p$ used to obtain the conditional expectation estimates $\{E[y_j | x_i = z_i]\}_1^p$ (47) in each region $R_k(\mathbf{z})$. The value of L determines the bias-variance trade-off for these estimates. The way in which these estimates are used in the machete does not require that they be estimated with high precision so that its performance is insensitive to the value chosen for L provided it is not too small ($L \simeq 1$), nor too large ($L \simeq N$). In all applications reported in Section 5 the value $L = 20$ was used.

The second parameter for which a value must be specified is α (64). It controls the rate at which the sample size decreases in the sequence of regions produced by the successive splitting. This in turn controls the total number of splits

$$n_s = \log_{1/\alpha}(N/K) \quad (66)$$

leading to the final region (containing K points) used for classification. Since the computation associated with the machete procedure is directly proportional to the number of splits n_s , computational considerations suggest a small value for α . On the other hand the total number of splits places an upper limit on the number of input variables $p_s \leq p$ whose range can be restricted in forming the final region

$$p_s \leq \min(p, n_s). \quad (67)$$

If the number of highly relevant variables (at a prediction point \mathbf{z}) is large compared to p_s (67) then prediction accuracy can suffer since the machete can only include the subset of these estimated to be the most relevant. For example suppose in a particular situation all p input variables were equally relevant. In this case the simple K -NN procedure (17) (26) would be optimal, whereas the machete (using a small value for α) would be highly suboptimal since one would have $p \gg p_s$ (66) (67). These statistical considerations argue for larger values of α , creating tension between fast computation and prediction accuracy in some situations.

The machete represents a hybrid approach to classification combining aspects of both the K -NN and recursive partitioning methodologies. The goal is a procedure that includes the best parts of those methodologies, resulting in performance comparable to the best of the two for most problems, and better than either for some. The results on both simulated and real data presented in Section 5 provide optimism that such is the case, especially for problems in which the number of locally relevant variables is small (66) (67). However the “basic” machete (described above) has limitations that can tend to hinder its performance. One is its inability to include a large number of locally relevant variables when the situation requires it without excessive computation. Another limitation is that the splits are restricted to the input variables. This limits the shape of the final classification region to be hyperrectangular, or equivalently, the derived metric $\mathbf{W}(\mathbf{x})$ (17) is restricted to a diagonal matrix (65). Sections 4.5 and 4.6 (below) describe modifications to the basic machete intended to mitigate these limitations, and as seen in Section 5, result in generally superior performance.

4.5 Derived variables

The basic machete splits each successive region $R_k(\mathbf{z})$ on the locally most relevant input variable at \mathbf{z} (62), as estimated in that region, to produce the next (smaller) one

$R_{k+1}(\mathbf{z})$. There is no fundamental reason why additional “derived” variables $\{d_i\}_1^D$ (features) cannot be incorporated into the list of those eligible for splitting, where the derived variables are functions of the original input variables $\{d_i = g_i(\mathbf{x})\}_1^D$. Including such derived variables is common practice in the application of recursive partitioning. If the derived variables tend to be more informative than the original inputs they will be preferentially selected for splitting, otherwise they will be largely ignored. Including derived variables along with the original inputs with the K -NN procedure is more problematic. If one or more of them are highly informative their effect is diluted by the presence of all the original inputs. On the other hand, if they are not sufficiently informative, they serve to increase the bias through the curse-of-dimensionality since they add to the variable count. However, including derived variables in a K -NN procedure with variable subset selection can be beneficial.

There is a special aspect of the machete that makes adding derived variables especially attractive; namely, they can be customized to the particular point \mathbf{z} where the prediction is to be made. That is, the functions

$$\{d_i = g_i(\mathbf{x} | \mathbf{z})\}_1^D \quad (68)$$

can depend on \mathbf{z} , thereby being different functions for different prediction locations in the input space.

There are a wide variety of possibilities for constructing derived variables. One such is the (scaled) Euclidean ($q = 2$) distance-squared (17) (26)

$$d_1 = g_1(\mathbf{x} | \mathbf{z}) = \sum_{i=1}^p [s_i \cdot (x_i - z_i)]^2. \quad (69)$$

Using d_1 as the only variable corresponds to the usual K -NN procedure. If all of the original inputs $\{x_i\}_1^p$ were roughly equally relevant (locally at \mathbf{z}) then d_1 would represent the most relevant variable among it and the original inputs, thereby being very likely to be chosen for splitting. On the other hand, if there were great dispersion in relevance among the original inputs, the relevance of d_1 (69) would be diluted compared to the most relevant input, and the latter would likely be chosen for splitting. Including (69) as a derived variable allows the machete, even with small values for α (64), greater power to deal with situations in which, at least for some locations \mathbf{z} , many or all of the original input variables are roughly equally relevant. There may be other locations (for the same problem) where only a few of the original inputs are highly relevant. For those other locations the machete will tend to concentrate on the few highly relevant original inputs, largely ignoring the derived distance variable (69).

Other derived variables, complementary to the distance d_1 (69), can be based on linear combinations of the original inputs

$$d = g(\mathbf{x} | \mathbf{z}) = \sum_{i=1}^p a_i(\mathbf{z}) x_i. \quad (70)$$

This type of variable can also involve all of the original inputs but in a different way. It singles out a particular direction in the input measurement space characterized by the vector $\mathbf{a}(\mathbf{z}) = \{a_1(\mathbf{z}), \dots, a_p(\mathbf{z})\}$ for special consideration. The distance (69) gives equal weight to all directions. The goal is to choose the coefficients $\{a_i(\mathbf{z})\}_1^p$ (locally at \mathbf{z}) so as to render d (70) highly relevant.

One strategy would be to directly maximize the relevance (42) (44) with respect to the coefficients $\{a_i(\mathbf{z})\}_1^p$. In this case d would automatically become the most relevant variable among it and the original inputs $\{x_i\}_1^p$ since the later are simply special cases

of d obtained by setting all the variable coefficients but one equal to zero. However, a direct numerical optimization would likely present too large a computational burden in many environments. This dictates the use of approximate methods that are likely (but not guaranteed) to produce linear combinations of increased relevance for predicting \mathbf{z} . In this case one relies on the machete strategy to choose the most relevant variable among the linear combination and the original inputs. If the approximate method succeeds in producing a linear combination with higher relevance, it will be chosen for the split. Otherwise it will be ignored in favor of the most relevant single input, or perhaps the distance variable d_1 (69).

Candidates for producing good (as opposed to optimal) linear combination variables can be based on discriminant analysis [see McLachlan (1992)]. Linear discriminant analysis (LDA) assumes that the covariance matrices associated with the probability density distributions $\Pr(\mathbf{x} | j)$ (8) of each class are the same so that most of the discriminating information is contained in the respective interpoint distances between the class means, using the common class covariance matrix as a metric. LDA produces an ordered set of vectors

$$\{\mathbf{a}_1, \dots, \mathbf{a}_{\min(J-1, p)}\} \quad (71)$$

that span the $\min(J-1, p)$ - dimensional (sub)space containing the class means. The vectors are ordered so that in the subspace spanned by the first M of them, the average squared distance between the class means is maximum (using the common covariance as a metric) among all linear M -dimensional linear subspaces, for all $1 \leq M \leq \min(J-1, p)$.

LDA can be used to produce a set of derived variables by simply incorporating the corresponding linear combinations derived from the discriminant vectors (71)

$$\{d_i = \mathbf{a}_i^t \mathbf{x}\}_1^{\min(J-1, p)} \quad (72)$$

into the list of candidates eligible for splitting. To the extent one believes that their LDA ordering actually reflects the order of their relevance for predicting the point \mathbf{z} , one might restrict the list to the first few to save computation. At each successive k th step in the machete splitting process the LDA computation is confined to the data contained in the current region $R_k(\mathbf{z})$. This helps insure that the corresponding derived variables (72) address local relevance.

Another way to employ LDA in an attempt to produce good (local) derived variables is to focus on each individual class rather than treating them collectively. Specifically, consider J partitions of the class labels, each isolating one of the classes j from all of the other classes. For each such partition assign the class j observations the label “A” and all of the others the label “B”, and compute a two-class linear discriminant vector \mathbf{a}_j between the A and B classes. The collection of all J such (two-class) discriminant functions

$$\{h_j(\mathbf{x}) = \mathbf{a}_j^t \mathbf{x}\}_1^J \quad (73)$$

are then included (along with the original inputs) as eligible candidates for splitting.

A reason one might prefer this approach (73) to that of (72) is that the (unknown) class $j(\mathbf{z})$, of the point \mathbf{z} to be predicted, is known to be one of the possible classes $j(\mathbf{z}) \in \{1, \dots, J\}$. Therefore one might expect that the discriminant function (73) corresponding to its class $h_{j(\mathbf{z})}(\mathbf{x})$ will be highly relevant for separating out the $j(\mathbf{z})$ th class and thereby correctly predicting the class of \mathbf{z} . To the extent that the relevance measure (42) (44) captures this, $h_{j(\mathbf{z})}(\mathbf{x})$ will likely be chosen for splitting. By contrast, ordinary LDA (72) does not focus on individual classes but instead is driven by an aggregate measure of separation among all the classes. Thus, it can often be the case that the dominant LDA directions (71) are being most influenced (and thus separate) classes other than the

actual (unknown) class $j(\mathbf{z})$ of the point being predicted. Note that for the case of only two classes ($J = 2$) both approaches (72) and (73) are equivalent.

As with ordinary LDA (72), the discriminant functions (73) can be ordered and only the first few included to save computation. The ordering is based on their decreasing values of $\{h_j(\mathbf{z})\}_1^J$ when evaluated at the prediction point \mathbf{z} .

One can go beyond linearity in forming discriminant functions (73) to be used as derived variables. For example, quadratic discriminant analysis (QDA) [see McLachlan (1992)] produces quadratic functions

$$\{q_j(\mathbf{x}) = (\mathbf{x} - \bar{\mathbf{x}}_j)^t \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \bar{\mathbf{x}}_j) + \log |\boldsymbol{\Sigma}_j|\}_1^J \quad (74)$$

where $\bar{\mathbf{x}}_j$ is the mean vector and $\boldsymbol{\Sigma}_j$ the covariance matrix of the class j training observations, computed either globally or locally within the current region $R_k(\mathbf{z})$. Like (73) these focus on the individual classes and the expectation would be that the one $q_{j(\mathbf{z})}(\mathbf{x})$ corresponding to the true (unknown) class of \mathbf{z} would have preferentially small values for the class $j(\mathbf{z})$ observations. If this turned out to be the case then $q_{j(\mathbf{z})}(\mathbf{x})$ would be highly relevant for separating that class from the others and be a likely candidate for splitting. As with those of LDA, the quadratic discriminant functions (74) can be ordered on (increasing) values of $\{q_j(\mathbf{z})\}_1^J$, as computed at \mathbf{z} , and only the first few included to save computation.

As the above discussion indicates there are a wide variety of possibilities for constructing derived variables to be included along with the original input variables as eligible candidates for splitting. The motivation for this is that if one or more of them turn out to be highly (locally) relevant they will tend to participate in the splitting giving rise to (sometimes substantially) reduced misclassification risk. If not, then they will tend to be ignored causing at most a mild degradation in performance. The results presented in Section 5 provide optimism that this is the case. From the point of view of the machete, including derived variables allows it to involve potentially a large number of the original input variables (in a highly structured way) to define the splits, and thereby in defining the final classification region. It also allows the machete to produce classification regions of far more complex shape than simply hyperrectangular.

4.6 The Scythe

The machete uses a “winner-take-all” splitting strategy. In each successive region $R_k(\mathbf{z})$ it chooses the variable (derived or original input) estimated to be the most relevant in that region at the prediction point \mathbf{z} . If there are other variables that have relevance close to that of the most relevant one, the machete ignores them and assigns all of the relevance to the highest one in defining the split. Since splitting on a particular variable deflates its relevance for all subsequent regions that are its descendants, the hope is that the other highly relevant variables (not strongly correlated with the split variable) will become successful candidates for later splits, thereby eventually becoming involved in the definition of the final classification region. Variables that are highly correlated with the splitting variable will have their relevance also deflated by the split. A problem with this strategy is that for an acceptable value of α (64) there may not be enough splits n_s (66) remaining to involve all of these other highly relevant variables, and performance may thereby suffer. Adding derived variables helps mitigate this problem by allowing parameterized variable combinations to help define the splits. Intuitively, it seems reasonable that if there are several highly relevant (input or derived) variables they should all participate in defining the split.

The “scythe” employs an alternative splitting strategy in which the respective variables influence each split in proportion to their estimated (local) relevance, rather than the

winner-take-all strategy of the machete. Specifically, a ($q = \infty$) distance measure is defined between the prediction point \mathbf{z} and the training points \mathbf{x}_n in the current region $R_k(\mathbf{z})$ under consideration

$$d_\infty(\mathbf{x}_n, \mathbf{z}) = \max_{1 \leq i \leq p} |w_i(\mathbf{z})(x_{ni} - z_i)| \quad (75)$$

to be used in place of the single variable distance $|z_{i^*} - x_{ni^*}|$ in the definition (63) of the next smaller region. That is,

$$R_{k+1}(\mathbf{z}) = \{\mathbf{x}_n \mid \mathbf{x}_n \in R_k(\mathbf{z}) \ \& \ d_\infty(\mathbf{x}_n, \mathbf{z}) \leq d(M_{k+1})\}. \quad (76)$$

The weights $\{w_i(\mathbf{z})\}_1^p$ are taken to be (monotonic) functions of the corresponding relevance estimates (42) (44) of the respective variables

$$w_i(\mathbf{z}) = m_i(r_i(\mathbf{z})). \quad (77)$$

Optimal choices for the functions $\{m_i(\cdot)\}_1^p$ depend on the detailed properties of the target functions $\{f_j(\mathbf{x})\}_1^J$ (10) and the data density distribution $p(\mathbf{x})$ within the current region $R_k(\mathbf{z})$. In highly idealized situations they can be determined. For example, if all of the target functions are linear (37) and $p(\mathbf{x})$ is uniform over $R_k(\mathbf{x})$ then the optimal choice for each $m_i(\cdot)$ is the identity function. That is $\{w_i(\mathbf{z}) = r_i(\mathbf{z})\}_1^p$. If all of the target functions are additive (40) and the functions associated with each individual variable are the same simple power function $\{f_i(x_i) = a_i x_i^s\}_1^p$ then $\{w_i(\mathbf{z}) = r_i^{1/s}(\mathbf{z})\}_1^p$ is optimal. Since these settings are fairly unrealistic and the true target functions are generally unknown the strategy adopted for the scythe is to take

$$\left\{w_i(\mathbf{z}) = r_i^\beta(\mathbf{z})\right\}_1^p \quad (78)$$

where the value of $\beta \geq 0$ is user selected. Setting $\beta = \infty$ gives total weight to the most relevant variable and thus produces the machete. At the other extreme $\beta = 0$, all variables receive equal weight, regardless of their estimated relevance, thereby producing the ordinary ($q = \infty$) K -NN procedure. Values of β between these extremes give rise to a spectrum of procedures ranging from K -NN to the machete.

Choice of a value for β depends upon the user's intuition concerning the number of variables that are likely to be relevant for the classification. If it is suspected that there are many such variables then a smaller value of β is indicated. However, since the estimated relevance of any variable has a minimum value of zero, relevance estimates have an upward bias. Sampling fluctuations are more likely to cause the relevance to be overestimated, and this is preferentially the case for (truly) low (near zero) relevance variables. Small values of β ($\beta < 1$) will tend to exaggerate this effect, thereby diluting the influence of the (truly) higher relevance variables. This is especially the case when there are a large number of such low relevance variables. Larger values of β provide resistance to this effect by giving proportionally more weight to the variables of higher estimated relevance. Also, the scythe will provide the greatest improvements over K -NN when there are fewer locally relevant variables and hence when higher values of β are most appropriate. One can always include simple K -NN (16) (17) (26) as an alternative to the scythe and compare the respective performances by cross-validation (leave-one-out technique). Thus, higher values of β should at least be tried. Finally, larger values of β are more appropriate when derived variables (Section 4.5) are included, since the presumption (hope) is that the derived variables are likely to be considerably more relevant than the individual inputs. All of these (latter) considerations suggest larger values for β .

For very large training samples ($N \rightarrow \infty$) and relatively small values of K (19) one might expect the value $\beta = 1$ to be appropriate. In this case the classification region $R(\mathbf{z})$

becomes arbitrarily small. If the target functions are continuous at \mathbf{z} then Taylor’s theorem suggests that they will be approximately linear over $R(\mathbf{z})$. If, in addition, the probability density distribution $p(\mathbf{x})$ of the inputs is continuous at \mathbf{z} then it should approach a constant density within $R(\mathbf{z})$. As noted above $\beta = 1$ is optimal for this case. However, as discussed in Section 3.1, sample sizes are seldom large enough for asymptotic results to apply. Never-the-less, $\beta = 1$ is a good choice if the dominant contribution to the bias of the target function estimates is the linear component caused by the fact that the prediction point \mathbf{z} is seldom located very close to the sample mean $\bar{\mathbf{x}}$ within the prediction region $R(\mathbf{z})$. This is especially likely in high dimensional problems with many input variables.

The choice of the $q = \infty$ norm in (75) (76) is not fundamental and other choices (such as Euclidean, $q = 2$) could be used to define the next region $R_{k+1}(\mathbf{z})$. An advantage of the $q = \infty$ norm (in any K -NN procedure) is that a group of highly correlated variables will have (nearly) the same collective influence on the distance measure as any single one of them in that group. For example, one cannot increase the influence of a single variable x_i by simply replicating it several times in the list of input variables. This is not the case, for example, with the $q = 2$ (Euclidean) norm. Since highly correlated variables will tend to have similar relevance (35) (42), the use of the $q = \infty$ norm insures that they do not receive undue influence, thereby weakening the relative contribution of other variables (uncorrelated with them) that contain genuinely independent information.

5 Examples

The scythe represents a family of K -nearest-neighbor procedures containing ordinary K -NN and the machete as extreme special cases. Particular members of the family are specified by choices for the values of several parameters that control the operation of the procedure. In addition the procedure can be augmented by the inclusion of derived variables (Section 4.5). In order to obtain an indication of the effects of these choices on performance we present results of a comparative study for several of them. In particular, we consider the scythe with three values of β (78): $\beta = 0$ (regular Euclidean “ K -NN”), $\beta = 1$ (“scythe”), and $\beta = \infty$ (“machete”). With the machete we further consider four versions: (1) using only the p original input variables, (2) adding the Euclidean distance (69) derived variable, (3) adding a linear combination derived variable (73)

$$h_{j(\mathbf{z})}(\mathbf{x}), \quad j(\mathbf{z}) = \arg \max_{1 \leq j \leq J} h_j(\mathbf{z}), \quad (79)$$

and (4) including both of these derived variables.

In all applications of the scythe/machete the splitting parameter α (64) was set to $\alpha = 0.5$, and the conditional expectation parameter L (48) was set to $L = 20$. With the regular K -NN procedure, the results with the metric (26) based on interquartile range (25) is presented, since it consistently outperformed using the alternative (23) and especially (24). In addition to these six procedures, two recursive partitioning methods are included in the comparisons: CART using only the p original input variables, and CART with linear combination splits included [see Breiman, et. al. (1984)].

5.1 Simulated data examples

Since one might expect the relative performance of the procedures under comparison to depend on the details of the particular problem to which they are applied we compare them in a variety of situations. Such situations are characterized by the number of input variables p , training sample size N , and especially the detailed distributions of the respective classes in the input measurement space. It is always possible to cause the performance

of ordinary K -NN to degrade arbitrarily by adding enough globally (completely) irrelevant input variables. Since employing variable subset selection with K -NN can largely remedy this effect, only situations in which every input variable has at least some global relevance are considered here. Thus substantial gains with (global) variable subset selection would not be expected. For all procedures being compared, the parameter that controls the size of the classification region (K for K -NN/scythe/machete and cost-complexity parameter with CART) were estimated by cross-validation.

In all simulated situations ten independent training samples (of size N) were generated. For each of these, an additional independent test sample consisting of 2000 observations was generated. These test observations were classified by each competing method using the respective training data set. Error rates computed over all 20000 such classifications are reported in the comparisons shown in Table 2

5.1.1 Example 1

This first example was constructed to be especially favorable to the adaptive methods (scythe/machete/CART) and unfavorable to the regular K -NN procedure. There are $p = 10$ input variables, $N = 200$ training observations, and $J = 2$ classes. The data for the first class was generated from a standard normal distribution $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$. Those for the second class were also generated from normal distribution $\mathbf{x}_n \sim N(\mathbf{m}, \mathbf{C})$ with the coordinate mean values and covariance matrix given by

$$\left\{ m_i = \sqrt{i}/2 \right\}_1^p, \quad \mathbf{C} = \text{diag} \left\{ 1/\sqrt{i} \right\}_1^p \quad (80)$$

Thus, although all input variables are relevant, the ones with the higher coordinate number i are clearly more so. Also because of the selected covariance matrix, much of the discriminating information is axis oriented.

The first row of Table 2 show the results for the eight methods under comparison. The scythe ($\beta = 1$) had the lowest error rate with the machete exhibiting similar performance, especially with the derived variables included. As would be expected the K -NN procedure performs relatively poorly in this situation. Somewhat of a surprise is the even worse performance of both versions of the recursive partitioning (CART) procedure.

5.1.2 Example 2

This example is similar to the first one, but designed to be more favorable to the K -NN procedure. There are $p = 10$ input variables, $N = 200$ training observations, and $J = 2$ classes. As above, data for the first class is generated from a standard normal. Data for the other class are also normal with the same covariance matrix as above (80), but this time the mean vector is given by

$$\left\{ m_i = \sqrt{p - i + 1}/2 \right\}_{i=1}^p \quad (81)$$

Although the variances (80) are most different between the two classes on the inputs with high coordinate number (as above), their means (81) are most separated in the lower coordinates, so that all variables contain substantial discriminating information.

The second row of Table 2 show the results for this example. The performance of K -NN is seen to be better here, but so is that of all the other procedures, with CART again exhibiting the worst performance. The scythe and the machete are again comparable, except for the latter when the distance (derived) variable (69) is included. This is seen to reduce the error rate by roughly 70% over the basic machete, and produce one third the error rate of K -NN, and one fourth that of basic CART.

5.1.3 Example 3

This example (and the next three) are designed to be more favorable to the recursive partitioning methodology. The class probability functions (10) are constructed to be piecewise-constant with no overlap in the class densities $\{p(\mathbf{x} | j)\}_1^J$ (8). Thus the (true) probability functions (10) assume only two values (0/1) and the minimum (Bayes) error rate (3) is zero for such problems. In this example all of the training data are generated according to a standard normal distribution $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$. The $J = 2$ classes are defined by

$$\sum_{i=1}^{10} x_i^2/i \leq 2.5 \Rightarrow \text{class 1, otherwise} \Rightarrow \text{class 2.} \quad (82)$$

As above there are $p = 10$ input variables and $N = 200$ training observations. Here again the higher numbered variables are the more relevant ones.

The results for this example are shown on the third row of Table 3. As expected CART shows nearly the best performance, being only slightly edged out by the machete with the distance derived variable (69). The scythe also shows close to the best performance. Other versions of the machete do not do quite as well, and K -NN exhibits especially poor performance in this situation due to the uneven relevance of the input variables.

5.1.4 Example 4

This example is similar to the previous one, but is designed to be more favorable to the K -NN procedure, in that all input variable have exactly the same global relevance. Here again there are $p = 10$ inputs, $J=2$ classes, but $N = 500$ training observations. The data is generated from a standard normal $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{1})$ and the classes are defined by

$$\sum_{i=1}^{10} x_i^2 \leq 9.8 \Rightarrow \text{class 1, otherwise} \Rightarrow \text{class 2.} \quad (83)$$

Results are shown on the fourth row of Table 2. The K -NN procedure is seen to be more competitive here. It has the same performance as in the previous example but that of the other procedures is degraded because their adaptive strategies are less effective here. The best performer is again the machete with the distance derived variable (69). The scythe shows the worst performance here.

5.1.5 Example 5

This example is constructed so that all input variables have equal local relevance everywhere in the input measurement space. However, there is a single direction in that space that contains all the discriminating information. There are $p = 10$ inputs, $N = 200$ training observations. The $J = 2$ classes are defined by

$$\sum_{i=1}^{10} x_i \leq 0 \Rightarrow \text{class 1, otherwise} \Rightarrow \text{class 2.} \quad (84)$$

The fifth row of Table 2 shows the results. As would be expected the linear combination derived variable (79) is highly effective here with the machete, as is its counterpart in CART. In the absence of this derived variable K -NN performs best because all variables are equally locally relevant everywhere; however the scythe comes close to it in performance. CART without linear combinations is the worst performer in this situation.

5.1.6 AC-circuit example

This example is taken from Friedman (1991). It consists of a simple electric circuit with an alternating current voltage generator (frequency ω), a resistor R , inductor L , and capacitor C , all connected in series. The shift in phase of the current with respect to that for the generated voltage is given by

$$\phi = \tan^{-1}[(\omega L - 1/\omega C)/R]. \quad (85)$$

The problem is to classify the phase shift as positive or negative given a specific set of values for (ω, R, L, C) . Thus there are $J = 2$ classes. The $p = 4$ input variables are generated from a uniform distribution in the respective intervals $20 \leq \omega/2\pi \leq 280$ hertz, $0 \leq R \leq 100$ ohms, $0 \leq L \leq 1$ henry, and $1 \leq C \leq 11$ microfarads. The training sample size was taken to be 200.

The sixth row of Table 2 shows the results for this example. The best performers here were the two machete procedures involving the linear combination derived variable (79), followed by CART with linear combinations. The worst performer in this situation was CART without linear combinations.

5.1.7 Waveform example

This simulated example was introduced in Breiman et. al. (1984) and has since become a commonly used benchmark for classification techniques. There are $p = 21$ input variables, $N = 300$ training observations, and $J = 3$ classes. An observation \mathbf{x}_n from each respective class is a random convex combination of two out of three basic waveforms $\{h_k(i)\}_{k=1}^3$ sampled at the integers $i = 1, 21$, with noise added

$$\begin{aligned} \text{class 1: } x_{ni} &= u_n h_1(i) + (1 - u_n) h_2(i) + \varepsilon_{ni} \\ \text{class 2: } x_{ni} &= u_n h_1(i) + (1 - u_n) h_3(i) + \varepsilon_{ni} \\ \text{class 3: } x_{ni} &= u_n h_2(i) + (1 - u_n) h_3(i) + \varepsilon_{ni} \end{aligned} \quad (86)$$

where $1 \leq n \leq 300$ and $1 \leq i \leq 21$. The basic waveforms are

$$h_1(i) = (6 - |i - 7|)_+, \quad h_2(i) = (6 - |i - 15|)_+, \quad h_3(i) = (6 - |i - 11|)_+. \quad (87)$$

The u_n are generated from a uniform distribution $u_n \sim U(0, 1)$, and the ε_{ni} from a standard normal distribution $\varepsilon_{ni} \sim N(0, 1)$.

The seventh row of Table 2 show the results on this waveform example. The K -NN procedure is seen to do the best with the scythe and machete (involving linear combinations) close behind. CART without linear combinations is seen to do much worse than all of the others. Using linear combinations with CART improves its performance almost to the point of the machete without linear combinations.

In problems where the input variable index i is itself an orderable quantity, such as waveforms or spectra, it is common practice to attempt to use this fact to advantage, especially when noise contamination is suspected. One way to do this is to replace each observed spectrum (observation) $\mathbf{x} = \{x_i\}_1^p$ by a smoothed version $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_1^p$ as the input variables to the classification procedure. The eighth row of Table 2 shows the corresponding results employing a very simple smoother

$$\tilde{x}_i = (x_{i-1} + 2x_i + x_{i+1})/4, \quad i = 2, 20, \quad (88)$$

and

$$\tilde{x}_1 = (2x_1 + x_2)/3, \quad \tilde{x}_p = (2x_p + x_{p-1})/3.$$

The performance of the K -NN procedure is seen to change very little as a result of the smoothing (88), whereas all of the adaptive procedures show dramatic improvement. In fact the scythe, and the machete with both derived variables (69) (79), achieve error rates very close to the minimum possible (Bayes) error rate of 0.14 for this problem [Breiman, et. al. (1984)].

These changes in performance as a result of the smoothing (88) are understandable. The basic waveforms (87) were chosen so that in the absence of the noise ε_{ni} , the three classes (86) are highly separable using only a small subset of the input variables. This is very favorable for the adaptive procedures, since they can identify and use only these highly relevant variables and not be confused by the others. In the presence of noise however this advantage is diluted by the fact that using more input variables produces an averaging effect that dampens the noise. The K -NN procedure uses all of the input variables with equal weight thereby producing the most noise suppression. This causes it to out perform the adaptive procedures despite its lack of flexibility. The smoothing preprocess (88) directly damps the noise so that involving fewer (smoothed) variables in the classification procedure itself is less of a disadvantage from the point of view of noise suppression. This permits the adaptive procedures to more actively take advantage of the (intrinsic) uneven relevance among the (smoothed) inputs to improve performance, whereas the (additional) smoothing provided by the K -NN procedure is of little increased value in this case.

Table 2

Ex.	CART	CART:ln	K -NN	scythe	mach.	mach:ln	mach:ds	mach:btb
1	15.1	14.8	13.2	6.3	7.4	6.4	6.9	6.5
2	14.6	11.7	10.3	5.2	5.7	5.2	3.4	3.5
3	21.9	21.8	35.9	22.8	28.6	24.5	21.7	24.5
4	32.2	32.3	34.0	37.9	28.0	29.1	26.2	31.7
5	32.4	7.6	17.4	19.0	26.5	6.2	25.4	6.2
AC	13.5	7.8	10.8	9.0	9.7	6.4	9.8	6.5
wave	29.1	21.1	17.1	18.7	23.2	19.3	22.3	19.1
wav-sm.	17.5	16.9	16.1	14.8	15.4	15.3	15.2	14.4

5.1.8 Discussion

The collection of simulated data examples was chosen so that each of the eight methods should do well in at least one of them. This is verified in Table 2. For each method there is at least one example where it had the best, or close to the best, performance. Since it is always possible to construct situations that favor a particular method over all of the others, the issue becomes one of robustness. That is, how well a particular method m performs on average in situations that are most favorable to other procedures. A quantity that captures this is the ratio r_m of its error rate e_m to the smallest error rate over all the methods being compared in a particular example

$$r_m = e_m / \min_{1 \leq k \leq 8} e_k. \quad (89)$$

The best method m^* for that example will have $r_{m^*} = 1$ and all other methods will have larger values $\{r_m \geq 1\}_{m \neq m^*}$. The larger the value for this ratio (89) the worse the performance of the m th method is in relation to the best one for that situation among the methods being compared. The distribution of r_m , for each method m , over all the examples provides information concerning its robustness. For example, if a particular method

had an error rate close to the best in every example its distribution of r_m values would cluster near the value 1.0. Departures from this distribution reflect lack of robustness.

Figure 1 shows box plots of the distribution of r_m (89) for each method over the eight simulated data examples. The shaded area of each box represents the lower and upper quartiles of the distribution and the horizontal line between them is the median. The extent of the outer vertical lines represents the entire range of values for the distribution. One sees that the most robust method over the eight simulated examples is the machete procedure involving both derived variables (69) (79). In half of the examples its error rate was no worse than 3% higher than the best (median = 1.03). In 3/4 of them it was no more than 12% higher, and in the worst case it was 20% higher than the best error rate. At the other extreme is basic CART (without linear combinations) where the corresponding numbers are 90%, 235%, and 423%.

All of the scythe/machete procedures are seen to outperform both recursive partitioning procedures and the K -NN method in terms of robustness. A possible exception is the basic machete (without derived variables) which has comparable performance to CART with linear combinations. The scythe ($\beta = 1$) performs better than the basic machete ($\beta = \infty$) and comparable to the machete with the distance derived variable (69). The two machete procedures that include the linear combination derived variable (79) show the most robust performance over these examples. It is important to remember however that these judgements are based on collective performance over all eight situations. There was at least one situation for each method where it had the best performance, or close to it.

It could be argued that the particular choice of examples was heavily weighted in favor of the recursive partitioning approach. Four of the eight examples (3, 4, 5, and 6) were constructed so that the true probability functions (10) are piecewise constant taking on only two values $\{f_j(\mathbf{x}) = 0/1\}_1^J$. Moreover, along the (decision) boundary between these two values the density of data points is relatively high. These situations are most favorable for the recursive partitioning approach and Table 2 shows that it was the most competitive in these examples. However, it seems unlikely that this type of situation would occur often in practice. Either the classes are well separated, so that even though the probability functions are only two-valued piecewise-constant, the data density near the boundaries is small, or the class distributions overlap so that the probability functions vary in a smooth manner as one passes through the overlap region. The reason for including these examples was to illustrate that even though they do favor recursive partitioning, the nearest-neighbor based procedures (K -NN, scythe, machete) often remain competitive. On the other hand, in cases where there is class overlap and the probability functions (10) vary continuously across decision boundaries (examples 1, 2, and waveform) the recursive partitioning methods are far less competitive.

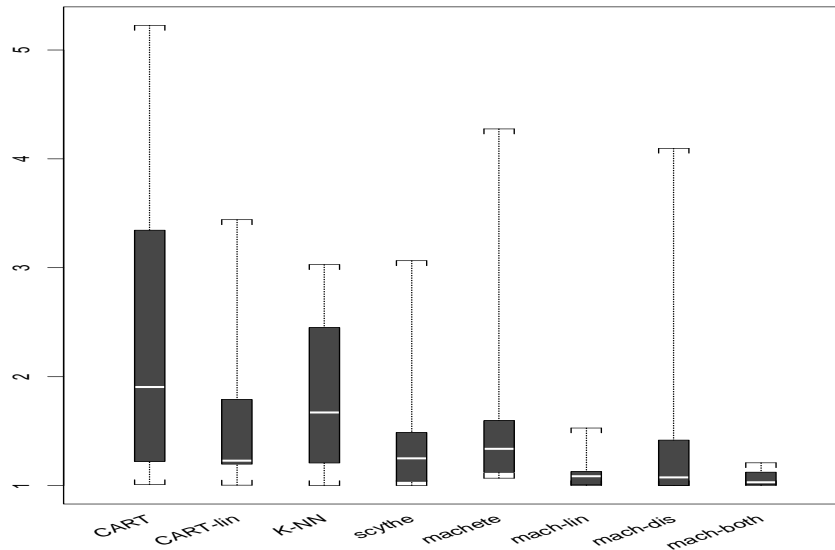


Figure 1

5.2 Real data examples

For several reasons comparative studies on real data tend to be less informative than those based on simulated data. First, the true underlying structure in terms of the individual class density distributions is unknown for a real data set. When performance differences between methods are observed one cannot determine the underlying causes and make appropriate generalizations. Second, it is more difficult to reliably estimate such performance differences. A particular data set represents only one (random) sample drawn from the class probability distributions that characterize the system producing the data. This introduces a statistical uncertainty (variance) in the observed performance differences. Different training samples (of the same size) produced from the same system will generally give rise to different observed relative performances among the methods. In a simulation study one can average performances over a large number of randomly drawn training samples thereby reducing this source of uncertainty at will.

There is an additional statistical uncertainty associated with observed performance differences even within a single training data set (of size N). Using leave-one-out cross-validation for the estimates, one has only N observations with which to estimate the respective error rates. For many data sets this is too small a number to achieve high statistical accuracy. In simulation studies one can generate a very large “test” data sets to get precise estimates of the error rates for each training data set. For these reasons, even substantial performance differences between methods as estimated using real data sets are seldom statistically significant. One can never be sure whether any such observed differences are due to chance artifacts of the particular data set, or are actually representative of the underlying system that produced the data.

The main advantage of real data is that it is produced without any knowledge of the particular classification procedures that it will be used to test. Therefore, it cannot be customized to favor any particular procedure. Also, there is always the suspicion that artificially constructed examples will not correspond to situations that are likely to occur in practice. For these reasons comparisons on real data can be useful, provided the

cautionary notes raised above are kept in mind.

Table 3 shows the (cross-validated) error rates for the eight methods under consideration on eight real data sets. These data sets represent those in Andrews and Herzberg (1985) relating to classification problems. A very brief description of each one is provided below along with the chapter in Andrews and Herzberg (“A&H”) where more detail is provided.

5.2.1 Iris data (A&H, Ch. 1)

This data set consists of $p = 4$ measurements made on each of $N = 100$ iris plants of $J = 2$ species. The measurements are the sepal length and width, and the petal length and width. The two species are iris versicolor and iris virginica. The problem is to classify each to its correct species based on the four measurements. Data for a third species (iris setosa) is also presented in A&H, Ch. 1, but it was excluded from this analysis since it is trivially separable from the other two classes. The results on this data set are presented in the first row of Table 3.

5.2.2 Product preference data (A&H, Ch. 31)

The first data set consists of 200 judges who scored the taste of two products on six quality attributes. The $p = 7$ predictors were taken to be the differences of the two scores on each attribute and a binary valued variable indicating which product was tasted first. The $J = 2$ classes are defined as the particular sample (first or second) the judge preferred. Judges expressing no preference were deleted leaving a sample of $N = 186$. Results are shown on the second row of Table 3.

A second taste preference data set consists of 28 judges rating two samples on each of $p = 11$ taste attributes. Each of the two samples presented to the judges was drawn from one of three products. Each judge was presented with all six possible pairings for a total of 168 observations. As above the $p = 11$ input variables were taken to be the score differences between the two samples, for each of the 11 taste attributes. The $J = 2$ classes were taken to be which sample (first or second) the judge preferred. Judgements expressing no preference were deleted resulting in $N = 165$ observations. Results are shown on the third row of Table 3.

5.2.3 Diabetes data (A&H, Ch. 36)

This data set represents $N = 145$ people divided into $J = 3$ classes: (1) those suffering from chemical subclinical diabetes, (2) those with overt nonketotic diabetes, and (3) normal nondiabetics. The $p = 5$ measurements were the person’s relative weight and four clinical measurements. Results are shown on the fourth row of Table 3.

5.2.4 Muscular Dystrophy Carriers (A&H, Ch. 38)

The purpose of collecting these data was to develop a method of screening female relatives of boys with Duchenne Muscular Dystrophy for being possible carriers of the disease. There were $N = 190$ women comprising the data set. The $J = 2$ classes are carrier/noncarrier. The $p = 5$ input variables are the persons age and measurements of the blood levels of four serum enzymes. Results are shown on the fifth row of Table 3.

5.2.5 Urine crystals (A&H, Ch. 44)

Measurements of $p = 6$ physical characteristics of 79 urine specimens were analyzed to determine if they relate to the formation of calcium oxalate crystals. The $N = 77$

observations with no missing measurements were used. The $J = 2$ classes relate to the presence/absence of crystals. Results are shown on the sixth row of Table 3.

5.2.6 Kangaroo skull measurements (A&H, Ch. 53)

These data represent $p = 18$ skull measurements made on 148 kangaroos from $J = 3$ species. The $N = 101$ observations with no missing data were used. The seventh row of Table 3 shows results for species classification. The eighth row shows the results of classifying the sex of the kangaroos based on these same skull measurements.

Table 3

Data	CART	CART:ln	K -NN	scythe	mach.	mach:ln	mach:ds	mach-bth
iris	11.0	6.0	8.0	3.0	6.0	5.0	6.0	5.0
tas1	12.2	12.2	16.1	13.4	11.8	15.6	11.8	16.7
tas2	12.5	14.4	12.7	11.5	9.7	9.7	10.9	10.9
diab.	4.3	3.6	11.7	7.6	9.6	8.3	11.0	7.6
dmd	15.8	13.2	10.5	10.0	10.0	10.0	8.9	8.4
urine	22.9	24.7	20.8	22.1	22.1	20.8	22.1	20.8
kng1	48.0	33.0	24.8	28.7	38.7	19.8	39.6	19.8
kng2	32.0	35.0	27.7	29.7	30.7	26.7	31.7	26.7

5.2.7 Discussion

Examination of Table 3 indicates qualitatively similar behavior of the performance differences over the eight real data sets as that observed for the simulated ones (Table 2). For each method there was at least one data set for which its error rate was the best, or close to it. Figure 2 examines the robustness issue with box plots for each of the eight columns of Table 3. Again one sees similar behavior as that reflected for the simulated examples in Figure 1. The median performance of all scythe/machete procedures was better than that for the recursive partitioning (CART) and K -NN procedures. In fact the respective medians have almost the same ordering over the eight procedures for the real data sets as they did for the simulated examples. The principal exception is the relationship between the two CART procedures. CART with linear combinations had a higher median than basic CART over the real data sets although the performance for the latter was considerably less consistent (larger interquartile range). Over all eight procedures the most consistent performer in this sense was the scythe ($\beta = 1$). Although its median performance was not as good as the two machete procedures with linear combinations included, its range of performance values was considerably narrower.

In general, the spread of performance ratios (interquartile range) is seen to be broader for the real data sets. This may be due to the fact that the individual performance estimates for the simulated data examples are considerably more accurate so that their range of values directly reflects the variability of their performance ratios over the various situations. For the real data examples there is considerable (within situation) variability caused by the lack of statistical accuracy with which the individual error rates can be measured, as discussed in Section 5.2. For example, the performance difference estimates on each of the individual data sets was seldom statistically significant.

The simulated data situations were constructed so that all input variables had some relevance to the classification. This is not necessarily the case for the real data examples. Therefore it is possible that the performance of the K -NN procedure might be enhanced through the use of input variable subset selection techniques (Section 3.2). It has been argued however [Langley (1994)] that real data tends to contain few totally irrelevant

input variables since an informal selection procedure is performed by the experimenter collecting the data. One is not likely to measure and record values of quantities that are suspected of being completely unrelated to the problem. Since all the real data used here were (originally) manually collected, that could be the case. On the other hand, data automatically collected by computers may well have many variables that are unrelated to the problem for which the data is ultimately used.

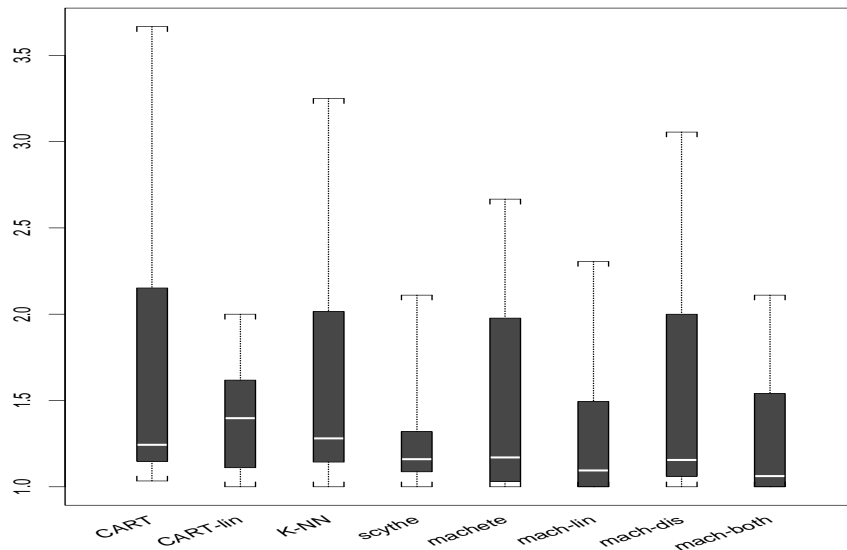


Figure 2

6 Conclusion

The results on both the real and simulated data examples indicate that the scythe/machete procedures proposed here have the potential to improve the performance of K -NN and recursive partitioning procedures in some classification problems. These problems are characterized by unequal influence of the measurement input variables on the class assignment, especially when their relative influence changes with the location of the object to be classified in the input measurement space. It is important to keep in mind however that these procedures have limitations (as do all procedures) that can limit their effectiveness in other problems for which they are not well suited.

First, like recursive partitioning methods the scythe/machete applies a “greedy” strategy. The (local) relevance of an input or derived variable is gauged by its individual influence when acting alone. It can sometimes be the case that a particular variable may have little relevance by itself but be highly relevant when used in concert with other variables. The expectation (hope) is that such variables leave enough of a “footprint” in terms of their individual relevance so that they will eventually be selected for splitting. The result of the split will then expose their increased relevance as well as that of the other variables that interact with them in this manner. Although this is often (usually) the case, it need not always be so. As with recursive partitioning procedures, (global) variable subset selection [John, Kohavi, and Pflieger (1994)] can be applied with the scythe/machete procedures to help mitigate this problem when it is suspected.

The scythe/machete is designed to enhance the performance of K -NN and recursive partitioning procedures. Therefore it is likely to be most effective in situations where at least one of these procedures already tends to be a relatively good performer. Judging from benchmark studies there are many such situations. There are settings however for which this is not the case. For example, settings where the target probability functions (10) tend to be (globally) additive (40), or involve at most low order interactions, will not be conducive to either of these procedures. This is because the local classification regions that they produce have a convex shape. Functions that mainly involve low order interactions are best approximated by procedures that (implicitly) produce equivalent kernels with concave shape. For such situations, linear discriminant analysis or “flexible” discriminant analysis (FDA) [Hastie and Tibshirani (1995)] based on MARS [Friedman (1991)] might be a better alternative.

The computation involved in implementing the scythe/machete, though intense, is not excessive. Preprocessing (“training”) involves simply sorting the training data on each input variable. If linear combination derived variables (Section 4.5) are included then they can be computed during this preprocessing stage for use in the first iteration of the splitting procedure for all points to be classified. After preprocessing, the dominant part of the computation for each classification is proportional to $p \cdot n_s \cdot L$ where p is the number of inputs, n_s is the number of splits (66), and L is the number of counts (48) for the conditional expectation estimates. There is an additional (smaller) computation associated with making the splits which totals $p \cdot N$ for performing all of them. If linear combination derived variables are employed there is additional computation involved (roughly) proportional to $p^2 N \log_{1/\alpha} N$ (64). This could be excessive for some problems with a very large number of input variables.

The computation required by the scythe/machete for classifying a point \mathbf{z} depends on the difficulty of its class assignment. If at any step k in the splitting procedure the current region $R_k(\mathbf{z})$ is pure (all training observations of the same class) then partitioning stops and the assignment is made. In addition, if when computing the conditional expectation estimates for the i th variable x_i , the corresponding interval Δ_i (47) is pure, then one can stop partitioning and make the class assignment provided there are no other variables for which this is also true with a different (pure) class. In such situations classification with the scythe/machete sometimes can be much faster than ordinary K -NN classification.

The scythe/machete can be directly adapted to the “regression” setting where the objective is to predict a real valued orderable output variable. The realized gain over K -NN and recursive partitioning would likely be greater here than for classification since performance is measured directly in terms of function prediction rather than misclassification risk, which does not require accurate function estimates in order to be small. However, the performance of K -NN and recursive partitioning methods is seldom competitive with other procedures expressly designed for regression (function approximation) problems. Thus, the gains associated with the scythe/machete would have to be quite large to make it a contender in this area. This is a topic of future research.

7 References

- Andrews, D. F. and Herzberg, A. M. (1985). *Data. A Collection of Problems for Many Fields for the Student and Research Worker*. Springer-Verlag.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.
- Chow, W. S. and Chen, Y. C. (1992). A new fast algorithm for effective training of neural classifiers. *Pattern Recognition* **25**, 423-429.
- Cover, T. M. (1968). Rates of convergence for nearest neighbor procedures. *Proc. Hawaii Inter. Conf. on Systems Sciences* (pp. 413-415). Honolulu: Western Periodicals.
- Fix, E. and Hodges, J. L. (1951). Discriminatory analysis - nonparametric discrimination: consistency properties. *Report No. 4*. Randolph Field Texas: U. S. Air Force School of Aviation Medicine.
- Friedman J. H. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Computers* **C26**, 404-408.
- Friedman J. H. (1985). Classification and multiple response regression through projection pursuit. Technical Report LCS012, Dept. of Statistics, Stanford Univ.
- Friedman J. H. (1991). Multivariate adaptive regression splines (with discussion). *Ann. Statist.* **19**, 1-141.
- Hastie, T. and Tibshirani, R. (1995). Flexible discriminant analysis. *J. Amer. Statist. Assoc.* (to appear).
- John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proc. Eleventh Inter. Conf. on Machine Learning* (pp. 121-129). New Brunswick, NJ. Morgan Kaufmann.
- Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE* **78**, 1464-1480.
- Langley, P. (1994). Selection of relevant features in machine learning. *Proc. AAAI Fall Symp. on Relevance (1994)*. New Orleans, LA. AAAI Press.
- Lippmann, R. (1989). Pattern classification using neural networks. *IEEE Communications Magazine* **11**, 47-64.
- McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley.
- Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *J. Amer. Statist. Assoc.* **58**, 415-435.
- Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Stat.* **33**, 1065-1076.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.