Nonparametric Classification
10716, Spring 2020
Pradeep Ravikumar (amending notes from Larry Wasserman)

1 Introduction

Let $h : \mathcal{X} \to \{0, 1\}$ to denote a classifier where $\mathcal{X}$ is the domain of X. In parametric classification we assumed that h took a very constrained form, typically linear. In nonparametric classification we aim to relax this assumption.

Let us recall a few definitions and facts. The *classification risk, or error rate*, of h is

$$R(h) = \mathbb{P}(Y \neq h(X)) \tag{1}$$

and the *empirical error rate* or *training error rate* based on training data $(X_1, Y_1), \ldots, (X_n, Y_n)$ is

$$\widehat{R}_n(h) = \frac{1}{n} \sum_{i=1}^{n} I(h(X_i) \neq Y_i). \tag{2}$$

$R(h)$ is minimized by the *Bayes' rule*

$$h^*(x) = \begin{cases} 1 & \text{if } m(x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } \frac{p_1(x)}{p_0(x)} > \frac{(1-\pi)}{\pi} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where $m(x) = \mathbb{P}(Y = 1 \,|\, X = x)$, $p_j(x) = p(x \,|\, Y = j)$ and $\pi = \mathbb{P}(Y = 1)$. The *excess risk* of a classifier h is $R(h) - R(h^*)$.

In the multiclass case, $Y \in \{1, \ldots, k\}$, the Bayes' rule is

$$h^*(x) = \text{argmax}_{1 \leq j \leq k} \pi_j \; p_j(x) = \text{argmax}_{1 \leq j \leq k} m_j(x)$$

where $m_j(x) = \mathbb{P}(Y = j | X = x)$, $\pi_j = \mathbb{P}(Y = j)$ and $p_j(x) = p(x|Y = j)$. **Proof.** We have

$$\begin{aligned} R(h) &= 1 - \mathbb{P}\left(h(X) = Y\right) && (4) \\ &= 1 - \sum_{k=0}^{K-1} \mathbb{P}\left(h(X) = k, Y = k\right) && (5) \\ &= 1 - \sum_{k=0}^{K-1} \mathbb{E}\Big[I\big(h(X) = k\big) \mathbb{P}\left(Y = k | X\right) \Big] && (6) \end{aligned}$$

It's clear that $h^*(X) = \text{argmax}_k \, \mathbb{P}\left(Y = k | X\right)$ achieves the minimized classification error $1 - \mathbb{E}\big[\max_k \mathbb{P}\left(Y = k | X\right)\big]$. $\square$

2 Classification Error for Parametric Models

We will consider the convergence rates for the ERM classifier $\widehat{h}$:

$$\widehat{h} \in \arg\min_{h \in \mathcal{H}} \widehat{R}_n(\beta) := \frac{1}{n} \sum_{i=1}^{n} I(Y_i \neq h(X_i)).$$

Suppose $\mathcal{H}$ has finite VC dimension d_{vc}. For instance, if $\mathcal{H} = \{\text{sign}(\beta^T \cdot) : \beta \in \mathbb{R}^d\}$ is the set of linear classifiers, then its VC dimension $d_{\text{vc}} = d + 1$. Due to standard uniform law based analysis, we obtain that with probability at least $1 - \delta$:

$$R(\widehat{h}) - R(h_*) \asymp \sqrt{\frac{d_{\text{vc}} \log n}{n}} + \sqrt{\frac{\log(1/\delta)}{n}}.$$

The result can be improved if the distribution is well-behaved, and there are not too many data points near the decision boundary. We'll state a result due to Koltchinski and Panchenko (2002) that involves *the margin.*(See also Kakade, Sridharan and Tewari 2009). Let us take $Y_i \in \{-1, +1\}$ so we can write $h(x) = \text{sign}(\beta^T x)$. Suppose that $|X(j)| \leq A < \infty$ for each j. We also restrict ourselved to the set of linear classifiers $h(x) = \text{sign}(\beta^T x)$ with $|\beta(j)| \leq A$. Define the *margin-sensitive loss*

$$\phi_\gamma(u) = \begin{cases} 1 & \text{if } u \leq 0 \\ 1 - \frac{u}{\gamma} & \text{if } 0 < u \leq \gamma \\ 0 & \text{if } u > \gamma. \end{cases}$$

Then, for any such classifier h, with probability at least $1 - \delta$,

$$P(Y \neq h(X)) \leq \frac{1}{n} \sum_{i=1}^{n} \phi_\gamma(Y_i h(X_i)) + \frac{4A^{3/2} d}{\gamma n} + \left(\frac{8}{\gamma} + 1\right) \sqrt{\frac{\log(4/\delta)}{2n}}.$$

This means that, if there are few observations near the boundary, then, by taking γ large, we can make the loss small. However, the restriction to bounded covariates and bounded classifiers is non-trivial.

In summary, classification error for simpler classes of parameteric models scales with the dimensionality of the features, perhaps additionally scaled by the margin. Let us see how the classification error behaves with more complex i.e. non-parametric hypothesis classes.

3 Classifiers based on Non-parametric Regression

One approach to nonparametric classification is to estimate the key unknown quantity: $m(x) = \mathbb{P}(Y = 1|X) = \mathbb{E}(Y|X)$ in the Bayes' rule (3) and simply plug the estimate in.

Thus using any nonparametric regression estimator $\widehat{m}$, we obtain the corresponding classifier:

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{m}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

For example, we could use the kernel regresson estimator

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n Y_i \, K\left(\frac{||x-X_i||}{h}\right)}{\sum_{i=1}^n K\left(\frac{||x-X_i||}{h}\right)}.$$

Howeve, the bandwidth should be optimized for classification error as described in Section 11.

We have the following theorem.

Theorem 1 *Let $\widehat{h}$ be the plug-in classifier based on $\widehat{m}$. Then,*

$$R(\widehat{h}) - R(h^*) \leq 2 \int |\widehat{m}(x) - m(x)| dP(x) \leq 2 \sqrt{\int |\widehat{m}(x) - m(x)|^2 dP(x)}. \tag{8}$$

Proof. Now,

$$\begin{aligned} &\mathbb{P}\big(Y \neq h(X) | X = x\big) = 1 - \mathbb{P}\big(Y = h(X) | X = x\big) \\ &= 1 - \Big(\mathbb{P}\big(Y = 1, h(X) = 1 | X = x\big) + \mathbb{P}\big(Y = 0, h(X) = 0 | X = x\big)\Big) \\ &= 1 - \Big(h(x)\mathbb{P}(Y = 1 | X = x) + \big(1 - h(x)\big)\mathbb{P}(Y = 0 | X = x)\Big) \\ &= 1 - \Big(h(x)m(x) + \big(1 - h(x)\big)\big(1 - m(x)\big)\Big). \end{aligned}$$

Hence,

$$\begin{aligned} &\mathbb{P}\big(Y \neq h(X) | X = x\big) - \mathbb{P}\big(Y \neq h^*(X) | X = x\big) \\ &= \Big(h^*(x)m(x) + \big(1 - h^*(x)\big)\big(1 - m(x)\big)\Big) - \Big(h(x)m(x) + \big(1 - h(x)\big)\big(1 - m(x)\big)\Big) \\ &= \big(2m(x) - 1\big)\big(h^*(x) - h(x)\big) = 2\left(m(x) - \frac{1}{2}\right)\big(h^*(x) - h(x)\big) \\ &= \big|2\widehat{m}(x) - 1\big| I\big(h^*(x) \neq \widehat{h}(x)\big) = 2\big|\widehat{m}(x) - 1/2\big| I\big(h^*(x) \neq \widehat{h}(x)\big). \end{aligned}$$

Now, when $h^*(x) \neq \widehat{h}(x)$, there are two possible cases: (i) $\widehat{h}(x) = 1$ and $h^*(x) = 0$; (ii) $\widehat{h}(x) = 0$ and $h^*(x) = 1$. In both cases, we have that $\big|\widehat{m}(x) - m^*(x)\big| \geq \big|\widehat{m}(x) - 1/2\big|$.

Therefore,

$$\begin{aligned}
\mathbb{P}\big(\widehat{h}(X) \neq Y\big) - \mathbb{P}\big(h^*(X) \neq Y\big) &= 2\int \big|\widehat{m}(x) - 1/2\big| I\big(h^*(x) \neq \widehat{h}(x)\big) dP_X(x) \\
&\leq 2\int \big|\widehat{m}(x) - m^*(x)\big| I\big(h^*(x) \neq \widehat{h}(x)\big) dP_X(x) \\
&\leq 2\int \big|\widehat{m}(x) - m^*(x)\big| dP_X(x) \qquad (9) \\
&\leq 2\sqrt{\int \big(\widehat{m}(x) - m^*(x)\big)^2 dP_X(x)}. \qquad (10)
\end{aligned}$$

The last inequality follows from the fact that $\mathbb{E}|Z| \leq \sqrt{\mathbb{E}Z^2}$ for any Z. $\square$

An immediate consequence of this theorem is that any result about nonparametric regression can be turned into a result about nonparametric classification. For example, if $\int |\widehat{m}(x) - m(x)|^2 dP(x) = O_P(n^{-2\beta/(2\beta+d)})$ then $R(\widehat{h}) - R(h^*) = O_P(n^{-\beta/(2\beta+d)})$. However, (8) is an upper bound and it is possible that $R(\widehat{h}) - R(h^*)$ is strictly smaller than $\sqrt{\int |\widehat{m}(x) - m(x)|^2 dP(x)}$.

When $Y \in \{1, \ldots, k\}$ the plugin rule has the form

$$\widehat{h}(x) = \text{argmax}_j \widehat{m}_j(x)$$

where $\widehat{m}_j(x)$ is an estimate of $\mathbb{P}(Y = j|X = x)$.

4 Is Classification easier than Regression; Minimax Results

The previous result showed that classification is easier than regression, in that if $R(\widehat{m}) \to 0$, then $R(\widehat{h}) - R(h_*) \to 0$. But can it be strictly easier?

To study this formally, consider the minimax classification risk over a set of joint distributions $\mathcal{P}$ is

$$R_n(\mathcal{P}) = \inf_{\widehat{h}} \sup_{P \in \mathcal{P}} \Big(\mathbb{E}R(\widehat{h}) - R_n^*\Big) \qquad (11)$$

where $R(\widehat{h}) = \mathbb{P}(Y \neq \widehat{h}(X))$, R_n^* is the Bayes error and the infimum is over all classifiers constructed from the data $(X_1, Y_1), \ldots, (X_n, Y_n)$. Recall our previous result that

$$R(\widehat{h}) - R(h^*) \leq 2\sqrt{\int |\widehat{m}(x) - m(x)|^2 dP(x)}$$

Class	Rate	Condition
$\mathcal{E}(\alpha)$	$n^{-\alpha/(2\alpha+d)}$	$\alpha > 1/2$
BV	$n^{-1/3}$	
MI	$\sqrt{\log n/n}$	
$L(\alpha, q)$	$n^{-\alpha/(2\alpha+1)}$	$\alpha > (1/q - 1/2)_+$
$B^{\alpha}_{\sigma,q}$	$n^{-\alpha/(2\alpha+d)}$	$\alpha/d > 1/q - 1/2$
Neural nets	$\left(\frac{\log n}{n}\right)^{\frac{1+(1/d)}{4+(2/d)}}$	

Table 1: Minimax Rates of Convergence for Classification.

Thus $R_n(\mathcal{P}) \leq 2\sqrt{\widetilde{R}_n(\mathcal{P})}$ where $\widetilde{R}_n(\mathcal{P})$ is the minimax risk for estimating the regression function m. Since this is just an inequality, it leaves open the following question: can the classification error $R_n(\mathcal{P})$ be substantially smaller than $2\sqrt{\widetilde{R}_n(\mathcal{P})}$?

4.1 $\mathcal{P}$ is substantially rich

In cases where $\mathcal{P}$ is substantially rich, Yang (1999) proved that the answer is no, and that for some rate r_n depending on the class,

$$\inf_{\widehat{m}} \sup_{m \in \mathcal{M}} R(\widehat{h}) \asymp r_n^2 \quad \text{and} \quad \inf_{\widehat{h}} \sup_{m \in \mathcal{M}} [R(\widehat{h}) - R(h_*)] \asymp r_n$$

which says that the minimax classification rate is the square root of the regression rate. As a byproduct, he showed that we can achieve minimax classification rates using the plugin regression method above.

We provide a brief elaboration of Yang's results under the richness assumption. This assumption is simply that if m is in the class, then a small hypercube containing m is also in the class. Yang's results are summarized in Table 1. The classes in Table 1 are the following: $\mathcal{E}(\alpha)$ is th Sobolev space of order α, BV is the class of functions of bounded variation, MI is all monotone functions, $L(\alpha, q)$ are α-Lipschitz (in q-norm), and $B^{\alpha}_{\sigma,q}$ are Besov spaces.

For neural nets, it appears that, as $d \to \infty$, we get the dimension independent rate $(\log n/n)^{1/4}$. However, this result requires some caution since due to the richness assumptions in the paper, the class of distributions implicitly gets smaller as d increases.

4.2 $\mathcal{P}$ is smaller

To motivate why classification could be much simpler than regression, note that it is possible for $\widehat{m}$ to be far from $m^*(x)$ and still lead to a good classifier. As long as $\widehat{m}(x)$ and $m^*(x)$

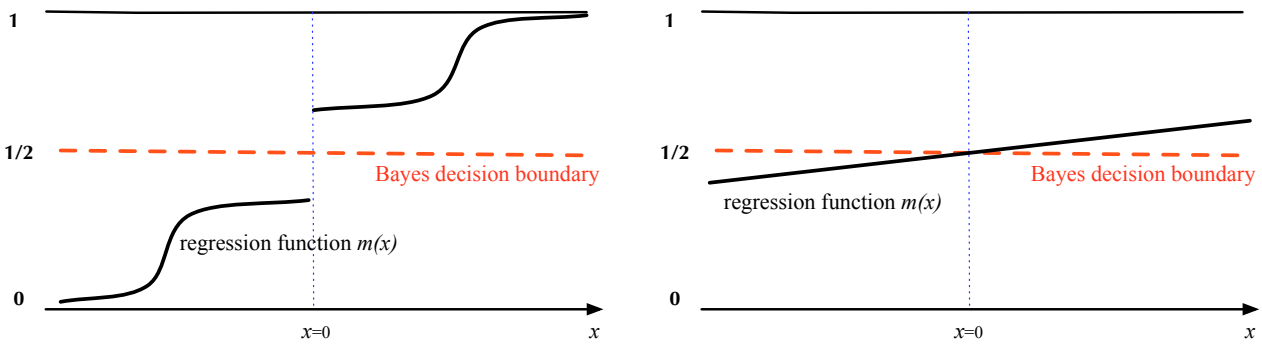

Figure 1: The Bayes rule is $h^*(x) = I(x > 0)$ in both plots, which show the regression function $m(x) = \mathbb{E}(Y|x)$ for two problems. The left plot shows an easy problem; there is little ambiguity around the decision boundary. The right plot shows a hard problem; it is hard to know from the data if you are to the left or right of the decision boundary.

are on the same side of 1/2 they yield the same classifier.

Example 2 *Figure 1 shows two one-dimensional regression functions. In both cases, the Bayes rule is* $h^*(x) = I(x > 0)$ *and the decision boundary is* $\mathcal{D} = \{x = 0\}$. *The left plot illustrates an easy problem; there is little ambiguity around the decision boundary. Even a poor estimate of* $m(x)$ *will recover the correct decision boundary. The right plot illustrates a hard problem; it is hard to know from the data if you are to the left or right of the decision boundary.*

In general, with smaller classes that invoke extra assumptions, such as the *Tsybakov low-noise condition*, the classification error can be dramatically lower. We briefly review classification error results for small (finite dimensional) classes.

5 Classifiers Based on Density Estimation

We can apply nonparametric density estimation for each class to estimate $p_0 = \mathbb{P}(X|Y = 0)$ and $p_1 = \mathbb{P}(X|Y = 1)$ via estimators $\widehat{p}_0$ and $\widehat{p}_1$. We then define

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \frac{\widehat{p}_1(x)}{\widehat{p}_0(x)} > \frac{(1-\widehat{\pi})}{\widehat{\pi}} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $\widehat{\pi} = n^{-1}\sum_{i=1}^n Y_i$. Hence, any nonparametric density estimation method yields a nonparametric classifier.

A simplification occurs if we assume that the covariate has independent coordinates, conditioned on the class variable Y. Thus, if $X_i = (X_{i1}, \ldots, X_{id})^T$ has dimension d and if we assume conditional independence, then the density factors as $p_j(x) = \prod_{\ell=1}^d p_{j\ell}(x_\ell)$. In this case we can estimate the one-dimensional marginals $p_{j\ell}(x_\ell)$ separately and then define $\widehat{p}_j(x) = \prod_{\ell=1}^d \widehat{p}_{j\ell}(x_\ell)$. This has the advantage that we never have to do more than a one-dimensional density estimate. This approach is called *naive Bayes*. The resulting classifier can sometimes be very accurate even if the independence assumption is false.

It is easy to extend density based methods for multiclass problems. If $Y \in \{1, \ldots, k\}$ then we estimate the k densities $\widehat{p}_j(x) = p(x|Y = j)$ and the classifier is

$$\widehat{h}(x) = \text{argmax}_j \ \widehat{\pi}_j \ \widehat{p}_j(x)$$

where $\widehat{\pi}_j = n^{-1} \sum_{i=1}^n I(Y_i = j)$.

6 Nearest Neighbors

The *k-nearest neighbor classifier* is

$$h(x) = \begin{cases} 1 & \sum_{i=1}^n w_i(x) I(Y_i = 1) > \sum_{i=1}^n w_i(x) I(Y_i = 0) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $w_i(x) = 1$ if X_i is one of the k nearest neighbors of x, $w_i(x) = 0$, otherwise. "Nearest" depends on how you define the distance. Often we use Euclidean distance $\|X_i - X_j\|$. In that case you should standardize the variables first.

The k-nearest neighbor classifier can be recast as a plugin rule. Define the regression estsimator

$$\widehat{m}(x) = \frac{\sum_{i=1}^n Y_i \ I(||X_i - x|| \leq d_k(x))}{\sum_{i=1}^n I(||X_i - x|| \leq d_k(x))}$$

where $d_k(x)$ is the distance between x and its k^{th}-nearest neighbor. Then $\widehat{h}(x) = I(\widehat{m}(x) > 1/2)$.

It is interesting to consider the classification error when n is large. First suppose that $k = 1$ and consider a fixed x. Then $\widehat{h}(x)$ is 1 if the closest X_i has label $Y = 1$ and $\widehat{h}(x)$ is 0 if the closest X_i has label $Y = 0$. As n gets large, the closest X_i in turn approaches x, and the probability of an error in turn

$$m(X_i)(1-m(x))+(1-m(X_i))m(x) \to m(x)(1-m(x))+(1-m(x))m(x) = 2m(x)(1-m(x)).$$

Define

$$L_n = \mathbb{P}(Y \neq \widehat{h}(X) \,|\, D_n)$$

where $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. Then we have that

$$\lim_{n\to\infty} \mathbb{E}(L_n) = \mathbb{E}(2m(X)(1 - m(X))) \equiv R_{(1)}. \tag{14}$$

The Bayes risk can be written as $R_* = \mathbb{E}(A)$ where $A = \min\{m(X), 1 - m(X)\}$. Note that $A \leq 2m(X)(1 - m(X))$. Note also that $m(x)(1 - m(x)) = A(x)[1 - A(x)]$. Next we claim that $\mathbb{E}[A(X)(1 - A(X))] \leq \mathbb{E}[A(X)]\mathbb{E}[1 - A(X)]$. This follows since

$$\int a(x)(1-a(x))dP(x) - \int a(y)dP(y)(1-a(x))dP(x) = \int\int (a(x)-a(y))(1-a(x))dP(x)dP(y)$$

$$= \int_x \int_{x>y} (a(x)-a(y))(1-a(x))dP(x)dP(y) + \int_x \int_{x<y} (a(x)-a(y))(1-a(x))dP(x)dP(y)$$

$$= \int_x \int_{x>y} (a(x)-a(y))(1-a(x))dP(x)dP(y) + \int_y \int_{x>y} (a(y)-a(x))(1-a(y))dP(x)dP(y)$$

$$= \int_y \int_{x>y} (a(x)-a(y))(a(y)-a(x))dP(x)dP(y) = -\int_y \int_{x>y} (a(x)-a(y))^2 dP(x)dP(y) \leq 0.$$

So

$$R_* \leq R_{(1)} = 2\mathbb{E}[m(X)(1-m(X))] = 2\mathbb{E}[A(X)(1-A(X))] \leq \mathbb{E}[A(X)]\mathbb{E}[1-A(X)] = 2R_*(1-R_*) \leq 2R_*.$$

This result is due to Cover and Hart (1967). Thus, for any problem with small Bayes error, $k = 1$ nearest neighbors should have small error. This is a remarkable result. It says that, in some cases, we can do very well without any regularization!

More generally, for any odd k, (Devroye et al 1996) showed the following result:

Theorem 3 (Devroye et al 1996) *For all odd k,*

$$R_* \leq R_{(k)} \leq R_* + \frac{1}{\sqrt{ke}}. \tag{15}$$

While the previous result was asymptotic, showing that nearest neighbors has good performance as the number of samples approaches infinity, we can also analyze its non asymptotic convergence rates.

If the distribution of X has a density function then we have the following.

Theorem 4 (Devroye and Györfi 1985) *Suppose that the distribution of X has a density and that $k \to \infty$ and $k/n \to 0$. For every $\epsilon > 0$ the following is true. For all large n, with probability at least $1 - \delta$,*

$$R(\widehat{h}) - R_* \leq \frac{6\sqrt{2}\gamma_d\sqrt{\log(1/\delta)}}{\sqrt{n}},$$

where $\widehat{h}_n$ is the k-nearest neighbor classifier estimated on a sample of size n, and where γ_d depends on the dimension d of X.

Recently, Chaudhuri and Dasgupta (2014) have obtained some very general results about k-nn classifiers. We state one of their key results here.

Theorem 5 (Chaudhuri and Dasgupta 2014) *Suppose that the distribution satisfies the following* ***margin condition****:*

$$P\left(\{x:\ |m(x)-(1/2)|\leq t\}\right)\leq Ct^{\beta}$$

for some $\beta\geq 0$ and some $C>0$. Also, suppose that m satisfies the following ***smoothness condition****: for all x and $r>0$*

$$|m(B)-m(x)|\leq LP(B^o)^{\alpha}$$

where $B=\{u:\ ||x-u||\leq r\}$, $B^0=\{u:\ ||x-u||<r\}$ and $m(B)=(P(B))^{-1}\int_B m(u)dP(x)$. Fix any $0<\delta<1$. Let h_* be the Bayes rule. If $k\asymp n^{\frac{2\alpha}{2\alpha+1}}$ then

$$\mathbb{E}R(\widehat{h})-R(h_*)\preceq n^{-\frac{\alpha(\beta+1)}{2\alpha+1}}.$$

These results thus show that nearest neighbor classification has good convergence rates, provided that we have a suitable metric over our feature space. This is of course easier said than done, but modern feature representation learning methods, such as via deep neural networks, do provide us with embeddings and corresponding metrics that might well satisfy these regularity conditions. Nearest neighbor approaches have thus had a recent resurgence in popularity, particularly in ultra-low-sampling regimes (e.g. "k shot learning").

6.1 Partitions and Trees

As with nonparametric regression, simple and interpretable classifiers can be derived by partitioning the range of X. Let $\Pi_n=\{A_1,\ldots,A_N\}$ be a partition of $\mathcal{X}$. Let A_j be the partition element that contains x. Then $\widehat{h}(x)=1$ if $\sum_{X_i\in A_j}Y_i\geq\sum_{X_i\in A_j}(1-Y_i)$ and $\widehat{h}(x)=0$ otherwise. This is nothing other than the plugin classifier based on the partition regression estimator

$$\widehat{m}(x)=\sum_{j=1}^{N}\overline{Y}_j\,I(x\in A_j)$$

where $\overline{Y}_j=n_j^{-1}\sum_{i=1}^{n}Y_iI(X_i\in A_j)$ is the average of the Y_i's in A_j and $n_j=\#\{X_i\in A_j\}$. (We define $\overline{Y}_j$ to be 0 if $n_j=0$.)

Recall from the results on regression that if

$$m\in\mathcal{M}=\left\{m:\ |m(x)-m(x)|\leq L||x-z||,\quad x,z,\in\mathbb{R}^d\right\} \tag{16}$$

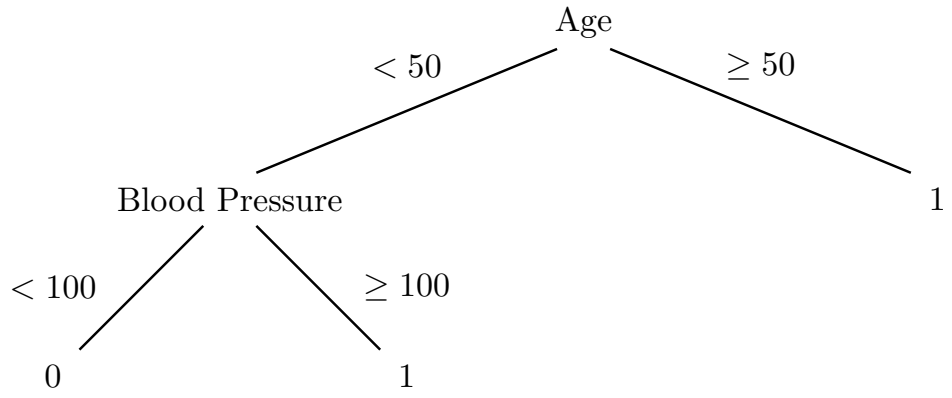

Figure 2: A simple classification tree.

and the binwidth b satsifies $b \asymp n^{-1/(d+2)}$ then

$$\mathbb{E}||\widehat{m} - m||_P^2 \leq \frac{c}{n^{2/(d+2)}}. \tag{17}$$

From (8), we conclude that $\mathbb{E}R(\widehat{h}) - R(h_*) = O(n^{-1/(d+2)})$. However, this binwidth was based on the bias-variance tradeoff of the regression problem. For classification, b should be chosen as described in Section 11.

Like regression trees, *classification trees* are partition classifiers where the partition is built recursively. For illustration, suppose there are two covariates, $X_1 =$ age and $X_2 =$ blood pressure. Figure 2 shows a classification tree using these variables.

The tree is used in the following way. If a subject has Age ≥ 50 then we classify him as $Y = 1$. If a subject has Age < 50 then we check his blood pressure. If systolic blood pressure is < 100 then we classify him as $Y = 1$, otherwise we classify him as $Y = 0$. Figure 3 shows the same classifier as a partition of the covariate space.

Here is how a tree is constructed. First, suppose that $y \in \mathcal{Y} = \{0, 1\}$ and that there is only a single covariate X. We choose a split point t that divides the real line into two sets $A_1 = (-\infty, t]$ and $A_2 = (t, \infty)$. Let $r_s(j)$ be the proportion of observations in A_s such that $Y_i = j$:

$$r_s(j) = \frac{\sum_{i=1}^n I(Y_i = j, X_i \in A_s)}{\sum_{i=1}^n I(X_i \in A_s)}, \tag{18}$$

and w_s be the proportion of observations in A_s:

$$w_s = \sum_{i=1}^n I(X_i \in A_s),$$

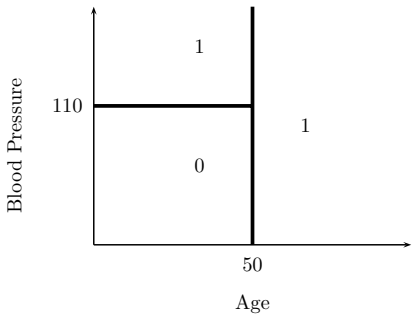

Figure 3: Partition representation of classification tree.

for $s = 1, 2$ and $j = 0, 1$. The *impurity* of the split t is defined to be $I(t) = \sum_{s=1}^{2} w_s \gamma_s$ where

$$\gamma_s = 1 - \sum_{j=0}^{1} r_s(j)^2. \tag{19}$$

This particular measure of impurity is known as the *Gini index*. If a partition element A_s contains all 0's or all 1's, then $\gamma_s = 0$. Otherwise, $\gamma_s > 0$. We choose the split point t to minimize the impurity. Other indices of impurity besides the Gini index can be used, such as entropy. The reason for using impurity rather than classification error is because impurity is a smooth function and hence is easy to minimize.

When there are several covariates, we choose whichever covariate and split that leads to the lowest impurity. This process is continued until some stopping criterion is met. For example, we might stop when every partition element has fewer than n_0 data points, where n_0 is some fixed number. The bottom nodes of the tree are called the *leaves*. Each leaf is assigned a 0 or 1 depending on whether there are more data points with $Y = 0$ or $Y = 1$ in that partition element.

This procedure is easily generalized to the case where $Y \in \{1, \ldots, K\}$. We define the impurity by

$$\gamma_s = 1 - \sum_{j=1}^{k} r_s^2(j) \tag{20}$$

where $r_i(j)$ is the proportion of observations in the partition element for which $Y = j$.

7 RKHS Classification

When we discussed RKHS regression, we also briefly discussed how we could use it for losses other than squared error, such as Hinge losses, as well as logistic log-likelihood losses.

Alternatively, these could be viewed as applying the kernel trick to linear approaches to classification.

Logistic Regression. Let

$$m(x) = \mathbb{P}(Y = 1|X = x) = \frac{e^{f(x)}}{1 + e^{f(x)}}.$$

We can estimate m by minimizing

$$-\text{loglikelihood} + \lambda||f||_K^2.$$

Then $\widehat{f} = \sum_j K(x_j, x)$ and α may be found by numerical optimization; see the chapter. In this case, smoothing kernels are much easier.

Support Vector Machines. Suppose $Y_i \in \{-1, +1\}$. Recall the the linear SVM minimizes the penalized hinge loss:

$$J = \sum_i [1 - Y_i(\beta_0 + \beta^T X_i)]_+ + \frac{\lambda}{2}||\beta||_2^2.$$

The RKHS version is to minimize

$$J = \sum_i [1 - Y_i f(X_i)]_+ + \frac{\lambda}{2}||f||_K^2.$$

The dual is the same except that $\langle X_i, X_j \rangle$ is replaced with $K(X_i, X_j)$.

8 Boosting

Recall the discussion of linear/additive non-parametric approaches to regression. As discussed there, boosting is essentially a non-parametric additive approach for classification.

9 Sparse Nonparametric Logistic Regression

For high dimensional problems we can use sparsity-based methods. The nonparametric additive logistic model is

$$\mathbb{P}(Y = 1 \,|\, X) \equiv p(X; f) = \frac{\exp\left(\sum_{j=1}^p f_j(X_j)\right)}{1 + \exp\left(\sum_{j=1}^p f_j(X_j)\right)} \tag{21}$$

where $Y \in \{0, 1\}$, and the population log-likelihood is

$$\ell(f) = \mathbb{E}\left[Y f(X) - \log\left(1 + \exp f(X)\right)\right] \tag{22}$$

where $f(X) = \sum_{j=1^p} f_j(X_j)$.

A sparsity penality can be incorporated, just as for sparse additive models (SpAM) for regression. The Lagrangian is given by

$$\mathcal{L}(f, \lambda) = \mathbb{E}\left[\log\left(1 + e^{f(X)}\right) - Y f(X)\right] + \lambda \left(\sum_{j=1}^{p} \sqrt{\mathbb{E}(f_j^2(X_j))} - L\right) \tag{23}$$

To fit this model, one can first compute a quadratic approximation of the loss (or equivalently, linearize the gradient), and then run the backfitting procedure. Let

$$Z_i = \widehat{f}(X_i) + \frac{Y_i - p(X_i; \widehat{f})}{p(X_i; \widehat{f})(1 - p(X_i; \widehat{f}))} \tag{24}$$

and weights $w(X_i) = p(X_i; \widehat{f})(1 - p(X_i; \widehat{f})$.

We would then be carrying out a weighted backfitting of (Z, X) with weights w.

To see this, note that the linearized stationary condition is given by: $\mathbb{E}\left[w(X)(f(X) - Z) \,|\, X_j\right] + \lambda v_j = 0$, where v_j is an element of the subgradient $\partial\sqrt{\mathbb{E}(f_j^2)}$.

When $\mathbb{E}(f_j^2) \neq 0$, this yields the backfitting step:

$$f_j(X_j) = \frac{\mathbb{E}(wR_j \,|\, X_j)}{\left(\mathbb{E}\left(w \,|\, X_j\right) + \frac{\lambda}{\sqrt{\mathbb{E}(f_j^2)}}\right)}. \tag{25}$$

In the finite sample case, in terms of the smoothing matrix $\mathcal{S}_j$, this becomes

$$f_j = \frac{\mathcal{S}_j(wR_j)}{\mathcal{S}_j w + \lambda \Big/ \sqrt{\mathbb{E}(f_j^2)}}. \tag{26}$$

If $\|\mathcal{S}_j(wR_j)\| < \lambda$, then $f_j = 0$. Otherwise, this implicit, nonlinear equation for f_j cannot be solved explicitly, so one simply iterates until convergence:

$$f_j \;\leftarrow\; \frac{\mathcal{S}_j(wR_j)}{\mathcal{S}_j w + \lambda\sqrt{n}\,/\|f_j\|}. \tag{27}$$

Example 6 (SpAM for Spam) *Here we consider an email spam classification problem, using the logistic SpAM backfitting algorithm above. This dataset has been studied Hastie et*

al (2001) using a set of 3,065 emails as a training set, and conducting hypothesis tests to choose significant variables; there are a total of 4,601 observations with $p = 57$ attributes, all numeric. The attributes measure the percentage of specific words or characters in the email, the average and maximum run lengths of upper case letters, and the total number of such letters.

$\lambda(\times 10^{-3})$	ERROR	# ZEROS	SELECTED VARIABLES
5.5	0.2009	55	{ 8,54}
5.0	0.1725	51	{ 8, 9, 27, 53, 54, 57}
4.5	0.1354	46	{7, 8, 9, 17, 18, 27, 53, 54, 57, 58}
4.0	0.1083 ($\surd$)	20	{4, 6–10, 14–22, 26, 27, 38, 53–58}
3.5	0.1117	0	ALL
3.0	0.1174	0	ALL
2.5	0.1251	0	ALL
2.0	0.1259	0	ALL

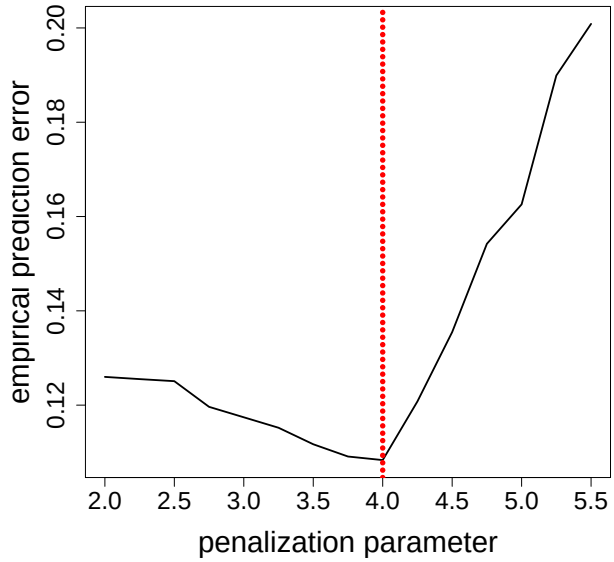

Figure 4: (Email spam) Classification accuracies and variable selection for logistic SpAM.

The results of a typical run of logistic SpAM are summarized in Figure 4, using plug-in bandwidths. A held-out set is used to tune the regularization parameter λ.

10 Bagging and Random Forests

Suppose we draw B bootstrap samples and each time we construct a classifier. This gives classifiers $h_1, \ldots, h_B$. We now classify by combining them:

$$h(x) = \begin{cases} 1 & \text{if } \frac{1}{B}\sum_j h_j(x) \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

This is called *bagging* which stands for *bootstrap aggregration*. The basline classifiers are usually trees.

A variation is to choose a random subset of the predictors to split on at each stage. The resulting classifier is called a random forests. Random forests often perform very well. Their theoretical performance is not well understood. Some good references are:

Biau, Devroye and Lugosi. (2008). Consistency of Random Forests and Other Average Classifiers. *JMLR*.

Biau, G. (2012). Analysis of a Random Forests Model. arXiv:1005.0208.

Lin and Jeon. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association*, 101, p 578.

Wager, S. (2014). Asymptotic Theory for Random Forests. arXiv:1405.0352.

Wager, S. (2015). Uniform convergence of random forests via adaptive concentration. arXiv:1503.06388.

11 Choosing Tuning Parameters

All the nonparametric methods involve tuning parameters, for example, the number of neighbors k in nearest neighbors. As with density estimation and regression, these parameters can be chosen by a variety of cross-validation methods. Here we describe the *data splitting* version of cross-validation. Suppose the data are $(X_1, Y_1), \ldots, (X_{2n}, Y_{2n})$. Now randomly split the data into two halves that we denote by

$$\mathcal{D} = \Big\{(\widetilde{X}_1, \widetilde{Y}_1), \ldots, (\widetilde{X}_n, \widetilde{Y}_n)\Big\}, \quad \text{and} \quad \mathcal{E} = \Big\{(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)\Big\}.$$

Construct classifiers $\mathcal{H} = \{h_1, \ldots, h_N\}$ from $\mathcal{D}$ corresponding to different values of the tuning parameter. Define the risk estimator

$$\widehat{R}(h_j) = \frac{1}{n}\sum_{i=1}^{n} I(Y_i^* \neq h_j(X_i^*)).$$

Let $\widehat{h} = \text{argmin}_{h\in\mathcal{H}}\widehat{R}(h)$.

Theorem 7 *Let $h_* \in \mathcal{H}$ minimize $R(h) = \mathbb{P}(Y \neq h(X))$. Then, with probability at least $1 - \delta$:*

$$R(\widehat{h}) - R(h_*) \leq 2\sqrt{\frac{1}{2n}\log\left(\frac{2N}{\delta}\right)}.$$

Proof. By Hoeffding's inequality, $\mathbb{P}(|\widehat{R}(h) - R(h)| > \epsilon) \leq 2e^{-2n\epsilon^2}$, for each $h \in \mathcal{H}$. By the union bound,

$$\mathbb{P}(\max_{h\in\mathcal{H}} |\widehat{R}(h) - R(h)| > \epsilon) \leq 2Ne^{-2n\epsilon^2} = \delta$$

where $\epsilon = \sqrt{\frac{1}{2n}\log\left(\frac{2N}{\delta}\right)}$. Hence, except on a set of probability at most δ,

$$R(\widehat{h}) \leq \widehat{R}(\widehat{h}) + \epsilon \leq \widehat{R}(\widehat{h}_*) + \epsilon \leq R(\widehat{h}_*) + 2\epsilon.$$

□

Note that the difference between $R(\widehat{h})$ and $R(h_*)$ is $O(\sqrt{\log N/n})$ but in regression it was $O(\log N/n)$ which is an interesting difference between the two settings. Under low noise conditions, the error can be improved.

A popular modification of data-splitting is *K-fold cross-validation*. The data are divided into K blocks; typicaly $K = 10$. One block is held out as test data to estimate risk. The process is then repeated K times, leaving out a different block each time, and the results are averaged over the K repetitions.

12 Example

The following data are from simulated images of gamma ray events for the Major Atmospheric Gamma-ray Imaging Cherenkov Telescope (MAGIC) in the Canary Islands. The data are from archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope. The telescope studies gamma ray bursts, active galactic nuclei and supernovae remnants. The goal is to predict if an event is real or is background (hadronic shower). There are 11 predictors that are numerical summaries of the images. We randomly selected 400 training points (200 positive and 200 negative) and 1000 test cases (500 positive and 500 negative). The results of various methods are in Table 2. See Figures 5, 6, 7, 8.

Method	Test Error
Logistic regression	0.23
SVM (Gaussian Kernel)	0.20
Kernel Regression	0.24
Additive Model	0.20
Reduced Additive Model	0.20
11-NN	0.25
Trees	0.20

Table 2: Various methods on the MAGIC data. The reduced additive model is based on using the three most significant variables from the additive model.

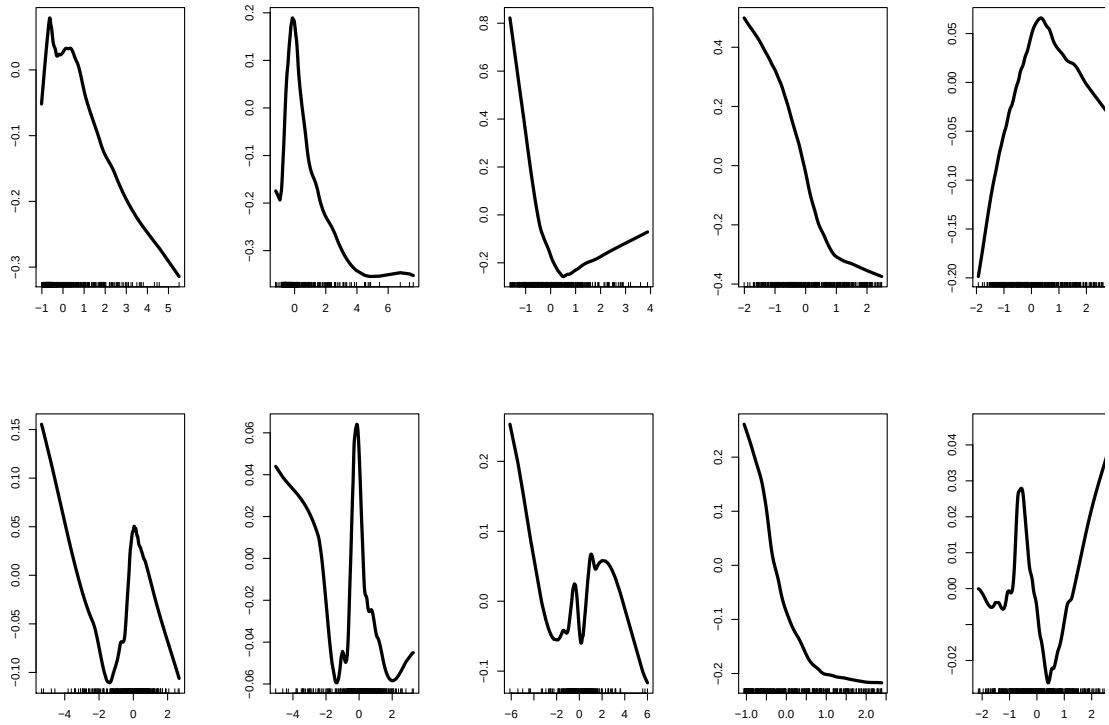

Figure 5: Estimated functions for additive model.

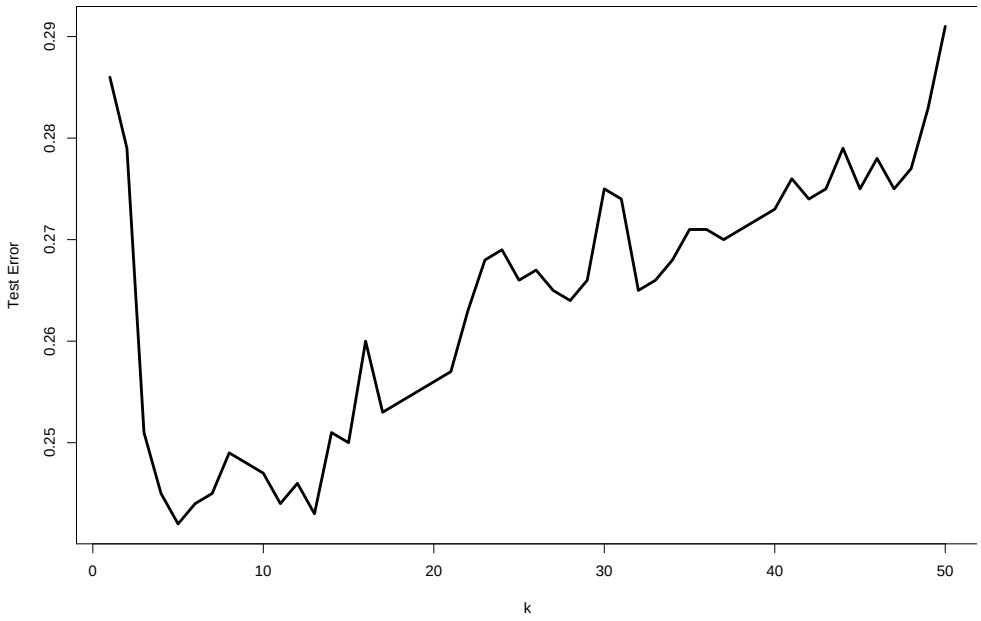

Figure 6: Test error versus k for nearest neighbor estimator.

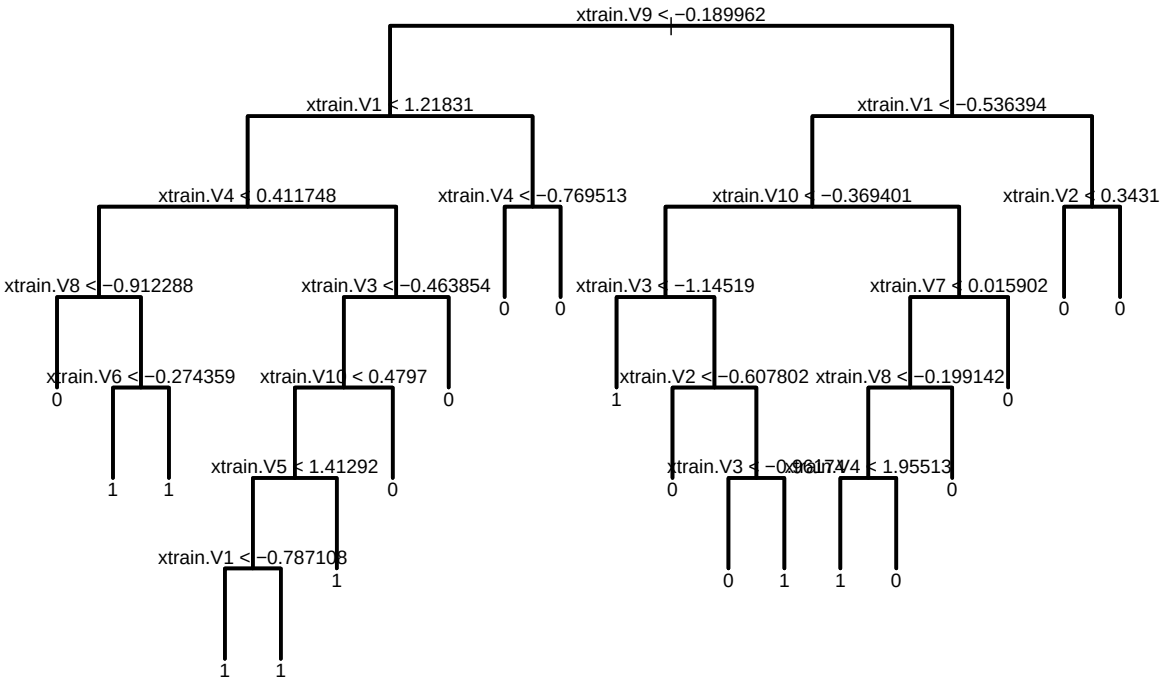

Figure 7: Full tree.

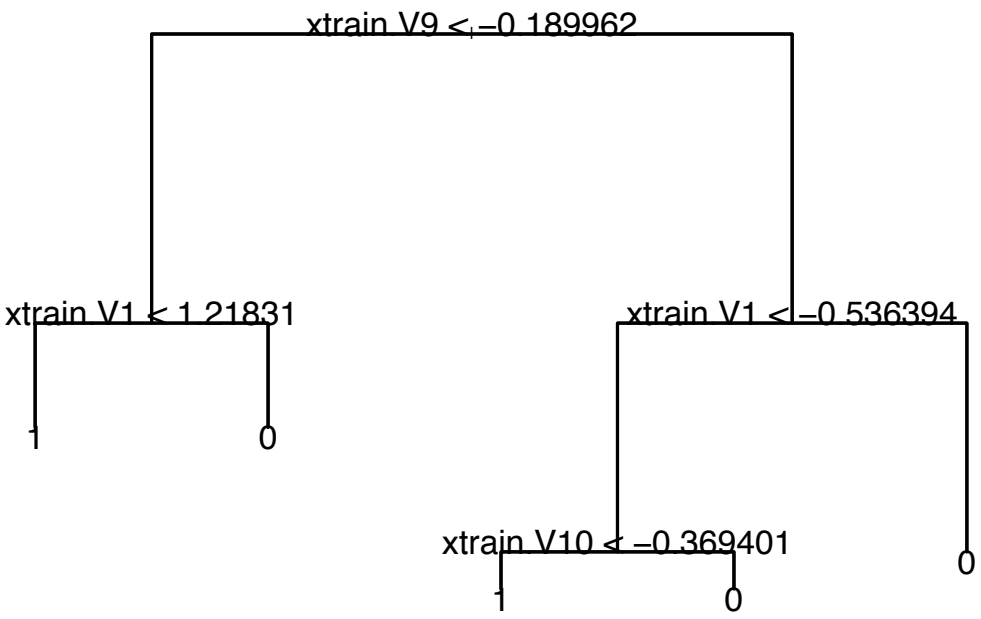

Figure 8: Classification tree. The size of the tree was chosen by cross-validation.