Nonparametric Classification
10/36-702

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 $p_j(x) = p(x|Y = j)$.

2 Plugin Methods

One approach to nonparametric classification is to estimate the unknown quantities in the expression for the Bayes' rule (3) and simply plug them in. For example, if $\widehat{m}$ is any nonparametric regression estimator then we can use

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{m}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

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 8.

We have the following theorem.

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

$$R(\widehat{h}) - R(h^*) \leq 2\int |\widehat{m}(x) - m(x)| dP(x) \leq 2\sqrt{\int |\widehat{m}(x) - m(x)|^2 dP(x)}. \tag{5}$$

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, (5) 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 $\mathbb{P}(Y = j|X = x)$.

3 Classifiers Based on Density Estimation

We can apply nonparametric density estimation to each class to get estimators $\widehat{p}_0$ and $\widehat{p}_1$. Then we define

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \frac{\widehat{p}_1(x)}{\widehat{p}_0(x)} > \frac{(1-\widehat{\pi})}{\widehat{\pi}} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

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

4 Nearest Neighbors

The *k-nearest neighbor classifier* is

$$h(x) = \begin{cases} 1 & \sum_{i=1}^n w_i(x) I(Y_i = 1) > \sum_{i=1}^n w_i(x) I(Y_i = 0) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

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

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

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 $Y = 0$. When n is large, the closest X_i is approximately equal to x. So the probability of an error is

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

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

$$\lim_{n\to\infty} \mathbb{E}(L_n) = \mathbb{E}(2m(X)(1 - m(X))) \equiv R_{(1)}. \tag{8}$$

The Bayes risk can be written as $R_* = \mathbb{E}(A)$ where $A = \min\{m(X), 1 - m(X)\}$. Note that $A \leq 2m(X)(1 - m(X))$. Also, by direct integration, $\mathbb{E}(A(1 - A)) \leq \mathbb{E}(A)\mathbb{E}(1 - A)$. Hence, we have the well-known result due to Cover and Hart (1967),

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

Thus, for any problem with small Bayes error, $k = 1$ nearest neighbors should have small error.

More generally, for any odd k,

$$\lim_{n\to\infty} \mathbb{E}(L_n) = R_{(k)} \tag{9}$$

where

$$R_{(k)} = \mathbb{E}\left(\sum_{j=0}^{k} \binom{k}{j} m^j(X)(1 - m(X))^{k-j} \left[m(X)I(j < k/2) + (1 - m(X))I(j > k/2)\right]\right).$$

Theorem 2 (Devroye et al 1996) *For all odd k,*

$$R_* \leq R_{(k)} \leq R_* + \frac{1}{\sqrt{ke}}. \tag{10}$$

Proof. We can rewrite $R_{(k)}$ as $R_{(k)} = \mathbb{E}(a(m(X)))$ where

$$a(z) = \min\{z, 1 - z\} + |2z - 1|\ \mathbb{P}\left(B > \frac{k}{2}\right)$$

and $B \sim \text{Binomial}(k, \min\{z, 1 - z\})$. The mean of $a(z)$ is less than or equal to its maximum and, by symmetry, we can take the maximum over $0 \leq z \leq 1/2$. Hence, letting $B \sim \text{Binomial}(k, z)$, we have, by Hoeffding's inequality,

$$R_{(k)} - R_* \ \leq \ \sup_{0\leq z\leq 1/2} (1 - 2z)\mathbb{P}\left(B > \frac{k}{2}\right) \leq \sup_{0\leq z\leq 1/2} (1 - 2z)e^{-2k(1/2-z)^2} = \sup_{0\leq u\leq 1} ue^{-ku^2/2} = \frac{1}{\sqrt{ke}}.$$

□

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

Theorem 3 (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,*

$$\mathbb{P}(R(\widehat{h}) - R_* > \epsilon) \leq e^{-n\epsilon^2/(72\gamma_d^2)}$$

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 4 (Chaudhuri and Dasgupta 2014) *Suppose that*

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

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. With probability at least $1-\delta$,

$$P(\widehat{h}(X)\leq h_*(X))\leq \delta C\left(\frac{\log(1/\delta)}{n}\right)^{\frac{\alpha\beta}{2\alpha+1}}.$$

If $k\asymp n^{\frac{2\alpha}{2\alpha+1}}$ then

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

4.1 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

$$m\in\mathcal{M}=\left\{m:\ |m(x)-m(x)|\leq L||x-z||,\quad x,z,\in\mathbb{R}^d\right\} \tag{11}$$

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

$$\mathbb{E}||\widehat{m}-m||_P^2\leq\frac{c}{n^{2/(d+2)}}. \tag{12}$$

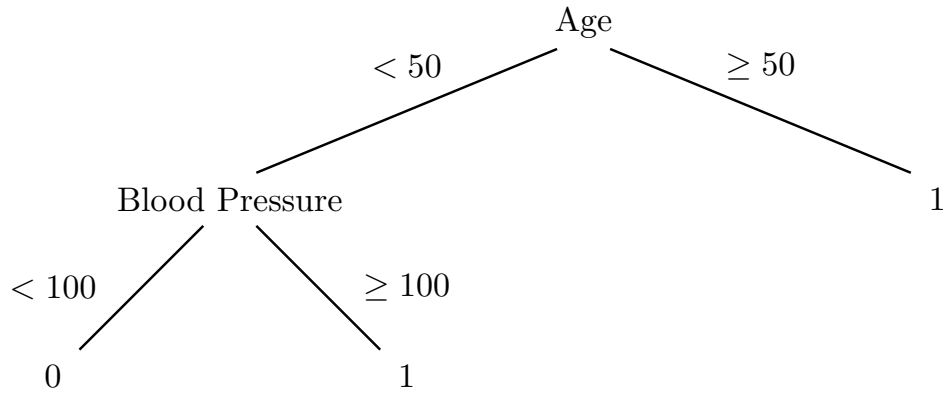

Figure 1: A simple classification tree.

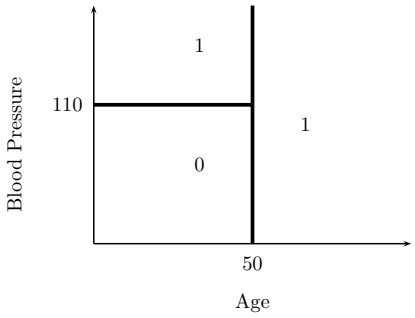

Figure 2: Partition representation of classification tree.

From (5), we conclude that $R(\widehat{h}) - R(h_*) = O(n^{-1/(d+2)})$. However, this binwidth was based on the bias-variance tradeoff of the regression problem. For classification, b should be chosen as described in Section 8.

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 1 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 2 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 $Y_i = j$:

$$r_s(j) = \frac{\sum_{i=1}^n I(Y_i = j, X_i \in A_s)}{\sum_{i=1}^n I(X_i \in A_s)} \tag{13}$$

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

$$\gamma_s = 1 - \sum_{j=0}^{1} r_s(j)^2. \tag{14}$$

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

$$\gamma_s = 1 - \sum_{j=1}^{k} r_s^2(j) \tag{15}$$

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

5 Minimax Results

The minimax classification risk over a set of joint distributions $\mathcal{P}$ is

$$R_n(\mathcal{P}) = \inf_{\widehat{h}} \sup_{P \in \mathcal{P}} \left(R(\widehat{h}) - R_n^* \right) \tag{16}$$

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 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	see text	

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 $R_n(\mathcal{P})$ be substantially smaller than $2\sqrt{\widetilde{R}_n(\mathcal{P})}$? Yang (1999) proved that the answer is no, in cases where $\mathcal{P}$ is substantially rich. Moreover, we can achieve minimax classification rates using plugin regression methods.

However, with smaller classes that invoke extra assumptions, such as the *Tsybakov noise condition*, there can be a dramatic difference. Here, we summarize 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 we have the bound, for every $\epsilon > 0$,

$$\left(\frac{1}{n}\right)^{\frac{1+(2/d)}{4+(4/d)}+\epsilon} \leq R_n(\mathcal{P}) \leq \left(\frac{\log n}{n}\right)^{\frac{1+(1/d)}{4+(2/d)}}$$

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 the class of distributions implicitly gets smaller as d increases.

6 Support Vector Machines

When we discussed linear classification, we defined SVM classifier $\widehat{h}(x) = \text{sign}(\widehat{H}(x))$ where $\widehat{H}(x) = \widehat{\beta}_0 + \widehat{\beta}^T x$ and $\widehat{\beta}$ minimizes

$$\sum_i [1 - Y_i H(X_i)]_+ + \lambda ||\beta||_2^2.$$

We can do a nonparametric version by letting H be in a RKHS and taking the penalty to be $||H||_K^2$. In terms of implementation, this means replacing every instance of an inner product $\langle X_i, X_j \rangle$ with $K(X_i, X_j)$.

7 Boosting

Boosting refers to a class of methods that build classifiers in a greedy, iterative way. The original boosting algorithm is called *AdaBoost* and is due to Freund and Schapire (1996). See Figure 3.

The algorithm seems mysterious and there is quite a bit of controversey about why (and when) it works. Perhaps the most compelling explanation is due to Friedman, Hastie and Tibshirani (2000) which is the explanation we will give. However, the reader is warned that there is not consensus on the issue. Futher discussions can be found in Bühlmann and Hothorn (2007), Zhang and Yu (2005) and Mease and Wyner (2008). The latter paper is followed by a spirited discussion from several authors. Our view is that boosting combines two distinct ideas: *surrogate loss functions* and *greedy function approximation*.

In this section, we assume that $Y_i \in \{-1, +1\}$. Many classifiers then have the form

$$h(x) = \text{sign}(H(x))$$

for some function $H(x)$. For example, a linear classifier corresponds to $H(x) = \beta^T x$. The risk can then be written as

$$R(h) = \mathbb{P}(Y \neq h(X)) = \mathbb{P}(YH(X) < 0) = \mathbb{E}(L(A))$$

where $A = YH(X)$ and $L(a) = I(a < 0)$. As a function of a, the loss $L(a)$ is discontinuous which makes it difficult to work with. Friedman, Hastie and Tibshirani (2000) show that AdaBoost corresponds to using a surrogate loss, namely, $L(a) = e^{-a} = e^{-yH(x)}$. Consider finding a classifier of the form $\sum_m \alpha_m h_m(x)$ by minimizing the exponential loss $\sum_i e^{-Y_i H(X_i)}$. If we do this iteratively, adding one function at a time, this leads precisely to AdaBoost. Typically, the classifiers h_j in the sum $\sum_m \alpha_m h_m(x)$ are taken to be very simple classifiers such as small classification trees.

The argument in Friedman, Hastie and Tibshirani (2000) is as follows. Consider minimizing the expected loss $J(F) = \mathbb{E}(e^{-YF(X)})$. Suppose our current estimate is F and consider updating to an improved estimate $F(x) + cf(x)$. Expanding around $f(x) = 0$,

$$\begin{aligned} J(F + cf) &= \mathbb{E}(e^{-Y(F(X)+cf(X))}) \approx \mathbb{E}(e^{-YF(X)}(1 - cYf(X) + c^2Y^2f^2(X)/2)) \\ &= \mathbb{E}(e^{-YF(X)}(1 - cYf(X) + c^2/2)) \end{aligned}$$

since $Y^2 = f^2(X) = 1$. Now consider minimizing the latter expression a fixed $X = x$. If we minimize over $f(x) \in \{-1, +1\}$ we get $f(x) = 1$ if $E_w(y|x) > 0$ and $f(x) = -1$ if

1. Input: $(X_1, Y_1), \ldots, (X_n, Y_n)$ where $Y_i \in \{-1, +1\}$.
2. Set $w_i = 1/n$ for $i = 1, \ldots, n$.
3. Repeat for $m = 1, \ldots, M$.
   (a) Compute the weighted error $\epsilon(h) = \sum_{i=1}^{n} w_i I(Y_i \neq h(X_i)$ and find h_m to minimize $\epsilon(h)$.
   (b) Let $\alpha_m = (1/2)\log((1-\epsilon)/\epsilon)$.
   (c) Update the weights:
   $$w_i \leftarrow \frac{w_i e^{-\alpha_m Y_i h_m(X_i)}}{Z}$$
   where Z is chosen so that the weights sum to 1.
4. The final classifier is
$$h(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right).$$

Figure 3: AdaBoost

$E_w(y|x) < 0$ where $E_w(y|x) = E(w(x,y)y|x)/E(w(x,y)|x)$ and $w(x,y) = e^{-yF(x)}$. In other words, the optimal f is simply the Bayes classifier with respect to the weights. This is exactly the first step in AdaBoost. If we fix now fix $f(x)$ and minimize over c we get

$$c = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)$$

where $\epsilon = E_w(I(Y \neq f(x)))$. Thus the updated $F(x)$ is

$$F(x) \leftarrow F(x) + cf(x)$$

as in AdaBoost. When we update F this way, we change the weights to

$$w(x,y) \leftarrow w(x,y)e^{-cf(x)y} = w(x,y)\exp\left(\log\left(\frac{1-\epsilon}{\epsilon}\right) I(Y \neq f(x))\right)$$

which again is the same as AadBoost.

Seen in this light, boosting really combines two ideas. The first is the use of surrogate loss functions. The second is greedy function approximation.

8 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 5 *Let $h_* \in \mathcal{H}$ minimize $R(h) = \mathbb{P}(Y \neq h(X))$. Then*

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

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.

9 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 4, 5, 6, 7.

10 Sparse Nonparametric Logistic Regression

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

$$\mathbb{P}(Y = 1 \,|\, X) \equiv p(X; f) = \frac{\exp\left(\sum_{j=1}^p f_j(X_j)\right)}{1 + \exp\left(\sum_{j=1}^p f_j(X_j)\right)} \tag{17}$$

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

$$\ell(f) = \mathbb{E}\left[Y f(X) - \log\left(1 + \exp f(X)\right)\right] \tag{18}$$

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.

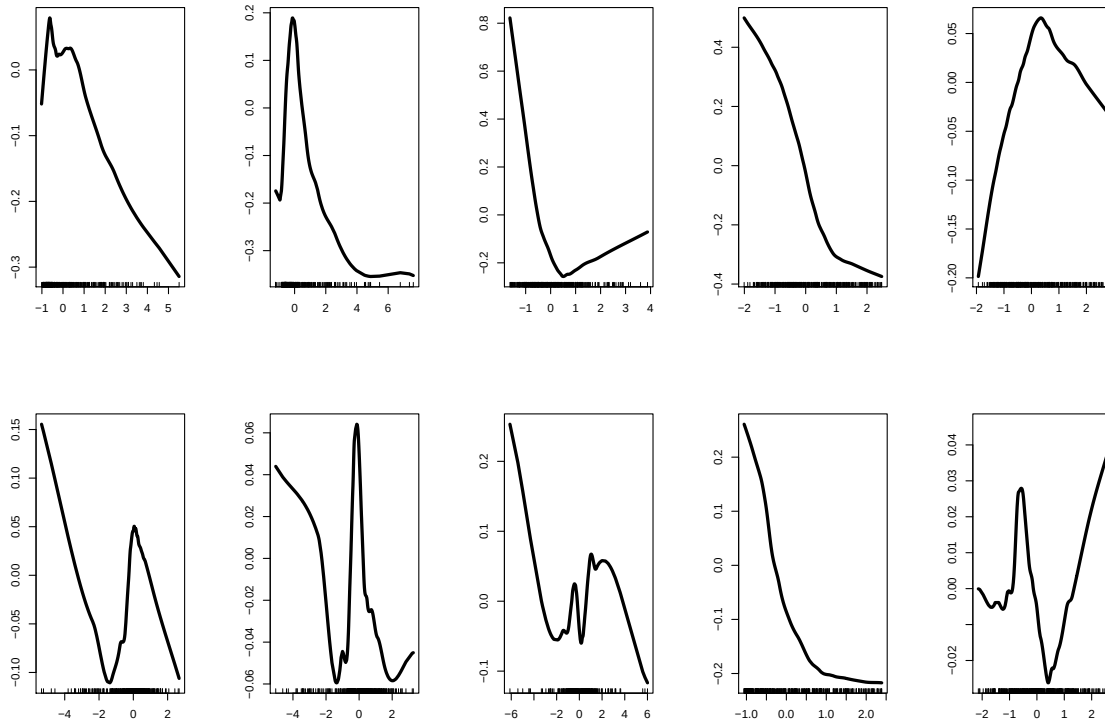

Figure 4: Estimated functions for additive model.

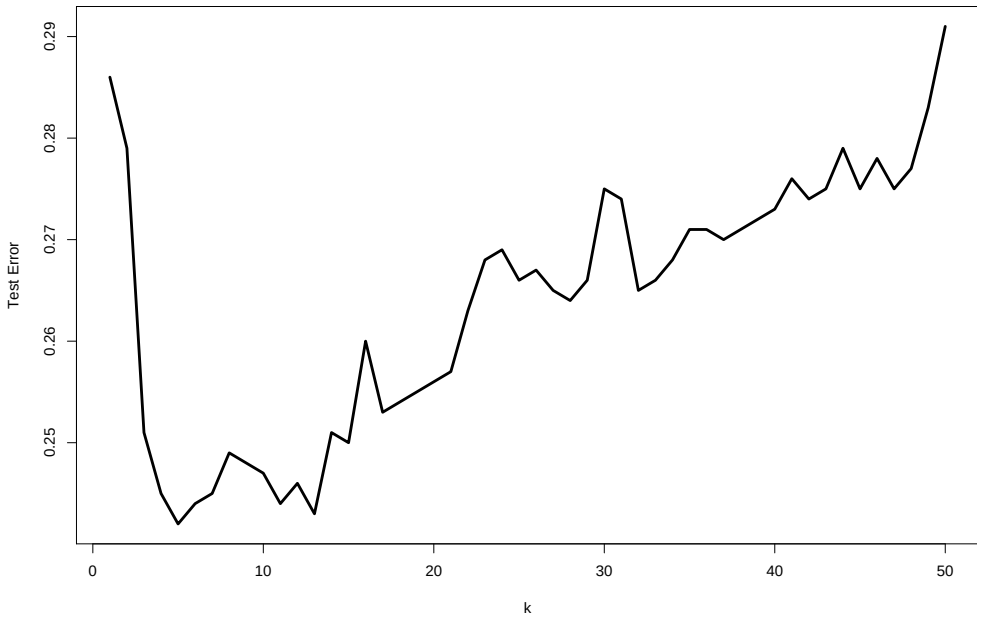

Figure 5: Test error versus k for nearest neighbor estimator.

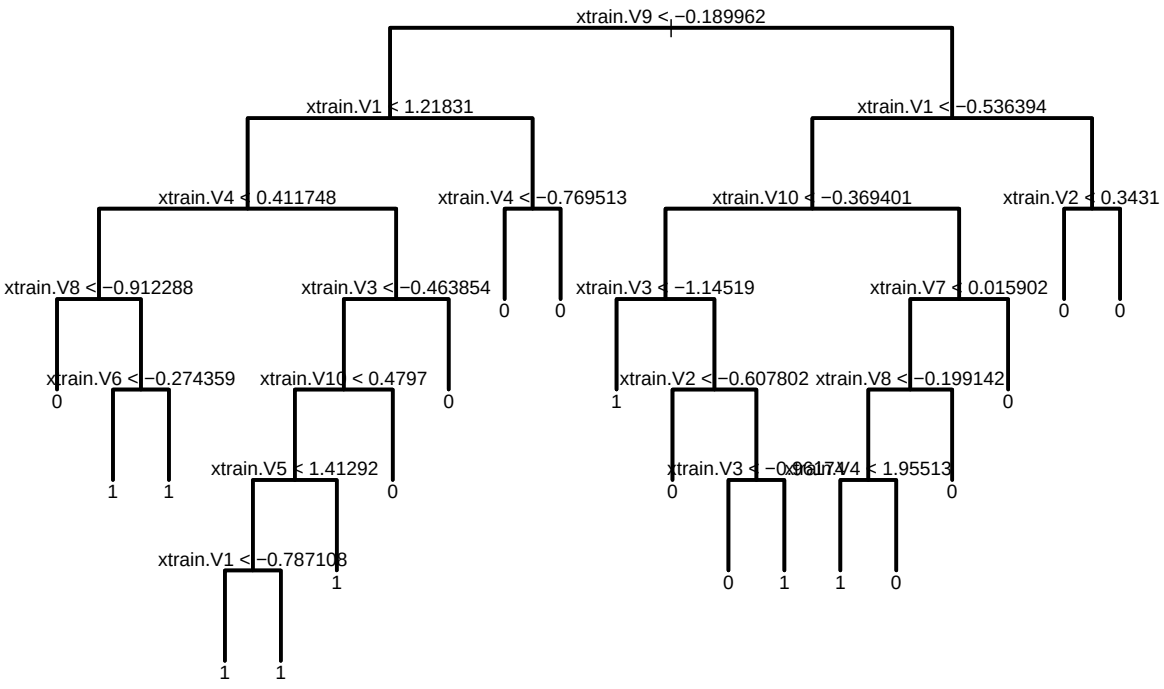

Figure 6: Full tree.

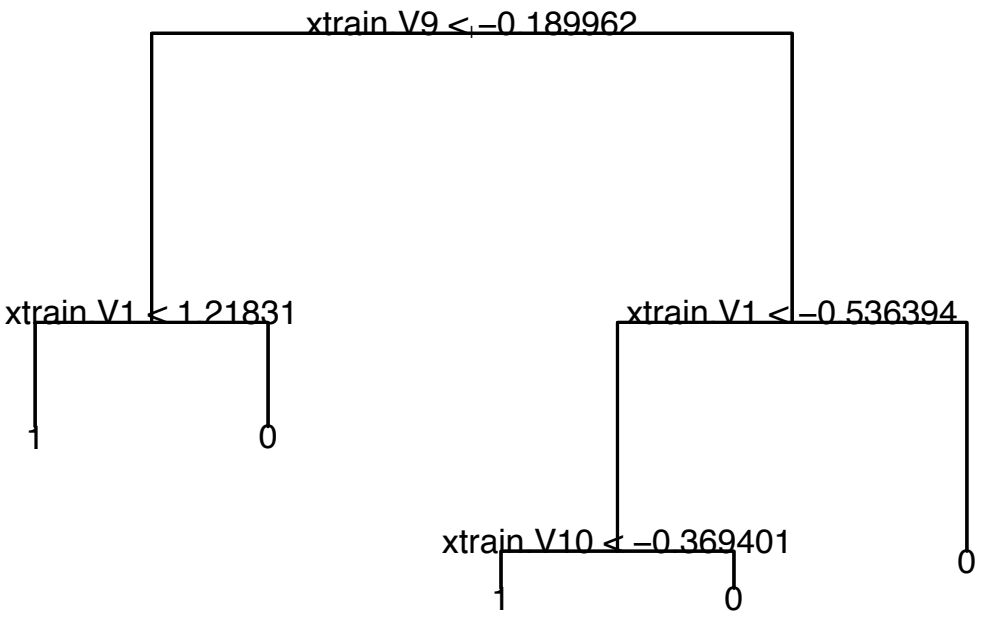

Figure 7: Classification tree. The size of the tree was chcosen by cross-validation.

where $f(X) = \sum_{j=1^p} f_j(X_j)$. To fit this model, the local scoring algorithm runs the backfitting procedure within Newton's method. One iteratively computes the transformed response for the current estimate $\widehat{f}$

$$Z_i = \widehat{f}(X_i) + \frac{Y_i - p(X_i; \widehat{f})}{p(X_i; \widehat{f})(1 - p(X_i; \widehat{f}))} \tag{19}$$

and weights $w(X_i) = p(X_i; \widehat{f})(1 - p(X_i; \widehat{f})$, and carries out a weighted backfitting of (Z, X) with weights w. The weighted smooth is given by

$$\widehat{P}_j = \frac{\mathcal{S}_j(wR_j)}{\mathcal{S}_j w}. \tag{20}$$

where $\mathcal{S}_j$ is a linear smoothing matrix, such as a kernel smoother. This extends iteratively reweighted least squares to the nonparametric setting.

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

$$\mathcal{L}(f, \lambda) = \mathbb{E}\left[\log\left(1 + e^{f(X)}\right) - Yf(X)\right] + \lambda\left(\sum_{j=1}^{p} \sqrt{\mathbb{E}(f_j^2(X_j))} - L\right) \tag{21}$$

and the stationary condition for component function f_j is $\mathbb{E}\left(p - Y \mid X_j\right) + \lambda v_j = 0$ where v_j is an element of the subgradient $\partial\sqrt{\mathbb{E}(f_j^2)}$. As in the unregularized case, this condition is

nonlinear in f, and so we linearize the gradient of the log-likelihood around $\widehat{f}$. This yields the linearized condition $\mathbb{E}\left[w(X)(f(X) - Z) \,|\, X_j\right] + \lambda v_j = 0$. To see this, note that

$$\begin{aligned} 0 &= \mathbb{E}\left(p(X;\widehat{f}) - Y + p(X;\widehat{f})(1 - p(X;\widehat{f}))(f(X) - \widehat{f}(X)) \,|\, X_j\right) + \lambda v_j \quad (22) \\ &= \mathbb{E}\left[w(X)(f(X) - Z) \,|\, X_j\right] + \lambda v_j \quad (23) \end{aligned}$$

When $\mathbb{E}(f_j^2) \neq 0$, this implies the condition

$$\left(\mathbb{E}\left(w \,|\, X_j\right) + \frac{\lambda}{\sqrt{\mathbb{E}(f_j^2)}}\right) f_j(X_j) = \mathbb{E}(wR_j \,|\, X_j). \quad (24)$$

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

$$f_j = \frac{\mathcal{S}_j(wR_j)}{\mathcal{S}_j w + \lambda \Big/ \sqrt{\mathbb{E}(f_j^2)}}. \quad (25)$$

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:

$$f_j \leftarrow \frac{\mathcal{S}_j(wR_j)}{\mathcal{S}_j w + \lambda\sqrt{n}\,/\|f_j\|}. \quad (26)$$

When $\lambda = 0$, this yields the standard local scoring update (20).

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.*

The results of a typical run of logistic SpAM are summarized in Figure 8, using plug-in bandwidths. A held-out set is used to tune the regularization parameter λ.

11 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:

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

$\lambda(\times 10^{-3})$	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

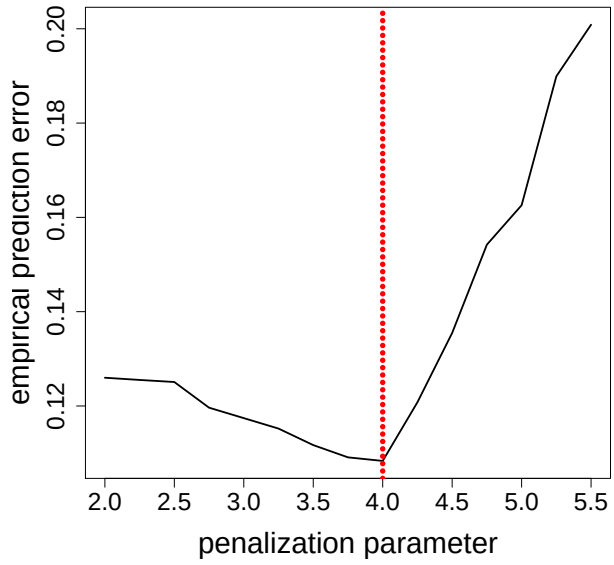

Figure 8: (Email spam) Classification accuracies and variable selection for logistic SpAM.

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.

Appendix: Multiclass Sparse Logistic Regression

Now we consider the multiclass version. Suppose we have the nonparametric K-class logistic regression model

$$p_f(Y = \ell \,|\, X) = \frac{e^{f_\ell(X)}}{\sum_{m=1}^K e^{f_m(X)}} \quad \ell = 1, \ldots, K \tag{27}$$

where each function has an additive form

$$f_\ell(X) = f_{\ell 1}(X_1) + f_{\ell 2}(X_2) + \cdots + f_{\ell p}(X_p). \tag{28}$$

In Newton's algorithm, we minimize the quadratic approximation to the log-likelihood

$$L(f) \;\approx\; L(\widehat{f}) + \mathbb{E}\left[(Y - \widehat{p})^T (f - \widehat{f})\right] + \frac{1}{2}\mathbb{E}\left[(f - \widehat{f})^T H(\widehat{f})(f - \widehat{f})\right] \tag{29}$$

where $\widehat{p}(X) = (p_{\widehat{f}}(Y = 1 \,|\, X), \ldots, p_{\widehat{f}}(Y = K \,|\, X))$, and $H(\widehat{f}(X))$ is the Hessian

$$H(\widehat{f}) = -\mathrm{diag}(\widehat{p}(X)) + \widehat{p}(X)\widehat{p}(X)^T. \tag{30}$$

Maximizing the right hand size of (29) is equivalent to minimizing

$$-\mathbb{E}\left((Y - \widehat{p})^T (f - \widehat{f})\right) - \mathbb{E}\left(\widehat{f}^T J f\right) + \frac{1}{2}\mathbb{E}\left(f^T J f\right) \tag{31}$$

which is, in turn, equivalent to minimizing the surrogate loss function

$$Q(f, \widehat{f}) \equiv \;=\; \frac{1}{2}\mathbb{E}\left(\|Z - Af\|_2^2\right). \tag{32}$$

where $J = -H(\widehat{f})$, $A = J^{1/2}$, and Z is defined by

$$\begin{aligned} Z &= J^{-1/2}(Y - \widehat{p}) + J^{1/2}\widehat{f} && (33)\\ &= A^{-1}(Y - \widehat{p}) + A\widehat{f}. && (34) \end{aligned}$$

The above calculation can be reexpressed as follows, which leads a multiclass backfitting algorithm. The difference in log-likelihoods for functions $\{\widehat{f}_\ell\}$ and $\{f_\ell\}$ is, to second order,

$$\mathbb{E}\left[\sum_{\ell=0}^{K-1} p_\ell(X)\left(\widehat{f}_\ell(X) - \sum_{k=0}^{K-1} p_k(X)\widehat{f}_k(X) + \frac{Y_\ell - p_\ell(X)}{p_\ell(X)} - f_\ell(X) + \sum_{k=0}^{K-1} p_k(X) f_k(X)\right)^2\right] \tag{35}$$

where $p_\ell(X) = \mathbb{P}(Y = \ell \,|\, X)$, and $Y_\ell = \delta(Y, \ell)$ are indicator variables. Minimizing over $\{f_\ell\}$ gives *coupled* equations for the functions f_ℓ; they can't be solved independently over ℓ.

A practical approach is to use coordinate descent, computing the function f_ℓ holding the other functions $\{f_k\}_{k \neq \ell}$ fixed, and iterating. Assuming that $f_k = \widehat{f}_k$ for $k \neq \ell$, this simplifies to

$$\mathbb{E}\left[p_\ell(1-p_\ell)^2\left(\widehat{f}_\ell + \frac{Y_\ell - p_\ell}{p_\ell(1-p_\ell)} - f_\ell\right)^2 + \sum_{k \neq \ell} p_k p_\ell^2 \left(\widehat{f}_\ell + \frac{p_k - Y_k}{p_k p_\ell} - f_\ell\right)^2\right]. \tag{36}$$

After some algebra, this can be seen to be the same as the usual objective function in the binary case, where we take $\widehat{f}_0 = 1$ and $\widehat{f}_1$ arbitrary.

Now assume f_ℓ (and $\widehat{f}_\ell$) has an additive form: $f_\ell(X) = \sum_{j=1}^p f_{\ell j}(X_j)$. Some further calculation shows that minimizing over each $f_{\ell j}$ yields the following backfitting algorithm:

$$f_{\ell j}(X_j) \leftarrow \frac{\mathbb{E}\left[p_\ell(1-p_\ell)\left(\widehat{f}_\ell - \sum_{k \neq j} f_{\ell k} + \frac{Y_\ell - p_\ell}{p_\ell(1-p_\ell)}\right) \,|\, X_j\right]}{\mathbb{E}\left[p_\ell(1-p_\ell) \,|\, X_j\right]}. \tag{37}$$

We approximate the conditional expectations by smoothing, as usual:

$$f_{\ell j}(x_j) \leftarrow \frac{\mathcal{S}_j(x_j)^T\left(w_\ell(X)\, R_{\ell j}(X)\right)}{\mathcal{S}_j(x_j)^T\left(w_\ell(X)\right)} \tag{38}$$

where

$$R_{\ell j}(X) = \widehat{f}_\ell(X) - \sum_{k \neq j} f_{\ell k}(X_k) + \frac{Y_\ell - p_\ell(X)}{p_\ell(X)(1-p_\ell(X))} \tag{39}$$

$$w_\ell(X) = p_\ell(X)(1-p_\ell(X)). \tag{40}$$

This is the same as in binary logistic regression. We thus have the following algorithm:

Multiclass Logistic Backfitting

1. Initialize $\{\widehat{f}_\ell = 0\}$, and set $Z(X) = K$.
2. Iterate until convergence:

   For each $\ell = 0, 1, \ldots, K-1$

   A. Initialize $f_\ell = \widehat{f}_\ell$

   B. Iterate until convergence:

For each $j = 1, 2, \ldots, p$

$$
\begin{aligned}
f_{\ell j}(x_j) &\leftarrow \frac{\mathcal{S}_j(x_j)^T \left(w_\ell(X)\, R_{\ell j}(X)\right)}{\mathcal{S}_j(x_j)^T \left(w_\ell(X)\right)} \text{ where} \\
R_{\ell j}(X) &= \widehat{f}_\ell(X) - \sum_{k \neq j} f_{\ell k}(X_k) + \frac{Y_\ell - p_\ell(X)}{p_\ell(X)(1 - p_\ell(X))} \\
w_\ell(X) &= p_\ell(X)(1 - p_\ell(X)).
\end{aligned}
$$

C. Update $Z(X) \leftarrow Z(X) - e^{\widehat{f}_\ell(X)} + e^{f_\ell(X)}$.

D. Set $\widehat{f}_\ell \leftarrow f_\ell$.

Incrementally updating the normalizing constants (step C) is important so that the probabilties $p_\ell(X) = e^{\widehat{f}_\ell(X)}/Z(X)$ can be efficiently computed, and we avoid an $O(K^2)$ algorithm. This can be extended to include a sparsity constraint, as in the binary case.