Logistic Response Projection Pursuit

CHARLES B. ROOSEN
Department of Statistics
Stanford University
Stanford, CA

TREVOR J. HASTIE
Statistics and Data Analysis Research Department
AT&T Bell Laboratories
Murray Hill, NJ

August 6, 1993
AT&T Bell Laboratories
Document No. BL011214-930806-09TM

Abstract

A highly flexible nonparametric regression model for predicting a response y given covariates $\mathbf{x}$ is the projection pursuit regression (PPR) model $\hat{y} = h(\mathbf{x}) = \beta_0 + \sum_j \beta_j f_j(\boldsymbol{\alpha}_j^T \boldsymbol{x})$, where the f_j are general smooth functions with mean zero and norm one, and $\sum_{k=1}^{d} \alpha_{kj}^2 = 1$. With a binary response y, the common approach to fitting a PPR model is to fit $\hat{y}$ to minimize average squared error without explicitly considering the binary nature of the response. We develop an alternative logistic response projection pursuit model, in which y is take to be binomial(p), where $\log(\frac{p}{1-p}) = h(\mathbf{x})$. This may be fit by minimizing either binomial deviance or average squared error. We compare the logistic response models to the linear model on simulated data.

In addition, we develop a generalized projection pursuit framework for exponential family models. We also present a smoothing spline based PPR algorithm, and compare it to supersmoother and polynomial based PPR algorithms.

1 Introduction

An important problem in statistics and machine learning is classification. In the simplest form of classification, we have some information on an object and want to classify it as being in one of two classes based on this information. Formulating the problem mathematically, we have a vector of observed information $\mathbf{x}$ on an individual and a response y which we define as being 1 if the individual is in class A and 0 if the individual is in class B. Using a set of known $(\mathbf{x}, y)$ combinations we want to develop a method for classifying a new individual as A or B based on the individual's $\mathbf{x}$ values.

A standard way to formulate this classification problem is as one of estimating $p = P(\textit{indiv in class A } |\mathbf{x}) = P(y = 1|\mathbf{x}) = E(y|\mathbf{x})$. We then classify an individual as being in class A if our estimate $\hat{p}$ for this individual is greater than 0.5, otherwise we classify the individual as being in class B.

The canonical way to predict p given $\mathbf{x}$ is through the logistic regression model. In this model, we suppose that p is related to $\mathbf{x}$ by:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \tag{1}$$

Here d is the number of predictors x_j. The lefthand side of the equation is known as the logistic link function, and is abbreviated $\text{logit}(p)$. The above model thus assumes that $\text{logit}(p)$ is approximately a hyperplane in the x's. The parameters β are fit by maximum likelihood, where y is distributed Binomial(p), with p specified by (1). That is, the parameters β are found which minimize the binomial deviance:

$$D(y; \hat{p}) = -2\sum_{i=1}^{N}(y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)) \tag{2}$$

This is the binomial response case of the generalized linear models (GLM) described by McCullagh and Nelder (1989).

Often, the underlying relationship between $\mathbf{x}$ and p is not well modelled by a hyperplane in the x's. Thus the model is modified to the more general form

$$\log\left(\frac{p}{1-p}\right) = h(\mathbf{x}) \tag{3}$$

where $h(\mathbf{x})$ is some nonlinear function of the x's. Simple forms of $h(\mathbf{x})$ include polynomials in the x's, or sums of transforms of the x's.

Viewed in this form, a major issue in model specification is determining a suitable function $h(\mathbf{x})$. If we have little subject-area based information on the expected relationship between $\mathbf{x}$ and y, we may want to use a flexible modelling procedure which will let the data guide the specification of $h(\mathbf{x})$. One such procedure is Hastie and Tibshirani's generalized additive models (Hastie and Tibshirani, 1990), in which logit(p) is taken to be additive in smooth functions of the predictors:

$$h(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} f_j(x_j) \tag{4}$$

The smooth functions f_j are estimated using scatterplot smoothers, and are usually restricted to have mean zero for identifiability.

Although the generalized additive models (GAM) approach has a number of attractive properties, not all possible functions $h(\mathbf{x})$ can be modelled as a sum of smooth functions as in (4). The GAM procedure can at best find the additive function closest to the true h in function space. In particular, an additive model is not invariant to rotations of the predictor space. The additive model may be thought of as a special case of a more general model in which these concerns are addressed. This is the projection pursuit regression (PPR) model (Friedman and Stuetzle, 1981), in which the response is modelled as a sum of smooth functions of linear combinations of the predictors:

$$h(\mathbf{x}) = \beta_0 + \sum_{j=1}^{M} \beta_j f_j\left(\boldsymbol{\alpha}_j^T \boldsymbol{x}\right) \tag{5}$$

with $\sum_{k=1}^{d} \alpha_{kj}^2 = 1$, $E(f_j) = 0$, and $E(f_j^2) = 1$. The PPR model has the property that, asymptotically in the number of linear combinations M, it can approximate any smooth function (Diaconis and Shahshahani, 1984).

The usual approach to fitting a PPR model to binary response data is to think of the problem as one of estimating a conditional expectation $p = E(y|\mathbf{x})$ by a PPR model. The binary nature of the problem is hence considered immaterial, and a model of the form $\hat{p} = h(\mathbf{x})$ is fit to minimize the residual sum of squares

$$RSS = \sum_{i=1}^{N} (y_i - \hat{p}_i)^2 \tag{6}$$

where $h(\mathbf{x})$ is the PPR model (5). This procedure works remarkably well (Ripley, 1992). However, it does not use the information that the response is binary, and does not restrict the fitted values to be between zero and one.

Other methods for classification using PPR have been proposed. Friedman has developed a version of PPR for classification based on a Bayes Risk measure of error (Friedman, 1984a). Flick et al. (1990) use PPR in a classification procedure based on estimating the likelihood ratio when the densities are unknown. Hastie, Tibshirani, and Buja's flexible discriminant analysis (1993) could also be used with PPR for classification.

This paper explores the use of a logistic regression framework with the PPR model, with the thought that a procedure specifically tailored to binary response may more successfully approximate the underlying surface relating $\mathbf{x}$ and y. We describe a *generalized projection pursuit* procedure analogous to GLM, and use it to compare three PPR-based procedures for binary response data.

2 Generalized Projection Pursuit

2.1 Basic Framework

A generalized linear model has three basic components. The *random component* describes the probabilistic structure of y, e.g. $y_i \sim N(\mu_i, \sigma^2)$. The *systematic component* gives the relationship between the covariates $\mathbf{x}$ and the predictor $\eta(\mathbf{x})$. In a GLM, the systematic component is the linear function $\eta(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}$. The *link function*, $g(\mu_i) = \eta(\mathbf{x}_i)$, relates the random and systematic components.

Many common models can be formulated within this framework. For instance, the standard multiple regression model assumes

$$y_i = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$. This is a GLM with random component $y_i \sim N(\mu_i, \sigma^2)$, systematic component $\eta(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}$, and the identity link function $\mu_i = g(\mu_i) = \eta(\mathbf{x}_i)$. Note that this model reformulates the linear least squares problem as a GLM through a Gaussian error model.

The logistic linear regression model parameterizes the mean by p_i rather than μ_i. The random component is $y_i \sim \text{Bin}(p_i)$, the systematic component

is again $\eta(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}$, and the link function is the logistic link $\log(\frac{p}{1-p}) = \eta(\mathbf{x})$. The logistic additive model replaces the systematic component by the additive function (4).

In a generalized projection pursuit (GPP) model, the systematic component is set to be $\eta(\mathbf{x}) = h(\mathbf{x})$, where $h(\mathbf{x})$ is the projection pursuit function (5). The standard PPR model attempts to find the least squares fit to the observed y's. PPR is thus a GPP model with a random component $y_i \sim N(\mu_i, 1)$, a systematic component which is a projection pursuit function, and an identity link function.

A logistic projection pursuit (LPP) model parallels a logistic regression model, in that it has a random component $y_i \sim \text{Bin}(p_i)$, a systematic component which is a projection pursuit function, and a logistic link. This differs from the standard PPR model in two important ways. The first is that it constrains the fitted values to be between zero and one. The second is that it attempts to minimize the binomial deviance (2) rather than the residual sum of squares (6). By minimizing deviance, the method attempts to fit the projection pursuit model $h(\mathbf{x})$ using maximum likelihood for a binary response model as formulated above.

Friedman (personal communication, 1993) argues that for the classification problem, it is preferable to minimize squared error rather than binomial deviance. This suggests a third model, in which we use a logistic link and projection pursuit systematic component, but have a Gaussian random component. We refer to this as the *squared error logistic response* model (SELR). This is analogous to the binary response model usually used with feedforward neural networks, in which the final response is mapped to $[0, 1]$ by a sigmoidal function, and the error criterion is squared error on the $[0, 1]$ scale.

Before making additional comparisons between the PPR, LPP, and SELR models, it is useful to develop the parameter estimation method for GPP.

2.2 Parameter and Function Estimation

2.2.1 Iteratively Reweighted Least Squares

A common procedure for fitting the β's in a GLM is the Iteratively Reweighted Least Squares (IRLS) algorithm (McCullagh and Nelder, 1989, pp. 40–43). This uses a Fisher scoring method, which in an exponential family with the canonical link is equivalent to the Newton-Raphson method. In the IRLS

procedure, we alternate between forming an adjusted dependent variable z with estimated variance w^{-1}, and regressing z on $\{x_j\}_{j=1}^d$ with weights w. Let η^0 be the current estimate of the linear predictor, with fitted value μ^0 derived from the link function $\eta = g(\mu)$. Then the adjusted dependent variable is

$$z = \eta^0 + (y - \mu^0)\left(\frac{d\eta}{d\mu}\right)_0$$

where the derivative of the link is evaluated at μ^0. The estimated variance of z is

$$w^{-1} = \left(\frac{d\eta}{d\mu}\right)_0^2 V^0$$

where V^0 is the variance function for y evaluated at μ^0.

Note that z is the first-order Taylor series approximation to $g(y)$ about μ.

Schematically, the IRLS algorithm is:

Loop until convergence of β

1. Calculate z and w given the current values of μ and η
2. Use weighted regression of z on $\{x_j\}_{j=1}^d$ with weights w to get new estimates β.
3. Update the estimates of μ and η given the new β's.

End Loop

In the Gaussian case with identity link, we get $z = y$ and $w^{-1} = \sigma^2$, so the IRLS procedure consists of a single regression of y on $\{x_j\}_{j=1}^d$.

In the logistic regression case, $g(p) = \log(\frac{p}{1-p})$ and hence $\frac{dg(p)}{dp} = \frac{1}{p(1-p)}$. Also, $V^0 = p(1-p)$. The response z is thus:

$$z = \eta + \frac{(y-p)}{p(1-p)}$$

with weights $w = p(1-p)$.

2.2.2 Transformations of the Additive Model

A method for optimizing over both functions f and coefficients $\boldsymbol{\alpha}$ within the IRLS context is provided in Hastie and Tibshirani's GAM Tech Report (1984, pp. 30–32) as "Transformations of the Additive Model". This section describes the method for a linear predictor.

In the basic GLM model we impose a particular link function $g(\mu) = \eta = \boldsymbol{\alpha}^T \boldsymbol{x}$. Here g is a known link function, and $\boldsymbol{\alpha}$ the fitted linear combination coefficients. A more general model is $g(\mu) = f(\eta) = f(\boldsymbol{\alpha}^T \boldsymbol{x})$, where f is an unspecified smooth function. The motivation for this approach was that we can examine whether the link g is correctly specified by fitting this more general model and examining whether the fitted f is essentially the identity function. If f departs significantly from the identity, then the validity of the specified link g is brought into question.

Hastie and Tibshirani modify the IRLS algorithm to include two alternating fit updates, one each for estimating $\boldsymbol{\alpha}$ and f. As before, the results for exponential families reduce conveniently. Let η^0 and μ^0 denote the current fits. The current estimate of $\mathrm{Var}(y_i)$ is V_i^0.

We estimate $\boldsymbol{\alpha}$ with f fixed using a Gauss-Newton step. To obtain new $\boldsymbol{\alpha}_k$'s, perform regression through the origin with response

$$\eta^0(\mathbf{x}) f'(\eta^0) + (y - \mu^0) \cdot \frac{dg(\mu)}{d\mu^0} \tag{7}$$

weights $w^{-1} = (\frac{dg(\mu)}{d\mu^0})^2 V^0$, and predictors $\{x_k f'(\eta^0)\}_{k=1}^d$. It is often preferable to solve instead for the change in the coefficients $\boldsymbol{\delta}_k = \boldsymbol{\alpha}_k^1 - \boldsymbol{\alpha}_k^0$. To do so, we use regression through the origin with the same weights and predictors, but with response $(y - \mu^0)\frac{dg(\mu)}{d\mu^0}$. Note that we need derivative estimates $f'(\eta^0)$ evaluated at the current linear combinations. These are easily obtained for smoothers with a basis function representation. For other smoothers, such as the supersmoother, a derivative estimation technique based on first differences is typically used.

To estimate f with $\boldsymbol{\alpha}$ fixed, let $v = \boldsymbol{\alpha}^T \boldsymbol{x}$. The update is then

$$f^1(v) = \mathrm{Smooth}\left[f^0(v) + (y - \mu^0)\frac{dg(\mu)}{d\mu^0} \right] \tag{8}$$

with weights $w^{-1} = (\frac{dg(\mu)}{d\mu^0})^2 V^0$, and predictor v.

2.2.3 Projection Pursuit Regression

Consider a PPR model with a single linear combination, i.e. $M = 1$. The model we wish to fit is then $g(\mu) = \beta_0 + \beta_1 f_1(\boldsymbol{\alpha}^T \boldsymbol{x})$, where $g(\mu) = \mu$. We can combine the location coefficient β_0 and scale coefficient β_1 with the function $f_1(v_1)$ to get a model $g(\mu) = f(\boldsymbol{\alpha}^T \boldsymbol{x})$. Now for the PPR model $V^0 = 1$ and $\frac{dg(\mu)}{d\mu^0} = 1$. Using the updates from the previous section, we get $\boldsymbol{\delta}$ given f by regression through the origin with response $(y - \mu^0)$, weights $w = 1$, and predictors $\{x_k f'(v)\}_{k=1}^d$. To get f given $\boldsymbol{\alpha}$, we smooth y on v with weights $w = 1$.

Those familiar with PPR will note that this updating procedure agrees with the standard PPR algorithm (Friedman, 1984a). For $M > 1$, a backfitting procedure is used, so that we only really consider one term at a time. A complete algorithm for fitting a PPR model is described in Section 3.1.

2.2.4 Logistic Response Projection Pursuit Models

Again, for a single-term model, we wish to fit $g(p) = f(\boldsymbol{\alpha}^T \boldsymbol{x})$. In the logistic case we have $g(p) = \log(\frac{p}{1-p})$. So $\frac{dg(\mu)}{d\mu^0} = \frac{1}{p(1-p)}$. In the LPP model $V^0 = p^0(1 - p^0)$. We get $\boldsymbol{\delta}$ given f by regression through the origin with response $\frac{(y-p^0)}{p^0(1-p^0)}$, weights $w = p^0(1-p^0)$, and predictors $\{x_k f'(v)\}_{k=1}^d$. We get f given $\boldsymbol{\alpha}$ by smoothing $f^0(v) + \frac{(y-p^0)}{p^0(1-p^0)}$ on v with weights $w = p^0(1 - p^0)$.

The only change when using SELR is that the variance V^0 changes. With $V^0 = 1$, the predictors and responses are identical to those in LPP, but the weights are now $w = [p^0(1 - p^0)]^2$.

The next question is how to combine these updates with the IRLS loop. In particular, how often do we recalculate p^0? One approach would be to calculate p^0 after every change in f. However, this does not give the algorithm a chance to converge in $\boldsymbol{\alpha}$ and f before changing the response. A more conservative approach is to update $\boldsymbol{\alpha}$ and f until convergence at each p^0 value. Essentially, we replace the linear regression step in the IRLS loop with an iterative procedure which adjusts $\boldsymbol{\alpha}$ and f until convergence for the current p^0. Section 3 elaborates upon this looping structure, and discusses the incorporation of multiple terms into a projection pursuit model.

3 The Smoothing Spline Projection Pursuit Algorithm

This section describes a version of the projection pursuit algorithm which utilizes smoothing splines as the scatterplot smoothers. Smoothing splines perform penalized least squares with a smoothness penalty proportional to $\int (f'')^2 dx$. This penalizes for large curvature, and hence for changes in slope. The fitting criterion is:

$$RSS_\lambda = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f'')^2 dx$$

The parameter λ controls the smoothness of the fitted function. For small values of λ more weight is given to fitting the squared error portion of the criterion, and a rough fit results. As λ increases, more weight is given to keeping the curvature small, and a smoother fit results. For a fixed value of λ, this reduces to a least squares regression problem in which the basis functions are a complicated function of λ and the x_i's.

In this implementation, the user specifies an approximate number of degrees of freedom df_{ss} to use in each $f_j(v_j)$, where we define the degrees of freedom for a smooth as the trace of the smoother matrix (Chambers and Hastie, 1992, p. 574). The parameter λ is then set to give approximately the specified degrees of freedom per smooth function.

The overall algorithm is based on the projection pursuit algorithm described by Hwang et al. (1993), which is in turn based on the projection pursuit algorithm of Friedman (1984a). Section 3.1 describes the overall algorithm for the standard PPR case. Section 3.2 describes the GPP algorithm obtained by introducing local scoring loops into this PPR algorithm.

3.1 Projection Pursuit Regression

This section presents a smoothing spline version of the PPR algorithm. Our model is $\hat{y} = h(\mathbf{x})$ where $h(\mathbf{x})$ is the PPR model specified by (5). For the present discussion y is taken to be a continuous response, although the same algorithm applies for y in $\{0, 1\}$. The general approach is to fit the terms in a stepwise manner, with a good deal of backfitting between the addition of terms. A total of $M_{max} \geq M$ terms are fitted, and a backwards

selection procedure is then used to prune the fit down to M terms. The overall algorithm structure is presented, after which the components of the algorithm are described.

3.1.1 PPR Algorithm Structure

INITIALIZE VARIABLES

$\beta_0 \leftarrow \bar{y}$.

α_{kj}'s random uniform [-0.5,0.5], normalized so that $||\boldsymbol{\alpha}_j|| = 1$.

$f_j(v_j) \leftarrow 0$.

$\beta_j \leftarrow 1$.

ADD TERMS

For (j in 1 to M_{max}) {

Calculate Partial Residual $r \leftarrow y - \beta_0 - \sum_{m=1}^{j-1} \beta_m f_m(v_m)$.

Update Term j with response r.

Backfit Terms 1 to j.

}

PRUNE TERMS

For (j in M_{max} to $M + 1$) {

Drop Term of least importance.

Backfit Terms 1 to (j-1).

}

CLEAN UP

Adjust $f_j(v_j)$'s to have mean zero, changing β_0 accordingly.

3.1.2 Update Term

The Update Term routine is the primary fitting component in the algorithm. It fits a single term PPR model to the partial residual r using the updates specified by (7) and (8). If an initial $f_j(v_j)$ has not been fit, the first step is to fit a smooth function to r with predictor v_j. The routine then alternates between changing $\boldsymbol{\alpha}_j$ and changing $f_j(v_j)$, until the percent change in error is less than a specified tolerance ξ.

Note that the $\boldsymbol{\alpha}$ update (7) is guaranteed to be in a descent direction, but may be too large a change. One approach is to perform a line search for the appropriate step size. However, for each new value of $\boldsymbol{\alpha}$ we have new predictors v_j, and hence must reevaluate the function $f_j(v_j)$ in order to calculate the error. Rather than putting this effort into a line search, we put effort into a backfitting routine. If the step determined by (7) results in an increase in error after refitting $f_j(v_j)$, we discard the step and stop updating this term. Due to the backfitting, we will have the opportunity to update the term later with a different response.

Some bookkeeping details are that after getting the new $\boldsymbol{\alpha}_j$, we normalize $\boldsymbol{\alpha}_j$ to norm one before recalculating v_j. After fitting $f_j(v_j)$, we adjust the fitted values to have norm one, changing the β_j scale coefficient to keep the overall fit the same. At this point we also recalculate our derivative estimates $f_j'(v_j)$.

This same Update Term routine is used in the GPP model, with the input response r depending on the other terms in the model and on the current working response for local scoring. Weights are introduced to the regression and smoothing, and the weighted squared error is monitored for convergence.

3.1.3 Backfit Terms

The Backfit Terms algorithm uses a backfitting procedure to adjust the j terms previously fitted. As in Update Term, the procedure is repeated until the percent change in error is less than the tolerance ξ. The first step in backfitting is to recalculate the β's by regressing y on $\{f_m(v_m)\}_{m=1}^{j}$. The second step is to loop over the j terms, where for the m^{th} term we call Update Term with response r the partial residual $r = y - \beta_0 - \sum_{j \neq m} \beta_j f_j(v_j)$. Note that the first step cannot increase the error, as the current β's are a valid choice for the fitted parameters, and the second step cannot increase the

error, as the Update Term routine does not change a term if doing so would increase the error.

In the GPP algorithm, the working response z from local scoring is used in place of y, weights are used in the regression and smoothing, and the weighted squared error is monitored for convergence.

3.1.4 Drop Term

The magnitude of the scale coefficient β_j is used as a proxy for the importance of term j to the overall fit. In the pruning step, the term with smallest absolute β_j is selected as the term to drop from the fit. This is the approach taken by Friedman (1984a) and Hwang et al. (1993). Another approach (which we have not yet implemented) would be to select the term to drop based on its t-statistic from the regression of y on $\{f_m(v_m)\}_{m=1}^{j}$. This approach would take into account information on correlations between the $f_j(v_j)$'s which is ignored when considering merely the magnitude of the scale coefficients.

3.2 Generalized Projection Pursuit Models

The GPP algorithm adds local scoring loops around the backfitting loops. So the procedure is analogous to the GLM procedure with the linear regression replaced by a call to Backfit Terms, and some additional work done in getting initial values and pruning.

3.2.1 GPP Algorithm Structure

INITIALIZE VARIABLES

$\beta_0 \leftarrow g(\bar{y})$.

α_{kj}'s random uniform [-0.5,0.5], normalized so that $||\boldsymbol{\alpha}_j|| = 1$.

$f_j(v_j) \leftarrow 0$.

$\beta_j \leftarrow 1$.

ADD TERMS

For (j in 1 to M_{max}) {

Calculate Adjusted Dependent Variable z with weights w.

Calculate Partial Residual $r \leftarrow z - \beta_0 - \sum_{m=1}^{j-1} \beta_m f_m(v_m)$.

Update Term j with response r.

Backfit Terms 1 to j within Local Scoring.

}

PRUNE TERMS

For (j in M_{max} to $M+1$) {

Drop Term of least importance.

Calculate Adjusted Dependent Variable z with weights w.

Backfit Terms 1 to (j-1) within Local Scoring.

}

CLEAN UP

Adjust $f_j(v_j)$'s to have mean zero, changing β_0 accordingly.

3.2.2 Calculate Adjusted Dependent Variable

It is in calculating the adjusted dependent variable and weights that differing error and link structures come into play. As specified in Section 2.2.1, we calculate the working response

$$z = \eta^0 + (y - \mu^0) \left(\frac{d\eta}{d\mu^0} \right)_0$$

and weights $w^{-1} = \left(\frac{d\eta^0}{d\mu^0} \right)_0^2 V^0$, where V_i^0 is the current variance estimate for y_i, η^0 is the fit $h(\mathbf{x})$ on the PPR scale, and $\mu^0 = g^{-1}(\eta^0)$ is the fit on the mean scale. For the three models which we are considering, the z and w updates are:

Model	z	w
PPR	y	1
LPP	$\eta + \frac{(y-p)}{p(1-p)}$	$p(1-p)$
SELR	$\eta + \frac{(y-p)}{p(1-p)}$	$[p(1-p)]^2$

where $\eta = h(\mathbf{x})$ and $p = \frac{1}{1+exp^{-\eta}}$.

3.2.3 Backfit Terms within Local Scoring

In the GPP algorithm, we imbed the Backfit Terms routine within a local scoring loop. That is, we alternate updating z and w based on the current fit as described above, with adapting the fit to this new response using Backfit Terms. Again, we iterate until the percent change in weighted squared error is less than ξ.

3.2.4 General Comments

The current version of GPP only implements these three particular models. To implement other single-response GPP models, the only part of the algorithm which needs to be changed is the procedure for updating z and w. For a multiple-response model, a GPP algorithm could be based on a multiple-response PPR algorithm. Friedman (1984a) presents a multiple-response PPR algorithm based on having differing scale coefficients β_{cj} for each response y_c.

In PPR mode, the GPP procedure does slightly more optimization than does the PPR algorithm described in Section 3.1. This is due to an extra pass through the Backfit Terms loop when using local scoring.

4 Simulation Studies

4.1 Comparing Smoothing Spline PPR with Supersmoother and Polynomial PPR

The original version of PPR proposed by Friedman and Stuetzle (1981) utilized Friedman's supersmoother (1984b). Since then, versions of PPR have been developed using other smoothers. Flick et al. (1990) develop an algorithm suitable for use with a variety of basis functions. Hwang et al. (1993) use polynomial basis functions. As Hwang et al. have already used supersmoother and polynomial based PPR algorithms on a number of functions, their work provides a valuable test bed in which to examine the smoothing spline PPR algorithm.

4.1.1 Simulation Protocols

Nonlinear Functions: Hwang et al. investigate the supersmoother and polynomial PPR algorithms on five nonlinear functions $g^{(j)} : [0,1]^2 \to R$. The functions are scaled to have standard deviation 1 and translated to have nonnegative range. They are plotted in Figure 1, and their equations are given below:

- *Simple Interaction Function*

$$g^{(1)}(x_1, x_2) = 10.391((x_1 - 0.4) \cdot (x_2 - 0.6) + 0.36)$$

- *Radial Function*

$$g^{(2)}(x_1, x_2) = 24.234(r^2(0.75 - r^2)), \qquad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$$

- *Harmonic Function*

$$g^{(3)}(x_1, x_2) = 42.659((2 + x_1)/20 + Re(z^5))$$

  where $z = x_1 + ix_2 - 0.5(1 + i)$, or equivalently, with $\tilde{x}_1 = x_1 - 0.5$, $\tilde{x}_2 = x_2 - 0.5$,

$$g^{(3)}(x_1, x_2) = 42.659(0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4))$$

- *Additive Function*

$$\begin{aligned} g^{(4)}(x_1, x_2) = 1.3356(&1.5(1 - x_1) + e^{2x_1 - 1}\sin(3\pi(x_1 - 0.6)^2) \\ &+ e^{3(x_2 - 0.5)}\sin(4\pi(x_2 - 0.9)^2)) \end{aligned}$$

- *Complicated Interaction Function*

$$g^{(5)}(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2))$$

Training Data: Hwang et al. generate a single set of 225 pairs (x_{i1}, x_{i2}) with each component an independent uniform random variable on $[0, 1]$. They then trained the algorithms twice for each function, once in the noiseless situation

$$y_i^{(j)} = g^{(j)}(x_{i1}, x_{i2}), \quad i = 1, \ldots, 225, \quad j = 1, \ldots, 5$$

and once in the slightly noisy situation

$$y_i^{(j)} = g^{(j)}(x_{i1}, x_{i2}) + 0.25\varepsilon_i, \quad i = 1, \ldots, 225, \quad j = 1, \ldots, 5$$

where the ε_i's are *i.i.d.* $N(0, 1)$. They appear to use a single set of $\mathbf{x}$'s and ε's, running the algorithm once for each smoother/function/noise-level combination.

We employ the same basic setup, but perform multiple runs in order to get information on the variability due to the location of the $\mathbf{x}$'s. We generate ten sets of 250 pairs (x_{i1}, x_{i2}), each with a corresponding 250 ε_i's.

Parameter Settings: We examine the $M = 3$ case, in which $M_{max} = 5$. The tolerance on the percent change in error was set to $\xi = 0.005$. To obtain approximately the same number of degrees of freedom per projection as was done in the polynomial model, we used $df_{ss} = 10$ in the noiseless case, and $df_{ss} = 8$ in the noisy case. This corresponds to polynomials of degree 9 and 7 respectively. (A single run of the algorithm was also performed at $df_{ss} = 9$ and $df_{ss} = 7$ respectively, with results superior to the averages over 10 runs presented below. However, as a single case provides little information on the overall performance of the algorithm, those results are not presented here.)

Evaluation of the Fit: To test the accuracy of the fitted surface, a test set of size $N = 10,000$ was generated over a regularly spaced grid on $[0, 1]^2$, with

$$x_{ir} = (2i - 1)/200, \quad i = 1, \ldots, 100, \quad r = 1, 2.$$

The error measure employed is the *fraction of variance unexplained* (FVU), which is given by

$$\text{FVU} = \frac{E(\hat{g}(\mathbf{x}_i) - g(\mathbf{x}_i))^2}{E(g(\mathbf{x}_i) - \bar{g}(\mathbf{x}_i))^2}.$$

where $g(\mathbf{x}_i)$ is the true value of the function and $\hat{g}(\mathbf{x}_i)$ is the fitted value. We evaluate the FVU for a fit by replacing the expectations with averages over

the grid of test set values. Note that the FVU is the mean squared error for the estimate $\hat{g}(\mathbf{x})$ scaled by the variance of the true function $g(\mathbf{x})$.

For the smoothing spline results, the FVU was calculated for each of the ten runs, then averaged to get a mean FVU and standard error of the mean for each function/noise-level combination.

4.1.2 Results

Results for the noiseless and noisy situations are presented in Tables 1 and 2 respectively. The smoothing spline columns include the mean FVU, and se(mean FVU), with the latter in parenthesis. The FVU values for supersmoother and polynomial PPR calculated by Hwang et al. are replicated here for comparison (from Tables 1 and 2 of (Hwang et al., 1993)).

Across the board, the smoothing spline and polynomial algorithms do better than the supersmoother, although the difference is not always statistically significant. If we take either 2 or 3 standard errors as a rough cutoff for significance, polynomial PPR does significantly better than the other methods on the Simple Interaction, Harmonic, and Complex Interaction functions with noiseless data. It is able to fit the Simple Interaction exactly. On the noisy data, polynomial PPR does significantly better than the other methods on the Harmonic function. However, the smoothing spline PPR does significantly better than the other methods on the Additive and Complex Interaction functions. This suggests that polynomials are useful in a low noise setting, but in the presence of noise, a locally adaptive smoother may do better.

One thing to note is that the smoothing spline and polynomial methods require the user to set the degrees of freedom in the smoother. The supersmoother picks the amount of smoothing to perform automatically. If the underlying functions had sharper local changes in curvature, the supersmoother might perform better. A current research topic of the authors is automatic smoothing parameter selection for the smoothing spline PPR model.

Noiseless Training Data			
Function	Smoothing Splines	Supersmoother	Polynomials
Simp Int	0.00007 (0.00001)	0.00035	0.00000
Radial	0.00669 (0.00231)	0.00973	0.00860
Harmonic	0.14982 (0.00615)	0.35454	0.07478
Additive	0.00069 (0.00015)	0.00101	0.00076
Comp Int	0.08315 (0.00927)	0.13459	0.04891

Table 1: FVU for Linear Response ($M = 3$, $M_{max} = 5$,$df_{ss} = 10$)

Noisy Training Data			
Function	Smoothing Splines	Supersmoother	Polynomials
Simp Int	0.00686 (0.00063)	0.00896	0.00767
Radial	0.02471 (0.00317)	0.03162	0.03072
Harmonic	0.21517 (0.01216)	0.36352	0.09885
Additive	0.01049 (0.00135)	0.33414	0.01841
Comp Int	0.10819 (0.00968)	0.26950	0.14633

Table 2: FVU for Linear Response ($M = 3$, $M_{max} = 5$,$df_{ss} = 8$)

4.2 Comparing PPR, LPP, and SELR for Binary Response

As the smoothing spline version of PPR performs satisfactorily, we may move on to examining the effect of different model structures when fitting binary response data.

4.2.1 Simulation Protocols

Nonlinear Functions: In the binary case we have

$$\log\left(\frac{p_i}{1-p_i}\right) = \nu(g^{(j)}(\mathbf{x}_i) - c_j)$$

where $g^{(j)}(\mathbf{x}_i)$ is one of the nonlinear functions described in the previous section. The parameter ν is a scale coefficient used to control the spread of the p's, and c_j is a constant selected to give a mean p of about 0.5 using function $g^{(j)}$. The values of c_j used are $\{3.6, 2.3, 4.4, 2.15, 2.7\}$ respectively.

Our observed value y_i is binary with probability p_i of being 1. We obtain p_i using

$$p_i = \text{logit}^{-1}(\nu(g^{(j)}(\mathbf{x}_i) - c_j)) = \frac{1}{1 + e^{-\nu(g^{(j)}(\mathbf{x}_i) - c_j)}}. \tag{9}$$

Those familiar with the neural networks literature will recognize this function as a common sigmoidal nonlinearity used in neural net models.

Training Data: The same ten sets of 250 pairs of $\mathbf{x}$'s were used. For each set of $\mathbf{x}$'s, probabilities p_i were obtained at two scale levels: $\nu = 5$ and $\nu = 1$. The first value leads to relatively pure regions of 0's and 1's, while the second leads to a "noisier" classification problem. For each set of p_i's we generated a training set of y_i's as random binomials. The randomness in the training data arises due to the random design of the $\mathbf{x}$'s, and the random generation of the y's. As before, we have ten runs at each of the ten function/purity-level combinations. A single sample with $\nu = 5$ is plotted in Figure 2, along with contour lines indicating the true decision boundaries at which $p = 0.5$. A corresponding plot with $\nu = 1$ is presented in Figure 3. (The *'s indicate values for which $y = 1$, and the 0's indicate values for which $y = 0$.)

Parameter Settings: For each set of test values, we ran the PPR, LPP, and SELR algorithms with $M = 3$, $M_{max} = 3$, $\xi = 0.005$, and $df_{ss} = 7$.

Evaluation of the Fit: The function (9) was evaluated on the same grid of test values $\mathbf{x}$ as before, to obtain p_{true}. For each of the three models, predicted values p_{pred} were obtained at the test values $\mathbf{x}$. Two error measure were calculated. The first was the FVU, comparing the fitted probability p_{pred} to the true probability p_{true}.

The second error measure was a proxy for the test set misclassification error. Each test value was said to have $y_{true} = 1$ if $p_{true} > 0.5$, and $y_{true} = 0$ otherwise. Similarly, the prediction $y_{pred} = 1$ if $p_{pred} > 0.5$, and $y_{pred} = 0$ otherwise. The misclassified observations are those for which $y_{true} \neq y_{pred}$. We report percent misclassified over the 10,000 test observations.

4.2.2 Results

Results for the three algorithms on five functions with two purity levels are presented in Tables 3 and 4. On two of the $\nu = 5$ runs, the SELR algorithm had numerical difficulties and did not converge on at least one of the ten data sets (as indicated by -'s in the tables). This is due to the familiar problem in logistic regression and logistic additive modelling of instability when the regions of 0's and 1's are disjoint (Hastie and Tibshirani, 1990, pp. 161–165).

Using 3 standard errors as a rough cutoff for significance, the LPP method has a significantly lower FVU on the Radial, Harmonic, and Additive functions in the "purer" case. On this case LPP also has a significantly lower percent misclassification with the Radial and Additive functions. Note that these are the two cases on which SELR had numerical problems, suggesting that they were particularly pure. On the "less pure" case, there are no significant differences between the three methods. The PPR model did tend to have lower FVU values, but the differences are not significant.

These results suggest that LPP may be of use when the underlying function is very pure, in the sense of having p's primarily at the extremities of the range. In a noisier situation, regular PPR does just fine, with much less computational effort. The main advantage of the PPR algorithm is speed. A faster version of the LPP algorithm might start with a PPR fit of M terms, and then backfit with local scoring from this starting point.

	Purer ($\nu = 5$)			Less Pure ($\nu = 1$)		
Function	PPR	LPP	SELR	PPR	LPP	SELR
Simp Int	0.083	0.079	0.093	0.471	0.514	0.544
	(0.009)	(0.007)	(0.008)	(0.047)	(0.043)	(0.055)
Radial	0.133	0.062	-	0.422	0.443	0.511
	(0.007)	(0.006)	-	(0.034)	(0.038)	(0.063)
Harmonic	0.300	0.240	0.278	0.975	1.066	1.013
	(0.016)	(0.015)	(0.018)	(0.085)	(0.096)	(0.075)
Additive	0.207	0.105	-	0.523	0.560	0.605
	(0.011)	(0.014)	-	(0.047)	(0.041)	(0.058)
Comp Int	0.277	0.257	0.362	0.687	0.722	0.810
	(0.013)	(0.012)	(0.034)	(0.037)	(0.053)	(0.080)

Table 3: FVU for Binary Response ($M = 3$, $M_{max} = 3$,$df_{ss} = 7$)

	Purer ($\nu = 5$)			Less Pure ($\nu = 1$)		
Function	PPR	LPP	SELR	PPR	LPP	SELR
Simp Int	6.4	6.6	6.9	17.0	17.9	18.6
	(0.7)	(0.4)	(0.5)	(1.2)	(1.2)	(1.0)
Radial	5.9	4.5	-	15.6	15.7	16.9
	(0.3)	(0.3)	-	(1.2)	(1.1)	(1.2)
Harmonic	10.7	10.5	10.3	26.5	26.8	26.3
	(0.8)	(0.7)	(0.8)	(1.8)	(2.1)	(1.6)
Additive	8.7	6.9	-	19.8	20.1	19.7
	(0.4)	(0.6)	-	(1.3)	(1.7)	(1.3)
Comp Int	13.5	12.3	14.9	25.3	25.8	26.5
	(0.7)	(0.8)	(0.1)	(1.6)	(1.9)	(1.6)

Table 4: % Misclass Error for Binary Response ($M = 3$, $M_{max} = 3$,$df_{ss} = 7$)

5 Concluding Remarks

A preliminary step in building the binary response models was constructing a basic linear response PPR model using smoothing splines. The smoothing spline method developed in this paper has shown itself to be competitive with supersmoother PPR and polynomial PPR. On the functions examined, both smoothing splines and polynomials tend to perform better than the supersmoother. In noiseless situations polynomials tend to dominate smoothing splines, while in higher noise situations the smoothing splines usually do as well as or better than polynomials.

For the binary response problem, the LPP model performs better than the PPR model when the true p values are near 0 and 1, and hence the regions of 0's and 1's in the predictor-space are relatively pure. In the less pure case, PPR actually seems to fit the underlying function slightly better, although the difference is not significant. In neither case does the SELR method significantly outperform the other methods, and in two of the five purer binary response examples the SELR method failed to converge on at least one of the ten data sets.

Acknowledgements

The authors thank Jerome Friedman for useful discussions regarding this material, the Statistics and Data Analysis Research Department of AT&T Bell Laboratories, and the Stanford University Department of Statistics, for support during the performance of this research.

References

[1] Chambers, J. M. and Hastie, T. J. *Eds.* (1992) *Statistical Models in S.* Pacific Grove, CA: Wadsworth & Brooks/Cole.

[2] Diaconis, P. and Shahshahani, M. (1984) On Nonlinear Functions of Linear Combinations. *SIAM J. Sci. Statist. Comput.*, **5**, 175-191.

[3] Flick, T. E., Jones, L. K., Priest, R. G. and Herman, C. (1990) Pattern Classification Using Projection Pursuit. *Pattern Rec.*, **23**, 1367-1376.

[4] Friedman, J. H. (1984a) Classification and Multiple Regression Through Projection Pursuit. Dept. of Statistics Technical Report LCS 12, Stanford University.

[5] Friedman, J. H. (1984b) A Variable Span Smoother. Dept. of Statistics Technical Report LCS 05, Stanford University.

[6] Friedman, J. H. and Stuetzle, W. (1981) Projection Pursuit Regression. *J. Amer. Statist. Ass.*, **76**, 817-823.

[7] Hastie, T. J. and Tibshirani, R. J. (1984) Generalized Additive Models. Dept. of Statistics Technical Report No. 2, Stanford University.

[8] Hastie, T. J. and Tibshirani, R. J. (1990) *Generalized Additive Models.* London: Chapman and Hall.

[9] Hastie, T., Tibshirani, R. and Buja, A. (1993) Flexible Discriminant Analysis: Adaptive Classification. AT&T Bell Laboratories Technical Report.

[10] Hwang, J-N, Lay, S-R, Maechler, M., Martin, D. and Schimert, J. (1993) Regression Modeling in Back-Propagation and Projection Pursuit Learning. *IEEE Trans. on Neural Networks.* In press.

[11] McCullagh, P. and Nelder, J.A. (1989) *Generalized Linear Models (2nd ed.).* London: Chapman Hall.

[12] Ripley, B. D. (1992) Neural Networks and Related Methods for Classification. Unpublished ms. Submitted to the Royal Statistical Society Research Section.

Figure 1: Nonlinear Functions

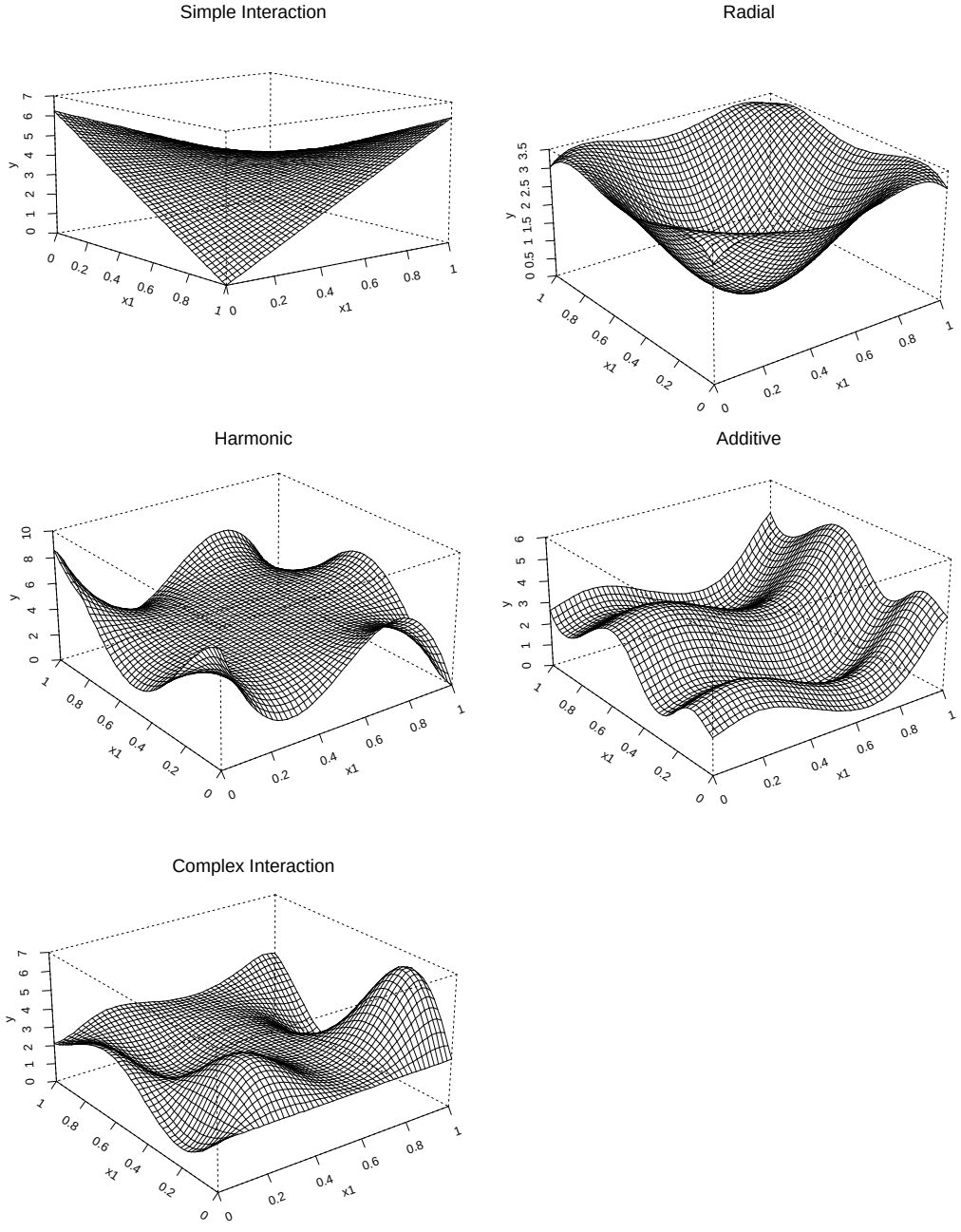

Figure 2: Contour Lines for p=0.5 with Purer Binary Data

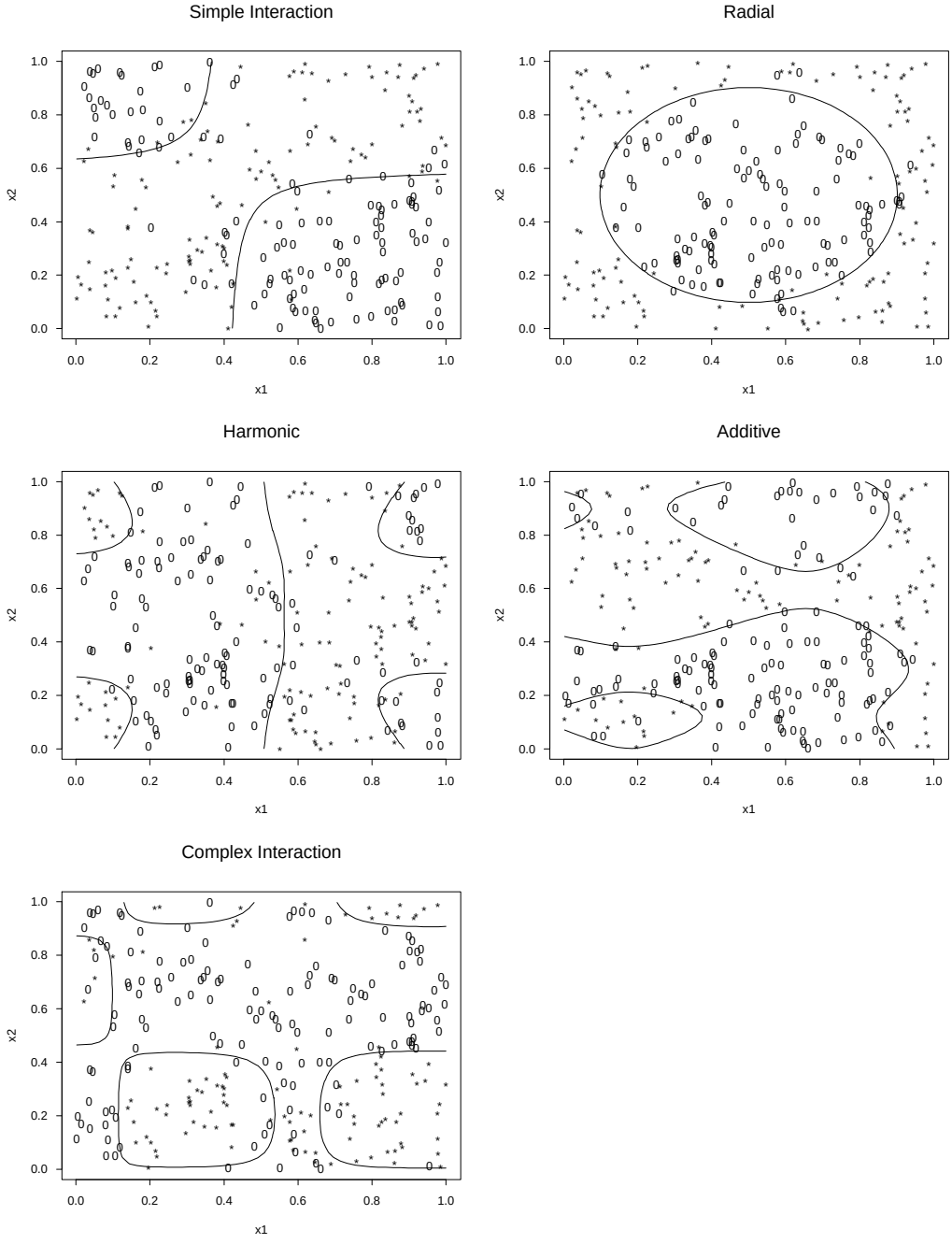

Figure 3: Contour Lines for p=0.5 with Less Pure Binary Data