Simultaneous Variable Selection and Structure Discovery in Generalized Additive Models

Mark Cus*

April 10, 2020

[DRAFT]

Abstract

We consider the problem of forecasting in settings where nonlinear relationships might necessitate a semiparametric generalized additive model. We develop a genetic algorithm (GAGAM) which is able to simultaneously select a subset of predictors to include in the regression and determine whether they should enter linearly or nonparametrically. We test GAGAM on real and artificial data, and show that it consistently outperforms the existing methods in terms of both model selection and prediction accuracy. Moreover, unlike the competing techniques, it allows for more general distributions of the response variable as well as discrete predictors. Our approach is implemented in an `R` package, making it easy to use in practice.

Keywords: Generalized Additive Models, Model Selection, Genetic Algorithm, High-dimensional Data

JEL Codes: C14, C52, C53, C61

*School of Economics, The University of Edinburgh, Edinburgh, United Kingdom. Electronic address: m.cus-babic@ed.ac.uk.

1 Introduction

Time series forecasting and cross-sectional prediction allow economists to better understand the workings of economic systems (Bai & Ng 2008), pursue more efficient monetary policy (Bernanke et al. 2004), formulate fiscal budgets (Pike & Savage 1998), and devise commercial plans (Gartner & Thomas 1993). Building such predictive models, researchers frequently rely on explanatory variables other than the target. In this framework, we often desire sparsity (only some, and perhaps few, of the considered regressors enter the model), which is motivated by model interpretability and performance. Variable selection is a key process in achieving this goal and is the subject of a large body of literature (see Desboulets 2018 for a review).

In addition, the forecaster must determine the model's functional form. Although linear models are popular for their simplicity, evidence suggests nonlinear relationships may be important in some economic structures (Bachmeier et al. 2007). To model such systems, one can employ a semiparametric generalized additive model (GAM; Hastie & Tibshirani 1990)[1]:

$$g(\mathbb{E}(y_i)) = \alpha + \sum_{j \in \mathcal{L}} \beta_j x_{ij} + \sum_{k \in \mathcal{N}} f_k(x_{ik}) + \varepsilon_i \tag{1}$$

where the variables in set $\mathcal{L}$ affect the dependent variable linearly whereas those in $\mathcal{N}$ enter nonparametrically and their (possibly nonlinear) functional form is estimated. GAMs are useful because their additivity alleviates the curse of dimensionality plaguing fully nonparametric regressions (Wood 2017). Moreover, they can differentiate between linear and nonlinear predictors. This is valuable because nonparametric estimation of truly linear effects reduces efficiency and risks overfitting while ignoring nonlinearities leads to bias.

However, model building now requires both feature selection *and* structure discovery. That is, we need to establish which variables to include and how to allocate them between $\mathcal{L}$ and $\mathcal{N}$. This is the problem addressed in the present study.

The existing literature (most notably Chouldechova & Hastie 2015, Lou et al. 2016) approaches this question using the group version of the lasso (Tibshirani 1996, Yuan & Lin 2006) which penalizes the magnitude of the model coefficients. Variable selection and, in a way that will become clear later, structure discovery are facilitated by some parameter estimates being deflated to exactly zero. However, with certain predictor correlation structures the current techniques are not model selection consistent, meaning that they do not extract the true model with probability tending to one as $n \to \infty$. This is because the lasso compensates for the bias induced by overshrinking truly nonzero coefficients by recruiting into the model the correlated noise variables (Zhao & Yu 2006). Therefore, we generate models that are too large and estimate too many effects nonparametrically.

The most immediate remedy is to penalize only the number of nonzero coefficients but not their magnitudes, e.g. by minimizing the Bayesian Information Criterion (BIC; Schwarz 1978). However, this leads to an NP-hard combinatorial problem (Natarajan 1995) which is computationally infeasible with even modestly dimensional data sets. Namely, we would normally have to fit every possible model.

[1] $g(\cdot)$ is a known link function which allows y to have a general distribution, e.g. if y is binomially distributed, $g(\cdot)$ could be the logistic curve. However, for the rest of the paper we focus on the Gaussian response, and $g(\cdot)$ is simply the identity function.

To avoid such exhaustive search and still perform feature filtering using the BIC in fully parametric models, Acosta-González & Fernández-Rodríguez (2007), Paterlini & Minerva (2010), Savin (2012), and others have utilized Holland's (1975) genetic algorithm (GA). GA is an evolution-inspired optimization algorithm well-suited for the pursuit of the global optimum in a highly irregular search space. The contribution of this paper is to develop the first[2] GA able to carry out simultaneous variable selection and structure discovery in semiparametric GAMs. We designate the acronym GAGAM (Genetic Algorithm for Generalized Additive Models) for this approach.

Having formulated the algorithm, our method is tested on a number of synthetic and two economic data sets, and its performance is compared to its most viable competitor, Chouldechova & Hastie's (2015) Generalized Additive Model Selection (GAMSEL) estimator. We demonstrate that GAGAM performs very well in model selection, finding the true model almost every time in certain settings, and produces low out-of-sample prediction error. It also scales to high-dimensional and $p \gg n$ problems. Relative to GAMSEL, GAGAM is always competitive and often noticeably better at excluding the noise variables. In addition, it allows a wider range of distributions of the dependent variable, non-continuous predictions, and greater control over the estimation of nonparametric terms. These benefits come at the cost of computational time, though the latter is not prohibitive.

The rest of this paper is organized as follows. Sections 2 and 3 review the literature. Section 2 provides an overview of the existing approaches based on penalized regression. Section 3 introduces genetic algorithms and reviews the existing applications in econometric model selection. Section 4 presents our algorithm while Section 5 tests it with simulated and real data. Section 6 describes our implementation of GAGAM in an `R` package. Section 7 concludes. Appendix A attempts to extend GAGAM to models with hierarchical interaction terms.

2 Background: Penalized Regression

We first introduce the concept of penalized regression and explain how nonparametric models are estimated using splines. This informs the discussion of the existing methods for model selection in Section 2.3.

2.1 Penalized Regression

Suppose we wish to fit an entirely linear model (Equation 1 with $\mathcal{N} = \emptyset$). The idea of penalized regression is to find the parameter estimates by minimizing a criterion such as:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + J(\boldsymbol{\beta}) \tag{2}$$

[2]During the writing of this article it has come to my attention that a method similar to ours may be implemented in the `cgam` `R` package (Meyer & Liao 2019). However, the information about this approach is very scarce, and, to the best of my knowledge, no description of it, beyond the brief practical instructions in the package manual, has ever been published.

where the first term is the sum of squared residuals and the second term is some penalty which is a function of the coefficients. A common choice for the penalty is the L_q-norm:

$$J(\boldsymbol{\beta}) = \lambda \|\boldsymbol{\beta}\|_q^q \tag{3}$$

where $\|\boldsymbol{\beta}\|_q \equiv (|\beta_1|^q + |\beta_2|^q + \cdots + |\beta_p|^q)^{\frac{1}{q}}$ is the L_q-norm of vector $\boldsymbol{\beta}$ and λ is a hyperparameter which controls the degree of penalization. The aim of penalization is to constrain the coefficients in a way that is advantageous to our application. For instance, $q = 2$ corresponds to ridge regression (Hoerl & Kennard 1970) which shrinks the magnitude of the coefficients to reduce the variance of the parameter estimates by discouraging them from taking extreme values. Penalties with $q \leq 1$ are able reduce some coefficients to exactly zero and thus perform variable selection. In fact, $q = 1$ is the well-known lasso estimator developed by Tibshirani (1996). Finally, the "quasi-norm" L_0 has the interpretation of counting the number of nonzero coefficients without regard for their magnitudes. As a special case, minimizing the information criteria such as the Akaike Information Criterion (AIC; Akaike 1974) and the BIC can be understood as minimizing Equation 1 with the L_0-norm penalty and a particular choice of λ (Louizos et al. 2017).

2.2 Estimation of Nonparametric Models Using Splines

An elementary grasp of the fitting process for nonparametric models is critical to appreciate approaches such as GAMSEL. As the simplest case, consider the model $y_i = f(x_i) + \varepsilon_i$ where the functional form of f is unknown. A straightforward way to estimate f, while allowing it to be nonlinear, is to divide the range of x into distinct regions, fit some relationship in each, and then define $\hat{f}$ piece-wise. The border values of these regions of x are called the knots. Figure 1 shows one such example where we fit a linear relationship in each of the four chosen regions of the scatter plot.

[Figure 1 about here]

The question now becomes *how* to estimate the pieces of f, e.g. the discontinuities in $\hat{f}$ in Figure 1 discredit the use of separate linear regressions in each region. This can be addressed by recognizing our a dual objective – the estimated function should fit close to the data but also be smooth to avoid overfitting. Thus, finding $\hat{f}$ can be written as a penalized regression where we reward closeness to the observations but penalize curvature, with λ governing the trade-off between the two (Hastie et al. 2009):

$$\hat{f} = \arg\min_f \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(x)^2 \mathrm{d}x \tag{4}$$

It can be shown that the unique minimizer of Equation 4 is a natural cubic spline (Hastie et al. 2009). The latter is constructed by placing the knots on all unique values of x, fitting a cubic in each region, and then joining them together in a way that $\hat{f}$ is continuous to the second derivative[3] (Hastie et al. 2009, Wood 2017).

[3]Formally, the second derivative should also be zero outside the observed range of x.

To implement this, $f(x)$ is represented using a linear basis expansion:

$$f(x) = \sum_{j=1}^{k} b_j(x)\beta_j \tag{5}$$

where $b_j(\cdot)$ are the known basis functions which together form basis b, and k is the basis dimension (in our case equal to the number of interior knots plus two). Choosing the basis appropriately ensures that $\hat{f}$ is a natural cubic spline. *One* possible basis[4] is the truncated power basis:

$$\begin{gathered} b_1(x) = 1 \quad b_2(x) = x \quad b_{j+2}(x) = d_j(x) - d_{k-1} \\ d_j(x) = \frac{(x-\xi_j)_+^3 - (x-\xi_k)^3}{\xi_k - \xi_j} \end{gathered} \tag{6}$$

where ξ_j is the value of x at knot j, and $(\cdot)_+ \equiv \max(0, \cdot)$. Having represented $f(x)$ using this basis expansion we form the design matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} b_1(x_1) & b_2(x_1) & \cdots & b_k(x_1) \\ b_1(x_2) & b_2(x_2) & \cdots & b_k(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_n) & b_2(x_n) & \cdots & b_k(x_n) \end{bmatrix}$$

This allows us to rewrite Equation 4 in a way that is feasible for estimation:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^k}{\arg\min} \|\mathbf{y} - \mathbf{B}\boldsymbol{\beta}\|_2^2 + \lambda \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\Omega} \boldsymbol{\beta} \tag{7}$$

where $\{\boldsymbol{\Omega}\}_{ij} = \int b_i''(x) b_j''(x) \mathrm{d}x$ (given the choice of basis in Equation 6 we can find the explicit expressions for the elements of $\boldsymbol{\Omega}$). Equation 7 is a well-known case of generalized ridge regression with the solution:

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^{\mathrm{T}}\mathbf{B} + \lambda\boldsymbol{\Omega})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{y}$$

Thus, the fitted spline is given by $\hat{f} = \mathbf{B}\hat{\boldsymbol{\beta}}$.

Many details about fitting Equation 1 are still missing, for instance, how to approach additive and parametric terms and how to choose λ. Similarly, in practice, we usually resort to a small number of well-dispersed knots. However, the above should provide a sufficient first step to understanding GAMSEL and related methods. For clarity, Figure 2 demonstrates fitting a spline to the data from Figure 1. Since we have three interior knots, panels (a) to (e) plot the five basis functions given in Equation 6. Panel (f) shows the resulting spline.

[Figure 2 about here]

[4]There are different bases that achieve the same result of fitting a natural cubic spline.

2.3 Previous Work

There is a substantial body of work applying the concepts of penalized regression beyond the linear world, and many authors have addressed aspects of variable selection and structure discovery in semiparametric additive models. For instance, Ravikumar et al. (2009) and Huang et al. (2010) apply group lasso (Yuan & Lin 2006) to select features in additive models with only nonparametric terms. The techniques of Liu et al. (2011) and Müller & van de Geer (2015) allow for both types of terms but only perform selection on the linear ones while assuming a fixed set of nonlinear variables. The methods developed in Lian & Liang (2013) and Yang et al. (2017) extend selection to nonparametric terms but still cannot carry out structure discovery. On the other hand, Huang, Wei & Ma (2012) can determine whether a variable should enter linearly or nonparametrically but do not exclude variables.

Within this literature, Zhang et al. (2011) and Lian et al. (2012) were the first to accomplish *simultaneous* variable selection and structure discovery. The former uses a reproducing kernel Hilbert space norm penalty while the latter employs a smoothly clipped absolute deviation penalty. However, these methods do not scale to high dimensions.

To address this, Chouldechova & Hastie (2015) and Lou et al. (2016) developed alternative solutions – Generalized Additive Model Selection (GAMSEL) and Sparse Partially Linear Additive Models (SPLAM), respectively. Since GAMSEL is the main benchmark for GAGAM, and to understand its limitations, we now describe it in greater detail.

Suppose our goal is to fit the model:

$$y_i = \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i \tag{8}$$

To make estimation of Equation 8 feasible we first transform it. Begin by shifting every f_j vertically so that their means are zero and add a common intercept, α_0. Letting g_j denote the shifted functions, we get:

$$y_i = \alpha_0 + \sum_{j=1}^{p} g_j(x_{ij}) + \varepsilon_i \tag{9}$$

This was to ensure identifiability since otherwise we could subtract a constant from f_j, add it to f_k, and the resulting prediction would remain the same. Second, separate each g_j into a linear and a nonparametric component, $g_j(x) = \alpha_j x + h_j(x)$:

$$y_i = \alpha_0 + \sum_{j=1}^{p} \alpha_j x_{ij} + \sum_{j=1}^{p} h_j(x_{ij}) + \varepsilon_i \tag{10}$$

Now form the basis expansion matrix ($\mathbf{B}$ in Section 2.2) and the corresponding penalty matrix ($\mathbf{\Omega}$) for every variable in the data set, $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(p)}$ and $\mathbf{\Omega}^{(1)}, \mathbf{\Omega}^{(2)}, \ldots, \mathbf{\Omega}^{(p)}$. However, rather than using the truncated power basis, GAMSEL uses the Demmler-Reinsch basis (Demmler & Reinsch 1975). The exact definition of this basis is mathematically involved but its key characteristics are as follows. The first two columns of $\mathbf{B}^{(j)}$, $b_1(x)$ and $b_2(x)$, correspond to the constant and linear basis functions, respectively, and thereafter the

columns of $\mathbf{B}^{(j)}$ are increasing in the order of complexity. Figure 3 illustrates the first few columns of $\mathbf{B}^{(j)}$. Second, the columns of $\mathbf{B}^{(j)}$ are orthonormal. Third, the penalty matrix $\mathbf{\Omega}^{(j)}$ is diagonal with entries $\text{diag}(0, d_2, d_3, \ldots, d_k)$ where $d_2 < d_3 < \cdots \leq d_k$ so that penalization is increasing in complexity.

[Figure 3 about here]

Since the h_j are the vertically shifted versions of the f_j with zero mean they are represented by $\tilde{\mathbf{B}}^{(j)}_{2:k}\boldsymbol{\beta}_j$. $\tilde{\mathbf{B}}^{(j)}_{2:k}$ is matrix $\mathbf{B}^{(j)}$ with the first column (corresponding to the constant basis function) removed and the other columns demeaned. $\boldsymbol{\beta}_j \in \mathbb{R}^{k-1}$ is a vector of coefficients on the remaining basis functions. Substituting the linear basis expansion for each h_j into Equation 10 yields:

$$y_i = \alpha_0 + \sum_{j=1}^{p} \alpha_j x_{ij} + \sum_{j=1}^{p} \tilde{\mathbf{B}}^{(j)}_{2:k}\boldsymbol{\beta}_j + \varepsilon_i \tag{11}$$

GAMSEL estimates Equation 11 by minimizing the criterion:

$$\begin{aligned} \underset{\alpha_0,\{\alpha_j\},\{\boldsymbol{\beta}_j\}}{\arg\min} \frac{1}{2} \left\| \mathbf{y} - \alpha_0 - \sum_{j=1}^{p} \alpha_j x_j - \sum_{j=1}^{p} \tilde{\mathbf{B}}^{(j)}_{2:k}\boldsymbol{\beta}_j \right\|_2^2 + \\ \lambda \left(\gamma|\alpha_j| + (1-\gamma)\sqrt{\boldsymbol{\beta}_j^{\mathrm{T}} \mathbf{\Omega}^{(j)}_{2:k} \boldsymbol{\beta}_j} \right) + \frac{1}{2} \sum_{j=1}^{p} \psi_j \boldsymbol{\beta}_j^{\mathrm{T}} \mathbf{\Omega}^{(j)}_{2:k} \boldsymbol{\beta}_j \end{aligned} \tag{12}$$

The first term is the sum of squared residuals while the third, through the choice of ψ_j, allows the modeler to control the maximum complexity of each g_j. The second term (selection penalty) is what allows GAMSEL to perform variable selection and structure discovery. Namely, $\sqrt{\boldsymbol{\beta}_j^{\mathrm{T}} \mathbf{\Omega}^{(j)}_{2:k} \boldsymbol{\beta}_j}$ is the group lasso penalty which sets either all or none of the elements of $\boldsymbol{\beta}_j$ to zero while $|\alpha_j|$ is the standard lasso penalty which shrinks α_j. If both of these penalties are strong enough optimization might result in $\hat{\alpha}_j = 0$ and $\hat{\boldsymbol{\beta}}_j = \mathbf{0}$. This implies $\hat{g}_j(x_j) = 0$ and x_j is excluded from the model. On the other hand, minimization of the GAMSEL criterion may find $\hat{\alpha}_j \neq 0$ and $\hat{\boldsymbol{\beta}}_j = \mathbf{0}$ in which case $\hat{g}_j(x_j) = \hat{\alpha}_j x_j$ and x_j enters the model linearly. Finally, if $\hat{\boldsymbol{\beta}}_j \neq \mathbf{0}$ then $\hat{g}_j(x_j)$ is a cubic spline and thus x_j is allowed to have a nonlinear effect on y. λ controls the degree of penalization, and γ allows one to favor linear or nonparametric terms.

SPLAM is conceptually very similar to GAMSEL. The main differences are that SPLAM uses the truncated power basis instead of the Demmler-Reinsch basis and it formulates the penalty slightly differently.

However, while these lasso-based approaches are useful and elegant, they do have certain drawbacks. Zhao & Yu (2006) show the lasso need not be model selection consistent, meaning that it will not select the true model with probability tending to one as $n \to \infty$. Whether it is consistent depends on the Irrepresentable Condition which states that none of the truly nonzero predictors can be "represented" by the noise variables. Zhao & Yu (2006) provide a formal definition but loosely speaking this can be interpreted as prohibiting high correlation between the relevant and irrelevant covariates.

To understand this intuitively, examine Equation 3 – both the truly zero and nonzero coefficients are penalized at the same rate, λ. This implies that a positive λ needed to shrink the noise variable coefficients to zero induces a downward (in absolute value terms) bias to our estimates of the truly nonzero coefficients. If (some) irrelevant predictors are highly correlated with the relevant ones, the lasso recruits them into the model to compensate for this bias. This implies that it chooses a model larger than the true one.

To make matters worse, the Irrepresentable Condition is difficult to satisfy in practice, especially in high-dimensional settings (Zhang 2010). Also, since group lasso, which GAMSEL and SPLAM are instances of, also uses a constant rate of penalization it is expected to behave in the same way (Huang, Breheny & Ma 2012, Breheny & Huang 2015).

Moreover, even when the Irrepresentable Condition holds, e.g. orthogonal design, Leng et al. (2006) and Meinshausen & Bühlmann (2006) show that choosing λ to minimize prediction error also leads to inconsistent model selection. The inconsistency is again manifested in the inclusion of noise variables. This is worrying because in applications λ is in fact usually found by optimizing the cross-validated mistakes in prediction.

It is these two disadvantages of the lasso that most directly motivate GAGAM. The aim of GAGAM is to minimize the BIC. As mentioned earlier, this can be interpreted as L_0-norm penalization which either sets the coefficient equal to zero or leaves it unshrunk. This allows us to avoid the issues raised by Zhao & Yu (2006). Similarly, the rate of penalization is determined a priori. However, the nature of the L_0 penalty precludes the use of standard optimization techniques.

3 Background: Genetic Algorithms

This section first introduces the concept of genetic algorithms and then reviews their previous applications in econometric model selection.

3.1 Overview of Genetic Algorithms

Genetic algorithms are a class of optimization algorithms first introduced by Holland (1975). They work by taking a number of possible solutions to a problem, called the *population* of solutions, and evolving them in a way that resembles evolution in nature. The aim is to arrive at a population with some high quality solutions just as natural evolution leads to organisms better adapted to their ecosystems.

Before explaining how GAs work, we may note that these algorithms are best suited for problems with characteristics similar to those that evolution faces in nature (Mitchell 1996). First, the number of possible solutions should be large, making exhaustive search is infeasible. Second, the space of possible solutions should not be smooth and unimodal, or it should not be well understood (other algorithms are more efficient in solving problems like locating the bottom of a single smooth valley). Third, we do not require *the* global optimum – a close approximation will suffice.

To illustrate how GAs solve such problems, consider finding the maximum of some real-valued function, $g(x)$, where the input is limited to integers between 0 and 31, $x \in \mathbb{Z} : x \in [0, 31]$. Each of the 32 values of x represents one candidate solution to the problem. The

function $g(x)$ assigns a fitness to each solution, that is, it determines its ability to solve the problem, i.e. maximize $g(x)$. For instance, if $g(16) > g(15)$, then the solution $x = 16$ is better than $x = 15$. For this reason, $g(x)$ is called the fitness function.

Next, we encode solutions in some way that is useful to the algorithm. Most often this is binary coding, meaning that every x is written as a string of ones and zeros. In our example, we can simply use the base 2 number system of length five, e.g. $x = 15$ is written as the string 01111 and $x = 16$ is 10000.

Having specified the fitness function and the coding, Mitchell (1996) outlines a basic GA:

1. Randomly generate an initial population of N candidate solutions.
2. Determine the fitness, $g(x)$, of every solution.
3. Repeat the following until N new solutions have been created.
   (a) Select two solutions from the existing population in a way that better solutions are more likely to be chosen [Selection].
   (b) Recombine the features of the chosen solutions to create two new solutions [Crossover].
   (c) Randomly change some of the features of each of the newly generated solutions [Mutation].
4. Replace the existing population with the solutions just created.
5. Return to step 2.

Each iteration of the algorithm is called a generation. The number of generations before termination depends on the application, and could vary from 30 to 1000 or more. There is no single stopping criterion as it could be essentially arbitrary, e.g. 100 generations, or not, e.g. stop when the best ten solutions in the population are identical. The complete set of iterations is called a run of the algorithm. At the end of a run, the algorithm might return the best solution in the final generation's population.

The process in step 3a is called selection. Since better solutions are more likely to contribute to the following generation, this operator ensures that the fitness of solutions in the population gradually improves. In other words, selection pulls the population of solutions toward the optima of $g(x)$.

[Figure 4 about here]

In step 3b we perform crossover. This operator somehow merges the characteristics of two solutions. This is beneficial because our solutions consist of building blocks, e.g. each bit in the string 01111 is a building block for the solution $x = 15$. Thus, if we have two solutions, each with some "good" and some "bad" building blocks, the hope is that crossover will take the "good" blocks from both solutions and combine them together. Figure 4 shows an example of a simple form of crossover, uniform crossover.

[Figure 5 about here]

The process in step 3c is called mutation, and it randomly alters some of the building blocks of the solution. The reason for this is twofold. First, we might arrive at a better solution by chance. Second, we wish to prevent convergence to a local optimum. Figure 5 illustrates a simple form of mutation, uniform mutation.

In summary, the role of selection is to *exploit* good solutions when they arise while crossover and mutation *explore* the space of possible solutions. In addition, notice the

element of randomness in crossover and mutation, e.g. mutation of each bit happens with some probability. For this reason, GAs are stochastic algorithms which may return a different result every time.

3.2 Previous Work

GAs have been used extensively in a wide variety of applications, from predicting protein structure (e.g. Schulze-Kremer 2003) to natural language processing (e.g. Alba et al. 2006). Axelrod & Davis (1987) implement a GA to determine the evolutionary optimal strategy in repeated games with a prisoner's dilemma structure. Closer to the present work, GAs have been fruitfully applied to variable selection in multiple linear regression in econometrics (Acosta-González & Fernández-Rodríguez 2007, Paterlini & Minerva 2010, Savin 2012) and chemometrics (Leardi et al. 1992, Broadhurst et al. 1997).

In fact, feature selection is a good example of the kind of problems GAs are designed to solve. First, the number of models that can be constructed from as few as thirty predictors precludes exhaustive search. Second, traditional model selection is discrete, with the fitness function (e.g. R^2) moving discontinuously as we add or remove variables. In addition, there are often many locally optimal models, leading to poor performance of greedy algorithms. Third, a model nearly as good as the globally optimal one is usually acceptable. Finally, the individual variables can be interpreted as building blocks for regression models so crossover and mutation make intuitive sense.

To use GAs in variable selection, the studies above had to adapt the coding and choose a fitness function. All of the mentioned authors use binary coding. This means the considered predictors are first ordered, and then a model is represented as a binary string where the bit in the same position as the variable determines whether that feature is included (1) or not (0). Figure 6 illustrates this coding.

[Figure 6 about here]

For the fitness function, the popular choices are estimators of out-of-sample predictive performance such as the AIC and the BIC (used in Acosta-González & Fernández-Rodríguez 2007, Paterlini & Minerva 2010, Savin 2012) and direct measures of prediction success such as the root mean squared error (RMSE; used in Leardi et al. 1992, Broadhurst et al. 1997). With the exception of Paterlini & Minerva (2010), the authors who use the information criteria estimate each model on the entire sample and then compute the value of the criterion. On the other hand, Leardi et al. (1992) and Broadhurst et al. (1997) split their sample into training and validation subsamples, estimate the model using the training set, produce predictions for the validation data points, and compute the RMSE as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

It is a priori unclear which fitness function is more appropriate. The BIC is model selection consistent (Sin & White 1996) but its minimum value need not correspond to the true model in any finite sample. On the other hand, the RMSE can lead to overfitting the

validation data set (Ng 1997). Specifically, if many models are tested on the same validation set, some noise variables may by chance reduce the prediction error in this set.

None of the reviewed papers compare the two fitness functions but the more recent work appears to favor the information criteria. Other implementation details, e.g. the method of selection, also vary in the literature but no approach has emerged as clearly superior.

The results of the studies using GAs for variable selection are encouraging. For instance, Savin (2012) uses a simulated data set with 50 features, 5 relevant ones and 45 noise variables. He finds that, on average, his GA includes between 79% and 88% of the truly nonzero regressors while introducing less than 1% of the irrelevant predictors. Similar, and sometimes even better, results are found when the true model contains 25 nonzero coefficients. Savin also varies the levels of collinearity and noise and reports that the GA is largely unaffected by the former but it is hampered by noise. Other studies such as Acosta-González & Fernández-Rodríguez (2007) find comparably promising results.

Relative to other methods like stepwise regression, Acosta-González & Fernández-Rodríguez (2007) and Paterlini & Minerva (2010) establish that a GA produces sparser models with fewer noise variables. In addition, the resulting models have lower values of the information criteria, though not necessarily also lower out-of-sample RMSE. Compared to the lasso, Savin (2012) finds that the GA is less sensitive to noise and regressor correlation, and it outperforms with large true models. On the other hand, his GA underperforms the lasso in small samples.

3.2.1 Extension to Nonlinear Effects

Various authors have also considered how to apply the GA when some explanatory variables are possibly nonlinearly related to the target. The literature offers two such extensions[5]. The first is presented by Dudek (2012) who applies a GA like the ones above to select the variables for a fully nonparametric regression of the form:

$$y_i = f(\mathbf{X}_i) + \varepsilon_i$$

where $\mathbf{X} \in \mathbb{R}^{p \times n}$ includes all predictors. To estimate $f(\cdot)$, Dudek uses the multivariate version of the Nadaraya-Watson kernel estimator (Nadaraya 1964, Watson 1964). The author applies this to a forecasting model for short-term electricity consumption and finds that, compared to using all available predictors, variable selection using his GA is able to reduce the forecast error.

However, this approach has two shortcomings. First, if some relationships are truly linear, it is wasteful to estimate them nonparametrically. Second, such regressions suffer from the curse of dimensionality (Li & Racine 2007). That is, because we estimate not only the effect of each individual regressor but also *any* possible interaction structure, the sample required to maintain a constant prediction error grows rapidly with the number of predictors. Thus, this method is limited to low-dimensional settings.

The other extension was proposed Hasheminia & Akhavan Niaki (2006) and Paterlini & Minerva (2010). Rather than use nonparametric regression, these studies exploit variable

[5]Another approach, found in the 1990s economics literature, is based on an algorithm similar to GA, called genetic programming (see Schmertmann 1996, Szpiro 1997, Beenstock & Szpiro 2002).

transformations. This means that a predictor can be included in the model untransformed, and have a linear effect, or, for instance, as a log or a polynomial transformation. Thus, we avoid the curse of dimensionality and allow the truly linear variables to enter as such.

However, this too has some disadvantages. For instance, the modeler may overlook some of the required transformations. Moreover, the GA is now tasked with not only feature selection but also finding the exact form of nonlinearity for every variable. This enlarges the space of possible solutions and may reduce the algorithm's performance.

3.2.2 Extension to High-dimensional Problems

Returning now to fully linear settings, some authors have reconsidered the use of binary coding, particularly for high-dimensional problems. Gan & Learmonth (2016) report that the efficiency of a GA (time required to find an equally good solution) is severely diminished when it operates on a very long string of 1s and 0s. Moreover, if $p > n$, binary notation can result in overdetermined models. For example, with 7 observations and 10 predictors, nothing is stopping the GA operators from producing, say, a model with 9 explanatory variables such as 1111111110. Finally, binary notation does not allow the researcher any control over the maximum model size.

As an alternative to binary coding, Jeong et al. (2015), Gan & Learmonth (2016), and Sousa et al. (2017) use integer coding. An example is shown in Figure 7. Each solution is written as a string of integers. The positive numbers correspond to the indices of the variables included in the model while zeros act as placeholders. This means they do not currently represent any variable but they are still subject to crossover and mutation so a zero could mutate to a variable. The operators need to be adapted to work with this coding.

[Figure 7 about here]

Gan & Learmonth (2016) show that integer coding is much more efficient than binary coding for high-dimensional variable selection. Furthermore, by choosing the length of the string, the modeler specifies the maximum model size, thus avoiding models with more variables than observations.

4 Methodology

This section presents our genetic algorithm. Compared to the GAs examined above, we modify the fitness function and the coding, and develop new crossover and mutation operators that are compatible with the chosen coding (Section 4.1). With this in hand, Section 4.2 outlines the GAGAM algorithm.

4.1 Fitness Function, Coding, and Operators

To compute the fitness of a model we first fit it on the entire sample using conventional means. We use cubic splines with the cardinal basis of dimension $k = 10$. The smoothing parameter (λ) is chosen based on restricted maximum likelihood. The details about this estimation process are in Wood (2017). However, fitting the models differently would not

change anything fundamental about GAGAM.

Once this is completed, we judge the model's quality by its predictive performance. To estimate it, we begin with the AIC derived by Wood et al. (2016). This version of the AIC is suitable for use with GAMs as it penalizes not only the number of parametric terms but also the effective degrees of freedom associated with the nonparametric terms. The criterion is given by:

$$\text{AIC} = -2l(\hat{\beta}) + 2\tau_2 \tag{13}$$

Conceptually, the first term rewards close fit to the data while the second term penalizes degrees of freedom. Wood et al. (2016) provide a formal definition.

This AIC yields an estimate of the model's out-of-sample prediction capabilities, and we could use it as our fitness function. In exploratory analysis, we experimented with this approach as well as with splitting the sample and using RMSE (like Broadhurst et al. 1997). However, both methods proved unsatisfactory as GAGAM included many noise variables. To correct this, we replace the factor of 2 in the second term of the AIC with $\ln(n)$ so model complexity is penalized more heavily:

$$\text{BIC} = -2l(\hat{\beta}) + \ln(n)\tau_2 \tag{14}$$

We admit that the theoretical consequences of this substitution are unclear, and it would be premature to claim that the new BIC is model selection consistent. We use it as our fitness function simply because it outperforms Wood et al.'s AIC.

[Figure 8 about here]

To encode the models for use in GAGAM we utilize a mix of integer and binary coding. Figure 8 presents an example. In the top row, integer coding is used to represent the variables. Positive numbers correspond to the indices of the included predictors while zeros are simply placeholders and do not represent any variable. In the bottom row, we use binary coding. The value in each bit determines whether the variable directly above it enters the model linearly (0) or nonparametrically (1). The column-wise length of the string, denoted K_{Var}, determines the maximum number of included variables.

With this novel coding we require new crossover and mutation operators. Because the order of variables has no bearing on the solution's quality we develop a version of uniform crossover (Figure 4 shows the original binary coding variant). The new crossover operator, an example of which is shown in Figure 9, proceeds in two steps.

First, consider the variables shared by both models, whether or not they have the same form. For each separately, child model 1 gets the variable-form pair from parent model 1 with probability 0.5 and the pair from parent 2 otherwise. For example, for variable 2 in Figure 9 we have the pair 2-0 in parent 1 and the pair 2-1 in parent 2. In this realization child 1 received the latter pair (2-1). Child model 2 receives the pairs not allocated to child 1. For the variable-form pairs that are identical in both parents (share the same form), this step simply means putting them in both child models. Once this process is completed, delete the common variables from the parent models.

The second step tackles the remaining variables and zeros. We start in the first position of the parents and move from left to right. In each position, take the variable-form pair from

parent 1 and place it into child 1 with probability 0.5. Otherwise, this child gets the pair from parent 2. Again, child 2 receives the left over pairs.

After crossover is completed, the order of the variable-form pairs in each child is randomized (not shown in Figure 9). This is because the newly generated models will again undergo crossover in the next generation, and the non-random order of variables just created (one can see in Figure 9 that all common variables are on the left) could skew the second step in this process.

[Figure 9 about here]

Compared to the binary coding crossover, our operator retains the feature that variables present in both of the original models must also be in both child models. Similarly, if a variable was in only one of the parent models it has to be in one and only one of the newly created models. No variables are getting lost or created. The main difference from before is that receiving a variable from the other model (step 2) can now mean losing one of your own, e.g. 3-0 in P1′ gets *replaced* with 5-1 from P2′.

[Figure 10 about here]

In addition to adapting crossover, we also develop a new mutation operator. An example of how it works is displayed in Figure 10. We again proceed in two steps.

First, each bit in the top row of the original model is chosen for mutation with probability p_m. If a bit is selected, there are two possibilities. If it contains a variable then it gets turned to zero. If it is a zero we replace it with a regressor randomly chosen from the set of variables not in the original model. The replacement variable is then removed from this set so that two zeros cannot get mutated to the same predictor. Every newly included variable is equally likely to be linear or nonparametric.

Second, we consider the forms of the variables not chosen for mutation in the first step. Each of those bits gets flipped (from 0 to 1 or vice versa) with probability p_{mnp}.

4.2 Algorithm for GAGAM

With the fitness function, the coding, and the operators appropriately redefined, we now present the algorithm for GAGAM. The pseudocode for it is given in Algorithm 1.

[Algorithm 1 about here]

We begin by generating 500, the chosen population size, models completely randomly – the number of nonzero predictors, the variables included, and their forms are determined by chance. GAGAM will evolve this population over a pre-determined number of generations, equal to *no_gen*.

Each generation begins by estimating the models in the current population, *pop*, calculating their BIC values, and sorting them in ascending order based on the BIC.

Next, we copy the best ten models directly into the next generation's population (a.k.a. new population, *new_pop*). This is called elitism, and it ensures that the best solution in the population never worsens over time.

We also make two more copies of the top ten models. In the first copy the models are randomly paired up, crossed over, and the crosses are placed into the new population. The pairing up process works without replacement so that every model is found in one and only one pair. The motivation for this step is that two of the better performing solutions may be near the top for different reasons, for example, because they each contain one variable with high predictive power. The hope is that crossover will sometimes place both of those variables in one of the children, thus generating an even better model.

In the second copy we mutate the nonparametric row of every model, leaving the variables included unchanged. This is done by flipping each 0 or 1 to its opposite with probability p_{mnp}. The idea is that some highly predictive variables might be included in the wrong form. Applying mutation to just the functional form row can remedy this without the risk of eliminating those variables like the full mutation operator. The mutants are again placed into *new_pop*.

Thus far the next generation's population contains 30 models, all either the best solutions of the current generation or constructed from them. This part of the algorithm allows us to exploit good solutions as they arise.

Next, we randomly select 190 models from the set of the top 240 in *pop*. Copies of the chosen solutions are then paired up and crossed over. The pairing up and crossover is also repeated with copies of the best 40 models in *pop*, giving us in total 190 + 40 = 230 crosses.

Each of these crosses is then mutated. We also mutate unaltered copies of the top 240 models from *pop*, giving us in total 470 mutants.

The crossover and mutation steps explore the space of all possible models to find new, better ones. Also, notice that crossover is not applied as widely as mutation. This is because while crossover can be helpful, it may be severely disruptive when the two models contain few common variables. Crossing over only about a half of the solutions therefore realizes the benefits without forcing us to bear the costs too broadly.

Finally, we add the mutants to the 30 solutions already in *new_pop*. This yields a set of 500 models which replace the existing population (*pop*), and a new generation is ready to begin. As alluded to earlier, this process is repeated for a fixed number of generations.

At the end of a run, GAGAM returns the model with the lowest BIC value in the final population. In fact, because of elitism this is the lowest BIC the algorithm witnessed over the entire run. It is our hope that this model is a close approximation to *the* globally optimal solution.

Note that selection, which pulls our population towards the optima, is implicitly operating throughout the algorithm. For instance, the best ten models contribute to the new population six times (lines 5,7,8,10,13,15) while the worst 260 specifications are completely discarded.

However, the minimum of the BIC need not correspond to the true model in any finite sample. Our simulations revealed that often some noise variables reduce the BIC (and thus get included) but they do so by a very small margin. To remove such variables and in general produce more conservative models we implement three "model reduction" procedures which are applied at the end of a GAGAM run.

The first, R1, loops through the variables included in the model returned by GAGAM, removing one at a time and recording the change in the BIC. Then, we remove all variables which result in the BIC increasing by less than 7. This is based on the rule-of-thumb that only differences in BIC larger than 6 provide strong evidence for the model with the lower

value of the criterion (Kass & Raftery 1995). The third, R3, is similar but rather than altering the variables themselves, it tries making the nonparametric variables linear. Any variable whose change of form increases the BIC by less than 7 is made linear. The second reduction procedure, R2, first applies R1 and then R3.

5 Results

This section implements GAGAM and examines its performance in an extensive simulation study and on two economic data sets. Throughout, GAMSEL is used as a benchmark[6].

The algorithm parameters used in the applications are: $no\_gen = 100$, $pop\_size = 500$, $p_m = 0.05$, $p_{mnp} = 0.10$, $K_{\text{Var}} = 30$ (for the simulation study and the second real data set) or $K_{\text{Var}} = 15$ (for the first real data set). In exploratory analysis, we found that running the algorithm for 100 generations is sufficient and additional iterations did not meaningfully affect the results. Increasing the population size (and appropriately scaling the other quantities in Algorithm 1) similarly did not yield better solutions.

GAMSEL is implemented with $\gamma = 0.5$ and 10 basis functions with 5 degrees of freedom. λ is chosen by 10-fold cross-validation according to the one standard error rule. This means that the entire sample is split into ten subsamples. GAMSEL is then estimated ten times, each time on nine subsamples, and those estimates are used to predict the left-out subsample. With the out-of-sample predictions for every observation we compute the cross-validated (CV) mean squared error (MSE) and its standard error. This is repeated for many λ's and we find the one which minimizes the CV MSE. We then increase λ as much as possible, subject to the constraint that CV MSE remains within one standard error of its minimum.

5.1 Simulation study

5.1.1 Data and Performance Metrics

We generally use the sample size of $n = 400$. Following Savin (2012), the independent variables are drawn from a multivariate normal distribution, $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, with the covariance matrix $\mathbf{\Sigma}$ set to either $\{\mathbf{\Sigma}\}_{i,j} = 0.5^{|i-j|}$ (low correlation) or $\{\mathbf{\Sigma}\}_{i,j} = 0.75^{|i-j|}$ (high correlation). We vary the signal-to-noise (SNR) ratio in the dependent variable between low noise (SNR = 5), medium noise (SNR = 2), and high noise (SNR = 0.5). The error term is then generated as $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, with σ given by:

$$\sigma = \frac{\sqrt{\text{Var}(\tilde{y})}}{SNR}$$

where $\tilde{y}$ is the dependent variable as generated from the predictors before adding the noise.

With this data we conduct five simulations[7]:

1. Variables 1–3 are linear, 4–6 are nonlinear, and 7–50 are zero.
2. Variables 1–20 are linear, 21–25 are nonlinear, and 26–50 are zero.

[6]We would ideally also compare our method to SPLAM but even after personal communication with one of the authors the code for SPLAM did not return satisfactory results.

[7]The exact data generating processes are in Appendix B.

3. Random 3 variables are linear, random 3 variables are nonlinear, 44 variables are zero.
4. Like simulation 1 but 7–1000 are zero[8].
5. Like simulation 1 but with $n = 100$.

Simulations 1 and 2 are the base simulations. The main difference between them is the level of sparsity. Also, in both 1 and 2 the truly nonzero predictors are all in the first 6 or 25 columns of the design matrix. Given our specification of $\mathbf{\Sigma}$, this means they are primarily correlated with each other.

In Simulation 3 we change this by placing the 6 relevant features in random columns of $\mathbf{X}$. This has the following effect. In simulation 1, the nonzero predictors were $x_1, \ldots x_6$, and the noise variables were $x_7, \ldots, x_{50}$. The correlation of x_1 with x_7 was either 0.50^6 or 0.75^6, and the correlation with the other noise predictors was even lower. In simulation 3 what was previously x_1 might now become x_{20}, and both x_{19} and x_{21} might be noise variables so that the former x_1 has a correlation of 0.5 or 0.75 with at least two noise variables. Similar arguments apply to the other relevant predictors. Moreover, because only 6 out of the 50 variables enter the true model, it is very likely that there will be many cases where the relevant variables are surrounded by, and thus highly correlated with, the noise ones.

Simulations 4 and 5 return back to the design matrix used in simulation 1. Simulation 4 explores GAGAM's capabilities in a case where the number of predictors greatly exceeds the number of observations, $p \gg n$. Simulation 5 tests the performance in small samples.

We perform 50 repeats of each simulation which entails generating new data and running GAGAM. The results are reported as average performance over the 50 restarts.

Performance is measured in terms of the ability to extract the true model and out-of-sample prediction error. We use five metrics:

- Zero: the number of noise regressors included in the model.
- Linear: the number of truly linear regressors excluded or mistaken for nonlinear.
- Nonlinear: the number of truly nonlinear regressors excluded or mistaken for linear.
- Zero vs. nonzero: the number of noise regressors included plus the number of relevant regressors excluded.
- RMSE: generate another 1000 observations of the data and compute the out-of-sample RMSE.

The linear and nonlinear mistakes can happen in two ways, e.g. a linear variable can be excluded *or* mistaken for nonlinear. Thus, we also break down the error by type.

All simulations were performed on an AWS-hosted machine with a 2nd generation 3.4GHz Intel Xeon Cascade Lake processor with 36 CPU cores. For GAGAM, we report the estimated time to run on a personal computer with 8 cores. We do not report GAMSEL's running time because it usually takes a few seconds.

Table 1 presents the results of simulation 1. Let us first examine the upper left quadrant (low correlation and low noise). GAGAM (labeled GA in the table) performs very well in this setting. In almost all repeats, the algorithm is able to extract all relevant predictors and assign them the correct form. Of the 44 irrelevant variables GAGAM, on average, includes less than 1. If we apply R1 (only reduce variables), GAGAM finds the true model in practically every restart of the simulation.

GAMSEL also tends to find all relevant predictors. However, it includes many noise

[8]In this case we make an exception use a population size of 1000.

variables (almost 19, on average). In addition, in a typical restart it estimates almost half of the linear variables nonparametrically. Whether these mistakes are due to λ being chosen by cross-validation or correlation between regressors is unclear. Furthermore, GAMSEL produces a substantially larger prediction error than GAGAM.

We can compare these results to the upper right quadrant where the predictor correlation is increased. The performance of GAGAM appears roughly unaffected while GAMSEL includes fewer noise variables. The difference in RMSE between the two methods remains approximately unchanged.

On the other hand, noise has a more severe effect (middle left quadrant). This is expected as GAGAM infers the recommended model from the data – a lower signal-to-noise ratio makes this task harder. While the algorithm's performance with zero and linear variables remains favorable with medium noise, it starts making mistakes with nonlinear variables. When such an error is made, the variable tends to be excluded. Nevertheless, overall performance remains satisfactory. However, with high noise (lower left quadrant) GAGAM starts making mistakes even with linear variables and almost all nonlinear predictors are either excluded or linear

We observe a similar trend with GAMSEL, though GAGAM remains superior, particularly in the medium-noise case. With high noise it would be fair to characterize both methods as performing similarly poorly.

The reduction techniques do not prove very valuable in this simulation. R1 can be helpful in the low noise setting, though the small difference in RMSE suggests that the noise variables already have near zero coefficients. With high noise, reduction can make things worse by excluding nonzero variables.

Table 1 also reveals the time required to run GAGAM. With low correlation and low noise this is about 25 minutes which is much longer than GAMSEL, which takes seconds, but not prohibitive. With more noise, the computational time falls because the algorithm starts including fewer nonparametric terms, speeding up estimation.

Table 2 presents the results of simulation 2. Overall, the results are similar to simulation 1. In absolute terms, GAGAM performs rather well and includes few irrelevant predictors. With low noise and low correlation, GAGAM, on average, confuses 3 out of the 20 linear variables, most of which are estimated as nonlinear. It does, however, have a much higher error rate with the nonlinear variables. This can perhaps be explained by the fact that every predictor now accounts for a smaller proportion of the target's variance. Thus, estimating a variable nonparametrically might not increase the log likelihood in the BIC by enough to offset the cost from additional degrees of freedom.

[Tables 1 and 2 about here]

High noise appears more problematic now that the true model is less sparse. GAGAM does not include a single nonlinear variable in any of the 50 restarts. Similarly, almost 15 out of the 20 linear variables are on average excluded (in simulation 1, for comparison, only 0.78 out of 5 linear variables were excluded in the high noise, low correlation setting).

Relative to GAMSEL, the trend from simulation 1 persists. GAMSEL experiences the same issues as GAGAM but the latter performs better in all six correlation-noise specifications. However, with high noise the difference becomes almost indistinguishable.

[Table 3 about here]

Table 3 contains selected results from simulations 3 (dispersed regressors), 4 ($p \gg n$), and 5 (small sample). In the top quadrant, GAGAM's performance is mostly unaffected by the higher correlation of the relevant regressors with the noise variables. GAMSEL, on the other hand, makes significantly more mistakes in all four categories. The low RMSE suggests that GAMSEL uses the correlated irrelevant predictors as substitutes for the truly nonzero regressors. GAGAM is noticeably less susceptible to this problem.

In simulation 4 (middle quadrant) we considered 1000 predictors, only five of which are relevant. GAGAM shows impressive ability to find the nonzero coefficients without introducing too many noise variables. Also, the reduction methods prove useful here. For instance, R1 reduces the number of irrelevant variables included down to just over 3 with no cost in terms of inducing mistakes on the nonzero variables. In contrast, GAMSEL includes over 40 irrelevant predictors. The difference in prediction error between GAGAM and GAMSEL is considerable.

Finally, simulation 5 examines performance of GAGAM in small samples. As expected, having only 100 observations hampers performance. On average, 0.56 linear variables are mistaken for nonlinear, compared to 0 with a sample size of 400. Similarly, GAGAM typically includes almost five noise variables, however, R1 and R2 are again useful and can eliminate about 50% of the irrelevant predictors. Like before, GAGAM outperforms GAMSEL on both the prediction error and in extracting the true model.

5.2 Boston Housing Data

[Table 4 about here]

In this section we apply GAGAM to the Boston Housing data set (Harrison & Rubinfeld 1978) which contains data on median house values (dependent variable) for 506 areas in Boston, MA, as well as 13 variables which might be relevant for predicting house prices (shown in Table 4). This data set has been used extensively in the machine learning literature, including in Chouldechova & Hastie (2015).

Our approach here is to replicate the experiment from Chouldechova & Hastie (2015). We first remove variables ZN, CHAS, and RAD because they are not sufficiently continuous to work with GAMSEL. This leaves ten predictors. Next, we generate ten more independent variables as random draws from the interval [0,1]. We also create ten regressors by randomly shuffling the rows of the original ten variables. We thus have 30 predictors in total, with the first 10 potentially helpful for predicting MEDV.

We apply GAGAM with no reduction and $K_{\text{Var}} = 15$ to this data set. Since GAGAM is stochastic, it may return different results in different runs and thus we restart it multiple times. However, the algorithm always returns the same model, suggesting that this is likely the global optimum of the BIC.

The results are encouraging. GAGAM does not include any of the irrelevant variables. It includes CRIM, NOX, and PTRATIO linearly and RM, DIS, TAX, and LSTAT nonparametrically. INDUS, AGE, and BLACK are excluded. Figure 11 shows the estimated functions for RM, LSTAT, NOX, and PTRATIO, along with the partial residuals. Our model describes the data reasonably well.

[Figure 11 about here]

GAMSEL performs worse. Like GAGAM, it estimates RM, TAX, and LSTAT nonlinearly (but not RM). NOX is also nonlinear with GAMSEL. CRIM, DIS, PTRATIO, and BLACK are thought to be linear while INDUS and AGE are excluded. However, GAMSEL also includes (linearly) 3 variables generated from the interval [0,1] and 4 of the variables created by randomly shuffling the original rows. Thus, the pattern from the simulations is repeated here – relative to GAGAM, GAMSEL is less conservative with including noise variables.

In addition, we compare predictive performance of the two methods using 10-fold cross-validation (same approach as for finding λ for GAMSEL). In doing that we apply GAGAM once on the entire sample to select the model and then use that specification in every fold. GAMSEL, on the other hand, reselects the variables each time. Similar to the simulations, GAGAM outperforms GAMSEL with an RMSE of 2.82 compared to 3.04.

5.3 Macroeconomic Time Series

We now apply GAGAM to forecasting macroeconomic series. To do this, we replicate, on a smaller scale, the study of Li & Chen (2014). The data is sourced from Stock & Watson (2012) and contains 107 representative macroeconomic time series from January 1959 to December 2008 (monthly observations). The complete list of the variables is in the Appendix of Li & Chen. We will attempt to forecast five variables: the US industrial production index, the University of Michigan index of consumer expectations, the unemployment rate (all workers aged 16 or over, %), total farm and nonfarm housing starts in the US, and the 10-year Treasury rate.

Our first step is to make all series stationary. This is accomplished by applying the transformations specified by Li & Chen (2014). Next, we compute the first four lags of each variable. Together, these two operations make us lose the first six observations. In addition, we remove the data points from 2008 because they represent outliers. Therefore, we are left with 582 observations between July 1959 and December 2007.

We will predict each of the five dependent variables with the first four lags of all 107 variables, i.e. including the autoregressive terms. The task is therefore to perform variable selection and structure discovery on these $4 \times 107 = 428$ predictors.

One concern is that parameter values are unstable over time in macroeconomic data sets (Stock & Watson 1996). This precludes us from applying GAGAM on the entire data set. Since it would be too computationally demanding to run it many times, we address this issue by using GAGAM on the last ten years of data only (120 observations). Since simulations suggested that reduction R2 can be helpful in small samples we apply it here.

Using the specification found, we apply the rolling regression procedure to estimate the forecast error. That is, we first estimate the GAGAM-recommended model using the first 120 observations of data (July 1959 to July 1969) and use that to predict the 121^{st} observation. Then, we drop the first observation, add the 121^{st}, reestimate the model, and predict the 122^{nd} observation. This is repeated until we get to the end of the sample. In total, the rolling regression produces 462 out-of-sample one-step-ahead forecasts.

As mentioned, GAGAM is a stochastic algorithm and may return different results in every run. Thus, we repeat this entire process five times and report the results of the best

and the worst run.

As a reference for GAGAM, we also attempt to forecast with GAMSEL. To do this, we again use the rolling procedure, reestimating GAMSEL in each subsample. Notice that this gives GAMSEL an a priori advantage over GAGAM since it is able to select the variables in every window while GAGAM is not. If the predictors helpful for forecasting the target variables have changed over time, GAMSEL may perform better simply because it is able to account for this.

As a benchmark for both methods we also estimate an autoregressive-moving-average (ARMA) model:

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

ARMA is applied in the same rolling fashion as GAMSEL. Orders p and q are chosen in each estimation window separately using the `auto.arima` function of the `forecast` package in `R` (Hyndman & Khandakar 2008).

Having produced the out-of-sample forecasts for all three forecasting methods, we use RMSE as the estimate of the error. To test whether the observed differences in RMSEs are statistically significant, and because the models are not nested, we implement the Diebold-Mariano test of equal predictive accuracy[9] (Diebold & Mariano 1995). The p-value from this test is interpreted as the probability of finding a difference in RMSEs between two methods as big or bigger as observed assuming that the two methods have the same predictive ability.

[Table 5 about here]

Table 5 reports the root mean squared forecasting errors for ARMA, GAMSEL, and the five repeats of GAGAM for all five series. The lowest RMSE in each row is in bold. The table also reports the realized run time of each GAGAM repeat, this time on a base 2019 MacBook Pro (8 CPU cores).

The results demonstrate that GAGAM is as good as or better than GAMSEL and ARMA in all five cases. In forecasting industial production, the best GAGAM run produces a noticeably lower RMSE than the competing methods, and the difference is highly statistically significant. For the next three series GAGAM is not statistically significantly better than GAMSEL, although it does return a lower forecast error and it is significantly better than ARMA. In the case of the 10-year Treasury rate, ARMA outperforms both GAMSEL and GAGAM, though not by much.

We see two interpretations for these results. First, the last four series are likely inherently very unpredictable, e.g. the efficient market hypothesis suggests that the 10-year Treasury rate should be unforecastable. If that is the case, then this forecasting exercise is similar

[9] If e_{1t} and e_{2t} are the forecast errors of the two methods under comparison, and we use (root) mean squared error as the loss function, then we can write the differences between the loss functions of the two methods as $d_t = e_{1t}^2 - e_{2t}^2$. The sample mean of these differences is $\bar{d} = n^{-1} \sum_{t=1}^{n} d_t$, and the asymptotic variance of $\bar{d}$ is $V\left(\bar{d}\right) \approx n^{-1} \left[\gamma_0 + 2\sum_{k=1}^{h-1} \gamma_k\right]$, where γ_k is the kth autocovariance of d_t and h is the forecast horizon (1 in our case). The Diebold-Mariano statistic is then $S_1 = \frac{\bar{d}}{\sqrt{\hat{V}(\bar{d})}}$. Under $H_0 : \mathbb{E}[d_t] = 0$ and the Diebold-Mariano assumption (1. $\mathbb{E}[d_t] = \mu, \forall t$, 2. $\mathrm{Cov}(d_t, d_{t-\tau}) = \gamma(\tau), \forall t$, 3. $0 < \mathrm{Var}(d_t) = \sigma^2 < \infty$), $S_1 \to \mathcal{N}(0,1)$.

to the high noise simulations presented above, and it is in line with expectations that both GAMSEL and GAGAM perform equally badly. Industrial production, on the other hand, might be more predictable which allows GAGAM to showcase its capabilities.

Another interpretation could be that GAMSEL has an advantage since it gets to select the variables in every rolling estimation window. However, we should note that this problem only relates to estimating the forecast error, not actual forecasting. Namely, it would be too time consuming to re-run GAGAM in each of the 462 subsamples but it is not too expensive to run once a year. Thus, in real-world applications, GAGAM would be equally capable of accounting for relevant variables changing over time.

In addition, we note that the times to run the algorithm vary substantially. This is because the population of models in GAGAM might sometimes include models with more nonparametric variables which increases the estimation time. Even so, in all cases the time required is not prohibitive.

6 Implementation in R

We implement GAGAM in an `R` package[10], thus making it easy to use in practice. While this work has only focused on the applications where the response variable has a Gaussian distribution, the package supports all distributions covered by the `mgcv` package (Wood 2019). We also allow the user to tweak any of the algorithm's or the model's parameters, e.g. the basis used for nonparametric terms, the number of knots etc. The extension to interactions presented in Appendix A is included. A user guide for the package is available HERE [IN DEVELOPMENT].

7 Discussion

This paper addressed the problem of simultaneous variable selection and structure discovery in semiparametric generalized additive models (GAM). While this has been attempted before using penalized regression, those methods are not selection consistent with correlated regressors or when the rate of penalization is chosen to minimize prediction error, as is usually the case in practice. To alleviate these issues, we develop GAGAM, a genetic algorithm enabling model comparison based on the BIC in high-dimensional settings.

We demonstrate the performance of GAGAM on synthetic and real data, and compare it to a competing method, Chouldechova & Hastie's (2015) GAMSEL. Simulations reveal that GAGAM consistently outperforms GAMSEL in model selection, introducing fewer noise variables. Particularly in low and medium noise settings, GAGAM also produces substantially more accurate predictions. We then apply both methods to the Boston Housing data set and to forecasting macroeconomic time series. In the former case, GAGAM again commits fewer false positive errors and yields a noticeably lower RMSE. With time series, GAGAM is at least as good as the competitors in all five experiments. Predicting industrial production, GAGAM generates forecasts which are statistically significantly more accurate than

[10]Install using the command `library(devtools); install_github(markcus1/gagam)`. [DO NOT DOWNLOAD - IN DEVELOPMENT]

GAMSEL's.

In addition, GAGAM is only a wrapper which estimates models using conventional techniques. This implies that it can, for instance, be applied with any type of dependent variable distribution for which estimation methods are available (e.g. binomial and negative binomial, Gamma, inverse Gaussian, Poisson etc.). Also, GAGAM can be used with predictors which are not continuous and allows the analyst more flexibility with modeling nonparametric terms, for instance, allowing any desired basis. On the other hand, the competing methods only support the Gaussian and binomial distributions for the response, and cannot intake discrete predictors.

On the other hand, GAGAM has two main drawbacks. First, it performs poorly in high noise settings. The reason is that it tries to use the data to extract the best model, and if the data are noisy, this task is much harder. However, this feature is shared by GAMSEL.

Second, it is substantially more computationally demanding than the penalized regression approaches. In our experiments, the time required for one run on a personal computer varied from 10 minutes to an hour and a half. This presents an obstacle but is not prohibitive. However, the computational intensity increases quickly with the number of nonparametric terms, making the use of GAGAM infeasible for large models with many nonlinear predictors.

Computational improvements thus present an avenue for future research. Similarly, rather than using GAGAM, it may be possible to address the issues of group lasso using adaptive group lasso (Wang & Leng 2008).

References

Acosta-González, E. & Fernández-Rodríguez, F. (2007), 'Model selection via genetic algorithms illustrated with cross-country growth data', *Empirical Economics* **33**(2), 313–337.
URL: *https://doi.org/10.1007/s00181-006-0104-3*

Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* **19**(6), 716–723.

Alba, E., Luque, G. & Araujo, L. (2006), 'Natural language tagging with genetic algorithms', *Information Processing Letters* **100**(5), 173–182.
URL: *http://www.sciencedirect.com/science/article/pii/S002001900600216X*

Axelrod, R. & Davis, L. (1987), The evolution of strategies in the iterated Prisoner's Dilemma, *in* 'Genetic Algorithms and Simulated Annealing', Morgan Kaufmann Publishers Inc., pp. 32–41.

Bachmeier, L., Leelahanon, S. & Li, Q. I. (2007), 'Money Growth and Inflation in the United States', *Macroeconomic Dynamics* **11**(1), 113–127.

Bai, J. & Ng, S. (2008), 'Forecasting economic time series using targeted predictors', *Journal of Econometrics* **146**(2), 304–317.
URL: *http://www.sciencedirect.com/science/article/pii/S0304407608001085*

Beenstock, M. & Szpiro, G. (2002), 'Specification search in nonlinear time-series models using the genetic algorithm', *Journal of Economic Dynamics and Control* **26**(5), 811–835.
URL: *http://www.sciencedirect.com/science/article/pii/S016518890000083X*

Bernanke, B. S., Boivin, J. & Eliasz, P. (2004), Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach, Technical Report 10220, National Bureau of Economic Research.
URL: *http://www.nber.org/papers/w10220*

Bien, J., Taylor, J. & Tibshirani, R. (2013), 'A lasso for hierarchical interactions', *Ann. Statist.* **41**(3), 1111–1141.
URL: *https://projecteuclid.org:443/euclid.aos/1371150895*

Breheny, P. & Huang, J. (2015), 'Group Descent Algorithms for Nonconvex Penalized Linear and Logistic Regression Models with Grouped Predictors', *Statistics and Computing* **25**(2), 173–187.
URL: *https://doi.org/10.1007/s11222-013-9424-2*

Broadhurst, D., Goodacre, R., Jones, A., Rowland, J. J. & Kell, D. B. (1997), 'Genetic algorithms as a method for variable selection in multiple linear regression and partial least squares regression, with applications to pyrolysis mass spectrometry', *Analytica Chimica Acta* **348**(1), 71–86.
URL: *http://www.sciencedirect.com/science/article/pii/S0003267097000652*

Chouldechova, A. & Hastie, T. (2015), 'Generalized Additive Model Selection', *arXiv:1506.03850 [stat]* .
URL: *http://arxiv.org/abs/1506.03850*

Demmler, A. & Reinsch, C. (1975), 'Oscillation matrices with spline smoothing', *Numerische Mathematik* **24**(5), 375–382.
URL: *https://doi.org/10.1007/BF01437406*

Desboulets, L. D. D. (2018), 'A Review on Variable Selection in Regression Analysis', *MDPI, Open Access Journal: Econometrics* **6**(4), 1–27.
URL: *https://ideas.repec.org/a/gam/jecnmx/v6y2018i4p45-d185046.html*

Diebold, F. X. & Mariano, R. S. (1995), 'Comparing predictive accuracy', *Journal of Business and Economic Statistics* **13**(3), 253–263.

Dudek, G. (2012), Variable Selection in the Kernel Regression Based Short-Term Load Forecasting Model, *in* L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh & J. M. Zurada, eds, 'International Conference on Artificial Intelligence and Soft Computing 2012', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 557–563.

Gan, C. C. & Learmonth, G. (2016), 'An improved chromosome formulation for genetic algorithms applied to variable selection with the inclusion of interaction terms', *arXiv:1604.06727 [cs, stat]* .
URL: *http://arxiv.org/abs/1604.06727*

Gartner, W. B. & Thomas, R. J. (1993), 'Factors affecting new product forecasting accuracy in new firms', *Journal of Product Innovation Management* **10**(1), 35–52.
URL: *http://www.sciencedirect.com/science/article/pii/073767829390052R*

Harrison, D. & Rubinfeld, D. L. (1978), 'Hedonic housing prices and the demand for clean air', *Journal of Environmental Economics and Management* **5**(1), 81–102.
URL: *http://www.sciencedirect.com/science/article/pii/0095069678900062*

Hasheminia, H. & Akhavan Niaki, S. T. (2006), 'A Genetic Algorithm Approach to Find the Best Regression/Econometric Model Among the Candidates', *Appl. Math. Comput.* **183**(1), 337–349.
URL: *http://dx.doi.org/10.1016/j.amc.2006.05.072*

Hastie, T. & Tibshirani, R. (1990), *Generalized additive models*, Chapman & Hall/CRC, Boca Raton, Fla.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd edn, Springer, New York.

Hoerl, A. E. & Kennard, R. W. (1970), 'Ridge Regression: Biased Estimation for Nonorthogonal Problems', *Technometrics* **12**(1), 55–67.
URL: *https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634*

Holland, J. H. (1975), *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, Oxford, England.

Huang, J., Breheny, P. & Ma, S. (2012), 'A Selective Review of Group Selection in High-Dimensional Models', *Statistical Science* **27**(4), 481–499.
URL: *http://www.jstor.org/stable/41714780*

Huang, J., Horowitz, J. L. & Wei, F. (2010), 'Variable Selection in Nonparametric Additive Models', *The Annals of Statistics* **38**(4), 2282–2313.
URL: *http://www.jstor.org/stable/20744490*

Huang, J., Wei, F. & Ma, S. (2012), 'Semiparametric Regression Pursuit', *Statistica Sinica* **22**(4), 1403–1426.
URL: *http://www.jstor.org/stable/24310181*

Hyndman, R. & Khandakar, Y. (2008), 'Automatic Time Series Forecasting: The forecast Package for R', *Journal of Statistical Software* **27**(3), 1–22.
URL: *https://www.jstatsoft.org/v027/i03*

Jeong, Y.-S., Shin, K. S. & Jeong, M. K. (2015), 'An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems', *Journal of the Operational Research Society* **66**(4), 529–538.
URL: *https://doi.org/10.1057/jors.2013.72*

Kass, R. E. & Raftery, A. E. (1995), 'Bayes Factors', *Journal of the American Statistical Association* **90**(430), 773–795.
URL: *http://www.jstor.org/stable/2291091*

Leardi, R., Boggia, R. & Terrile, M. (1992), 'Genetic algorithms as a strategy for feature selection', *Journal of Chemometrics* **6**(5), 267–281.
URL: *https://doi.org/10.1002/cem.1180060506*

Leng, C., Lin, Y. & Wahba, G. (2006), 'A Note on the Lasso and Related Procedures in Model Selection', *Statistica Sinica* **16**(4), 1273–1284.
URL: *http://www.jstor.org/stable/24307787*

Li, J. & Chen, W. (2014), 'Forecasting macroeconomic time series: LASSO-based approaches and their forecast combinations with dynamic factor models', *International Journal of Forecasting* **30**(4), 996–1015.
URL: *http://www.sciencedirect.com/science/article/pii/S0169207014000636*

Li, Q. & Racine, J. S. (2007), *Nonparametric Econometrics: Theory and Practice*, Princeton University Press, Princeton.

Lian, H., Chen, X. & Yang, J.-Y. (2012), 'Identification of Partially Linear Structure in Additive Models with an Application to Gene Expression Prediction from Sequences', *Biometrics* **68**(2), 437–445.
URL: *https://doi.org/10.1111/j.1541-0420.2011.01672.x*

Lian, H. & Liang, H. (2013), 'Generalized Additive Partial Linear Models with High-dimensional Covariates', *Econometric Theory* **29**(6), 1136–1161.
URL: *https://www.cambridge.org/core/article/generalized-additive-partial-linear-models-with-highdimensional-covariates/386198295BEE3DA504FB90563B9FEC76*

Lim, M. & Hastie, T. (2015), 'Learning Interactions via Hierarchical Group-Lasso Regularization', *Journal of Computational and Graphical Statistics* **24**(3), 627–654.
URL: *https://doi.org/10.1080/10618600.2014.938812*

Liu, X., Wang, L. & Liang, H. (2011), 'Estimation and Variable Selection for Semiparametric Additive

Partial Linear Models', *Statistica Sinica* **21**(3), 1225–1248.
URL: *http://www.jstor.org/stable/24309561*

Lou, Y., Bien, J., Caruana, R. & Gehrke, J. (2016), 'Sparse Partially Linear Additive Models', *Journal of Computational and Graphical Statistics* **25**(4), 1126–1140.
URL: *https://doi.org/10.1080/10618600.2015.1089775*

Louizos, C., Welling, M. & Kingma, D. P. (2017), 'Learning Sparse Neural Networks through L0 Regularization'.
URL: *http://arxiv.org/abs/1712.01312*

Meinshausen, N. & Bühlmann, P. (2006), 'High-Dimensional Graphs and Variable Selection with the Lasso', *The Annals of Statistics* **34**(3), 1436–1462.
URL: *http://www.jstor.org/stable/25463463*

Meyer, M. C. & Liao, X. (2019), 'cgam: Constrained Generalized Additive Model'.

Mitchell, M. (1996), *An introduction to genetic algorithms*, MIT Press, Cambridge, Mass.

Müller, P. & van de Geer, S. (2015), 'The Partial Linear Model in High Dimensions', *Scandinavian Journal of Statistics* **42**(2), 580–608.
URL: *https://doi.org/10.1111/sjos.12124*

Nadaraya, E. (1964), 'On Estimating Regression', *Theory of Probability & Its Applications* **9**(1), 141–142.

Natarajan, B. K. (1995), 'Sparse Approximate Solutions to Linear Systems', *SIAM J. Comput.* **24**(2), 227–234.
URL: *https://doi.org/10.1137/S0097539792240406*

Ng, A. Y. (1997), Preventing "Overfitting" of Cross-Validation Data, *in* 'Proceedings of the Fourteenth International Conference on Machine Learning', ICML '97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 245–253.

Paterlini, S. & Minerva, T. (2010), Regression Model Selection Using Genetic Algorithms, *in* 'Proceedings of the 11th WSEAS International Conference on Nural Networks and 11th WSEAS International Conference on Evolutionary Computing and 11th WSEAS International Conference on Fuzzy Systems', NN'10/EC'10/FS'10, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 19–27.
URL: *http://dl.acm.org/citation.cfm?id=1863431.1863437*

Pike, T. & Savage, D. (1998), 'Forecasting the Public Finances in the Treasury', *Fiscal Studies* **19**(1), 49–62.
URL: *http://www.jstor.org/stable/24437688*

Radchenko, P. & James, G. M. (2010), 'Variable Selection Using Adaptive Nonlinear Interaction Structures in High Dimensions', *Journal of the American Statistical Association* **105**(492), 1541–1553.
URL: *https://doi.org/10.1198/jasa.2010.tm10130*

Ravikumar, P., Lafferty, J., Liu, H. & Wasserman, L. (2009), 'Sparse additive models', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(5), 1009–1030.
URL: *https://doi.org/10.1111/j.1467-9868.2009.00718.x*

Savin, I. (2012), A Comparative Study of the Lasso-Type and Heuristic Model Selection Methods.
URL: *https://ssrn.com/abstract=2104428*

Schmertmann, C. P. (1996), 'Functional search in economics using genetic programming', *Computational Economics* **9**(4), 275–298.
URL: *https://doi.org/10.1007/BF00119476*

Schulze-Kremer, S. (2003), Application of Evolutionary Computation to Protein Folding BT - Advances in Evolutionary Computing: Theory and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 915–940.

Schwarz, G. (1978), 'Estimating the Dimension of a Model', *The Annals of Statistics* **6**(2), 461–464.
URL: *http://www.jstor.org/stable/2958889*

Sin, C.-Y. & White, H. (1996), 'Information criteria for selecting possibly misspecified parametric models', *Journal of Econometrics* **71**(1), 207–225.
URL: *http://www.sciencedirect.com/science/article/pii/0304407694017018*

Sousa, R. S., de Lima, T. W., de Paula, L. C. M., Lima, R. L., Filho, A. R. G. & Soares, A. S. (2017), Integer-based genetic algorithm for feature selection in multivariate calibration, *in* '2017 IEEE Congress on Evolutionary Computation (CEC)', IEEE, San Sebastian, pp. 2315–2320.

Stock, J. H. & Watson, M. W. (1996), 'Evidence on Structural Instability in Macroeconomic Time Series

Relations', *Journal of Business & Economic Statistics* **14**(1), 11–30.
URL: *https://www.tandfonline.com/doi/abs/10.1080/07350015.1996.10524626*

Stock, J. H. & Watson, M. W. (2012), 'Generalized Shrinkage Methods for Forecasting Using Many Predictors', *Journal of Business & Economic Statistics* **30**(4), 481–493.
URL: *https://doi.org/10.1080/07350015.2012.715956*

Szpiro, G. G. (1997), 'A Search for Hidden Relationships: Data Mining with Genetic Algorithms', *Computational Economics* **10**(3), 267–277.
URL: *https://doi.org/10.1023/A:1008673309609*

Tibshirani, R. (1996), 'Regression Shrinkage and Selection via the Lasso', *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(1), 267–288.
URL: *http://www.jstor.org/stable/2346178*

Wang, H. & Leng, C. (2008), 'A note on adaptive group lasso', *Computational Statistics & Data Analysis* **52**(12), 5277–5286.
URL: *http://www.sciencedirect.com/science/article/pii/S0167947308002582*

Watson, G. S. (1964), 'Smooth Regression Analysis', *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)* **26**(4), 359–372.
URL: *http://www.jstor.org/stable/25049340*

Wood, S. N. (2017), *Generalized Additive Models: An Introduction with R*, 2nd edn, CRC Press, Boca Raton.

Wood, S. N. (2019), 'mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation'.
URL: *https://cran.r-project.org/web/packages/mgcv/index.html*

Wood, S. N., Pya, N. & Säfken, B. (2016), 'Smoothing Parameter and Model Selection for General Smooth Models', *Journal of the American Statistical Association* **111**(516), 1548–1563.
URL: *https://doi.org/10.1080/01621459.2016.1180986*

Yang, L., Fang, Y., Wang, J. & Shao, Y. (2017), 'Variable selection for partially linear models via learning gradients', *Electron. J. Statist.* **11**(2), 2907–2930.
URL: *https://projecteuclid.org:443/euclid.ejs/1502157627*

Yuan, M. & Lin, Y. (2006), 'Model selection and estimation in regression with grouped variables', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(1), 49–67.
URL: *https://doi.org/10.1111/j.1467-9868.2005.00532.x*

Zhang, C.-H. (2010), 'Nearly Unbiased Variable Selection Under Minimax Concave Penalty', *The Annals of Statistics* **38**(2), 894–942.
URL: *http://www.jstor.org/stable/25662264*

Zhang, H. H., Cheng, G. & Liu, Y. (2011), 'Linear or Nonlinear? Automatic Structure Discovery for Partially Linear Models', *Journal of the American Statistical Association* **106**(495), 1099–1112.
URL: *http://www.jstor.org/stable/23427577*

Zhao, P. & Yu, B. (2006), 'On Model Selection Consistency of Lasso', *J. Mach. Learn. Res.* **7**, 2541–2563.

A Extension of GAGAM to Interactions

As an additional step in complexity, we might allow semiparametric GAMs to include first-order interactions:

$$g(\mathbb{E}(y_i)) = \alpha + \sum_{j\in\mathcal{L}} \beta_j x_{ij} + \sum_{k\in\mathcal{N}} f_k(x_{ik}) + \sum_{\{j,k\}\in\mathcal{I}} \beta_{j:k} x_{ij} x_{ik} + \sum_{\{j,k\}\in\mathcal{M}} f_{j:k}(x_{ij}, x_{ik}) + \varepsilon_i \quad (15)$$

This directly extends the problem from the paper to the interaction terms – how to allocate the interactions between $\mathcal{I}$ and $\mathcal{M}$, and which to exclude. In addition, to make the models interpretable, we usually require that these interactions be present in a hierarchical fashion, meaning that both main effects have to be included for an interaction to be legal.

Variations of this question have been addressed in the literature. Bien et al. (2013) and Lim & Hastie (2015) developed lasso methods for hierarchical interaction selection in fully parametric models. Radchenko & James (2010) present a similar method for main effect and interaction selection in additive models without linear components. However, to the best of my knowledge, there exists no technique for simultaneous variable selection and structure discovery for both main effects and hierarchical interactions.

[Figure 12 about here]

It is relatively straightforward to adapt GAGAM for this purpose. First, the coding is augmented with another block which represents the interactions included (top row) and their form (bottom row; see Figure **??**). The column-wise length of the interactions block (denoted K_{Int}) sets a limit on the maximum number of interactions allowed in the model. Second, we extend the crossover and mutation operators.

In the former case, we first apply regular crossover to the main effects exactly the same as before. This produces two children main effect blocks (denoted C1-M and C2-M). Then, we make two copies of each parent's interaction block. Let us call them P1-1 (first copy of parent 1's interactions), P1-2 (second copy of parent 1's interactions), P2-1 (first copy of parent 2's interactions), and P2-2 (second copy of parent 2's interactions). Now, take P1-1 and P2-1 and delete the interactions illegal according to C1-M. Finally, crossover the corrected P1-1 and P2-1 to produce the interactions block[11] for child 1. Appending this to C1-M yields the complete child model 1. The same is repeated with P1-2, P2-2, and C2-M for the second child.

The adapted mutation operator is simpler. We first mutate the main effects, then remove the interactions that have become illegal, and then we mutate the interactions with the original operator. When a mutation would result in including a new interaction we ensure that only legal interactions are being chosen from.

With these adaptations in hand, the rest of the algorithm remains identical to the original GAGAM.

We test this version of GAGAM in two simulations (A1 and A2). Simulation A1 is identical to simulation 1 above. In other words, there are no interactions in the true model. Simulation A2 is based on simulation 1 but we also add six interactions, three linear and three

[11] This will actually produce two new interaction blocks but we only keep the first one.

nonlinear. We again perform 50 repeats for each and report the mistakes in model selection, now for both main effects and interactions, as well as out-of-sample RMSE. Since no other method has been developed for this purpose, we compare the prediction error of GAGAM to the oracle (the true model). The algorithm is implemented with the same parameters as before and with $K_{\text{Int}} = 15$. No reduction method is applied. All data is generated with low correlation and low noise.

[Table 6 about here]

Table **??** shows the results. Allowing for the presence of interactions where there were none (A1) did not have a big effect on GAGAM which still performed nearly as well as in simulation 1. It made few mistakes in identifying the true model, e.g. on average it included 0.66 interactions, and the prediction performance is close to the oracle's.

In simulation A2 the true model included six interactions and GAGAM was able to find most of them, though it performed worse with the nonlinear interactions. Also, GAGAM's performance with the main effects is now noticeably worse. However, overall, the results are still respectable, particularly when we examine the prediction error.

Nevertheless, we make two observations about GAGAM with interactions. First, Table **??** shows results from a somewhat ideal data set (relatively large sample, low noise, low correlation). Thus, GAGAM's performance is far from impressive, and the struggles in simulation A2 would likely be magnified in higher noise data. Secondly and more importantly, the computational expense of estimating GAMs with interactions makes GAGAM infeasible in most cases. For instance, it would, on average, take about three hours to complete one run of simulation A2 on a personal computer.

For these reasons, we do not think that GAGAM is particularly promising when the data may include important interactions, and other methods should be developed.

B Data Generating Processes for Simulations

In the simulation study, we use the following data generating processes:

$$\text{DGP 1:} \qquad y_i = \alpha + \mathbf{X}_i\beta_{\text{L}} + f_1(x_{i,4}) + f_2(x_{i,5}) + f_3(x_{i,6}) + \varepsilon_i$$

$$\text{DGP 2:} \qquad y_i = \alpha + \mathbf{X}_i\beta_{\text{H}} + f_1(x_{i,21}) + f_2(x_{i,22}) + f_3(x_{i,23}) + f_4(x_{i,24}) + f_5(x_{i,25}) + \varepsilon$$

where $\alpha = 3$, $\beta_{\text{H}} = (7, 9, -3, 4, -9, 1, 2, 6, 8, 3, -6, 10, -5, -4, -10, 5, 4, 7, -4, 2, 0, \ldots, 0)^{\text{T}}$, and $\beta_{\text{L}} = (-8, 10, 3, 0, \ldots, 0)^{\text{T}}$. The nonlinear functions are given by:

$$f_1(x) = \begin{cases} 10x^{\frac{1}{6}x} & \text{if } x > 0 \\ 10 & \text{if } x = 0 \\ 10(-x)^{\frac{1}{6}x} & \text{if } x < 0 \end{cases} \qquad f_2(x) = 10\frac{\sin(\pi x)}{\pi x} \qquad f_3(x) = x + x^2 + x^3$$

$$f_4(x) = \exp(x) + 1.5 \qquad f_5(x) = 3.2|\sin(x)|$$

Simulations 1, 4, and 5 use DGP 1, simulation 2 uses DGP 2, and simulation 3 uses DGP 1 but instead of concentrating the relevant predictors in the first six columns of $\mathbf{X}$ they are assigned random columns (with β_{L} and the indices in DGP 1 changed appropriately).

C Figures

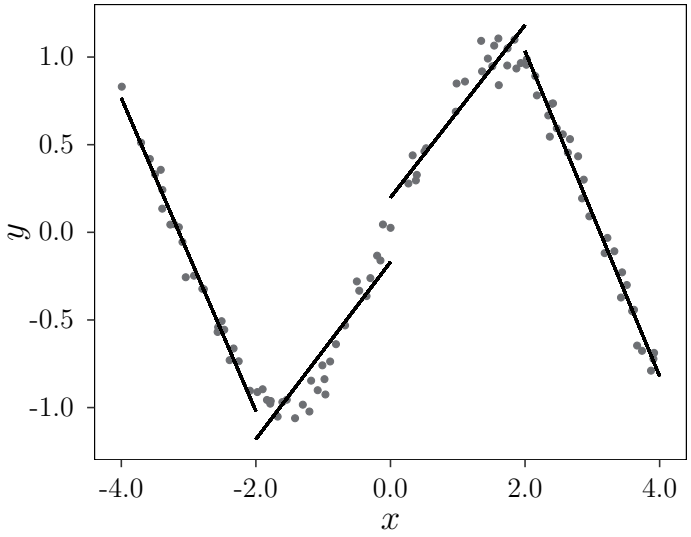

Figure 1: An example of fitting a model of the form $y_i = f(x_i)+\varepsilon$. The range of x is separated into four regions with the values of $x \in \{-2, 0, 2\}$ acting as the knots. We then fit a linear relationship in each region of the scatter plot.

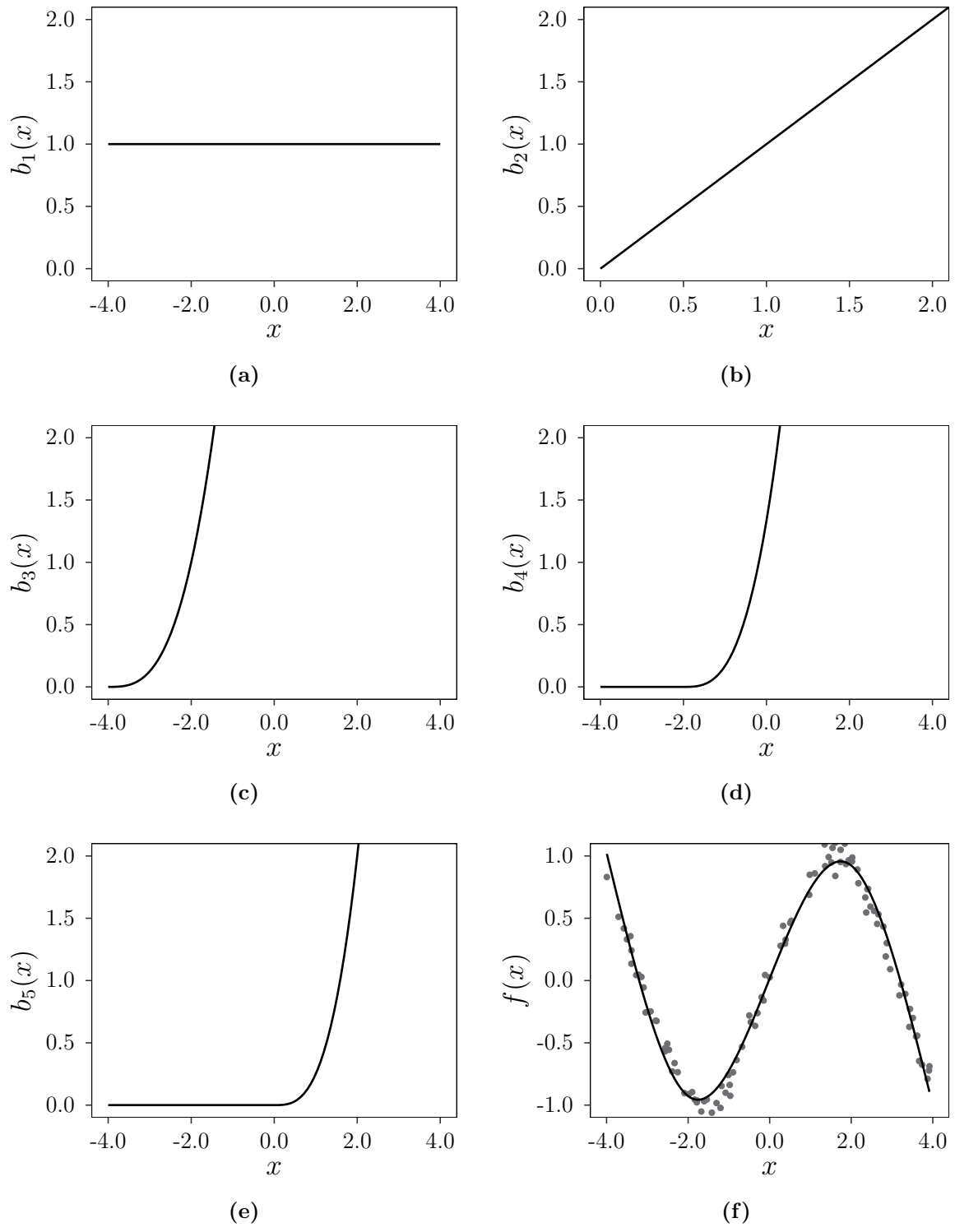

Figure 2: Panels (a) to (e) show the five basis functions (given in Equation 6) we use to fit a spline to the data in Figure 1. Panel (f) shows the fitted spline given by $\hat{f}(x) = -4.33b_1(x) - 1.34b_2(x) + 0.73b_3(x) - 1.10b_4(x) + 0.03b_5(x)$.

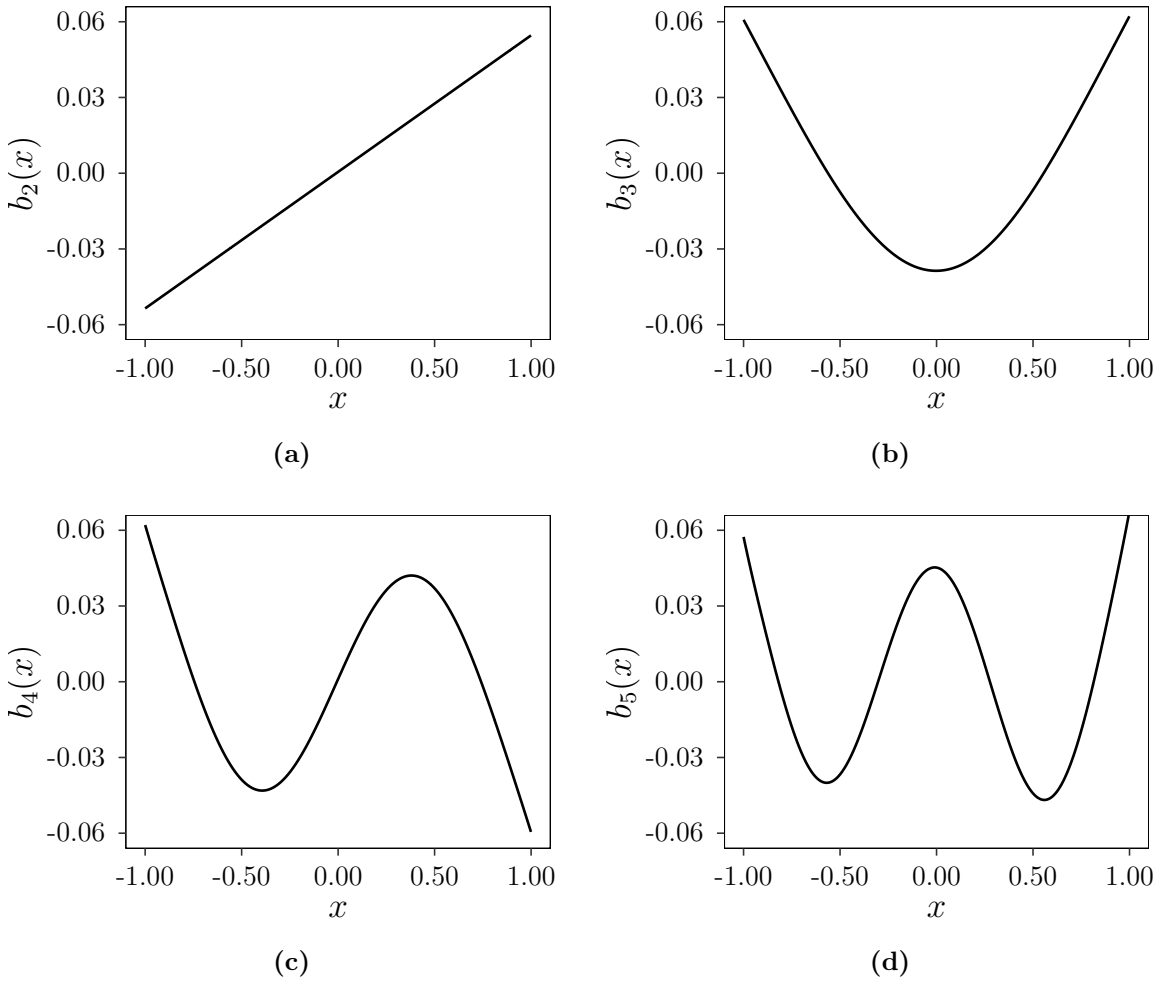

Figure 3: Columns 2 to 5 of the Demmler-Reinsch basis design matrix $\mathbf{B}_j$, $b_2(x), \ldots, b_5(x)$ (Column 1, $b_1(x)$, not shown because it is just a constant).

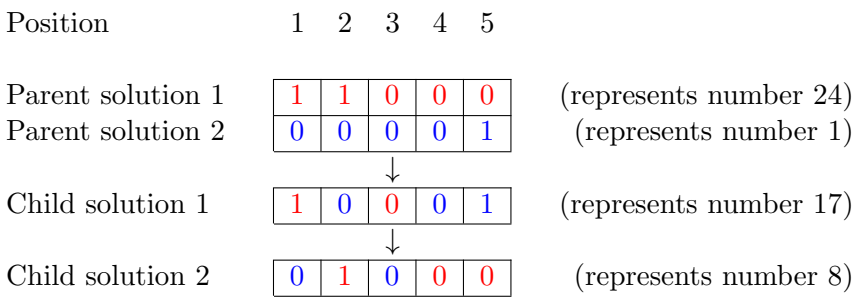

Figure 4: An example of uniform crossover. The bit in each position in the child solution 1 is set equal to the value of that bit in parent 1 with some probability, say, 0.5, and to the value of that bit in parent 2 with the probability 1 - 0.5. This recombines the features of the two parent solutions into a new child solution. The not chosen bits are placed in child solution 2.

Position	1	2	3	4	5	
Solution	0	1	0	0	0	(represents number 8)
			↓			
Mutated solution	0	1	0	0	1	(represents number 9)

Figure 5: An example of uniform mutation. The bit in each position of the original solution gets mutated (flipped from 0 to 1 or vice versa) with some low probability, say, 0.05. In the example shown, the bit in position 5 got mutated from 0 to 1.

Variables	x_1	x_2	x_3	x_4	x_5
Binary form	0	1	0	1	0

Meaning $y = \alpha + \beta_2 x_2 + \beta_4 x_4 + u$

Figure 6: An example illustrating binary coding in a GA for variable selection. 0 indicates that the variable is excluded and 1 indicates that it is included.

Model code	3	7	0	14	0	1

Meaning $y = \alpha + \beta_3 x_3 + \beta_7 x_7 + \beta_{14} x_{14} + \beta_1 x_1 + \varepsilon$

Figure 7: An example of integer coding based on Gan & Learmonth (2016). We are considering variables indexed from 1 to 20. The number in each bit tells us the index of the variable included in the model; zeros are placeholders that do not represent any variable.

Model code									
	Variables	1	3	7	0	0	2	0	0
	Form	1	0	1			0		

Meaning $y = \alpha + f_1(x_1) + \beta_3 x_3 + f_7(x_7) + \beta_2 x_2 + \varepsilon$

Figure 8: An example of mixed coding used in GAGAM. The integers in the top row are the indices of the variables included in the model; 0s do not represent any variable. The binary string in the bottom row determines whether each included predictor enters linearly (0) or nonparametrically (1).

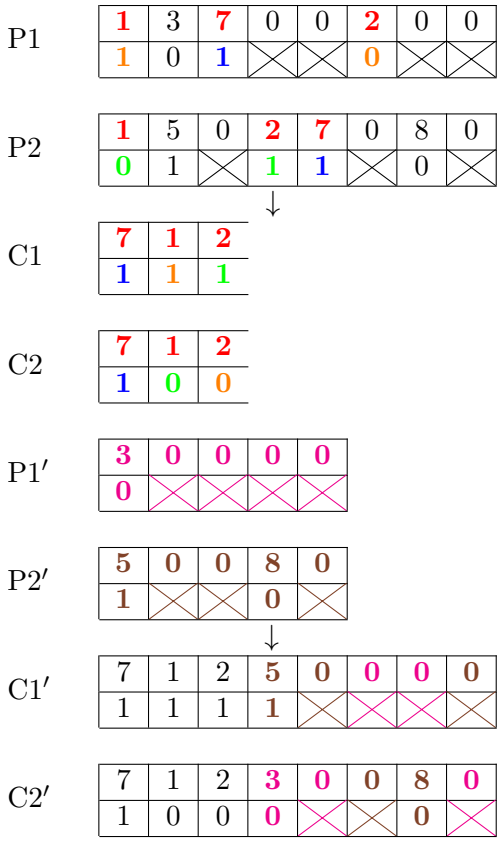

Figure 9: An example of the crossover operator used in GAGAM. P stands for parent model and C stands for child model. Section 4.1 explains how the operator works.

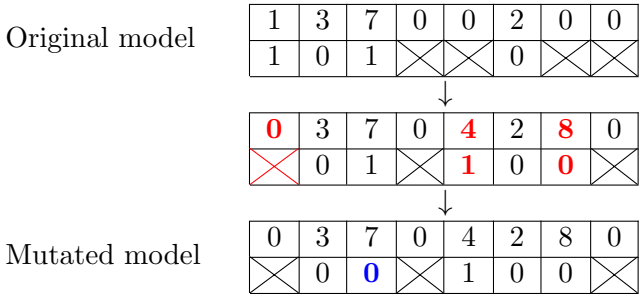

Figure 10: An example of the mutation operator used in GAGAM. Variable 1 got mutated to zero; two placeholders got mutated, one to variable 4 and one to variable 8; variable 7 had its form mutated to linear.

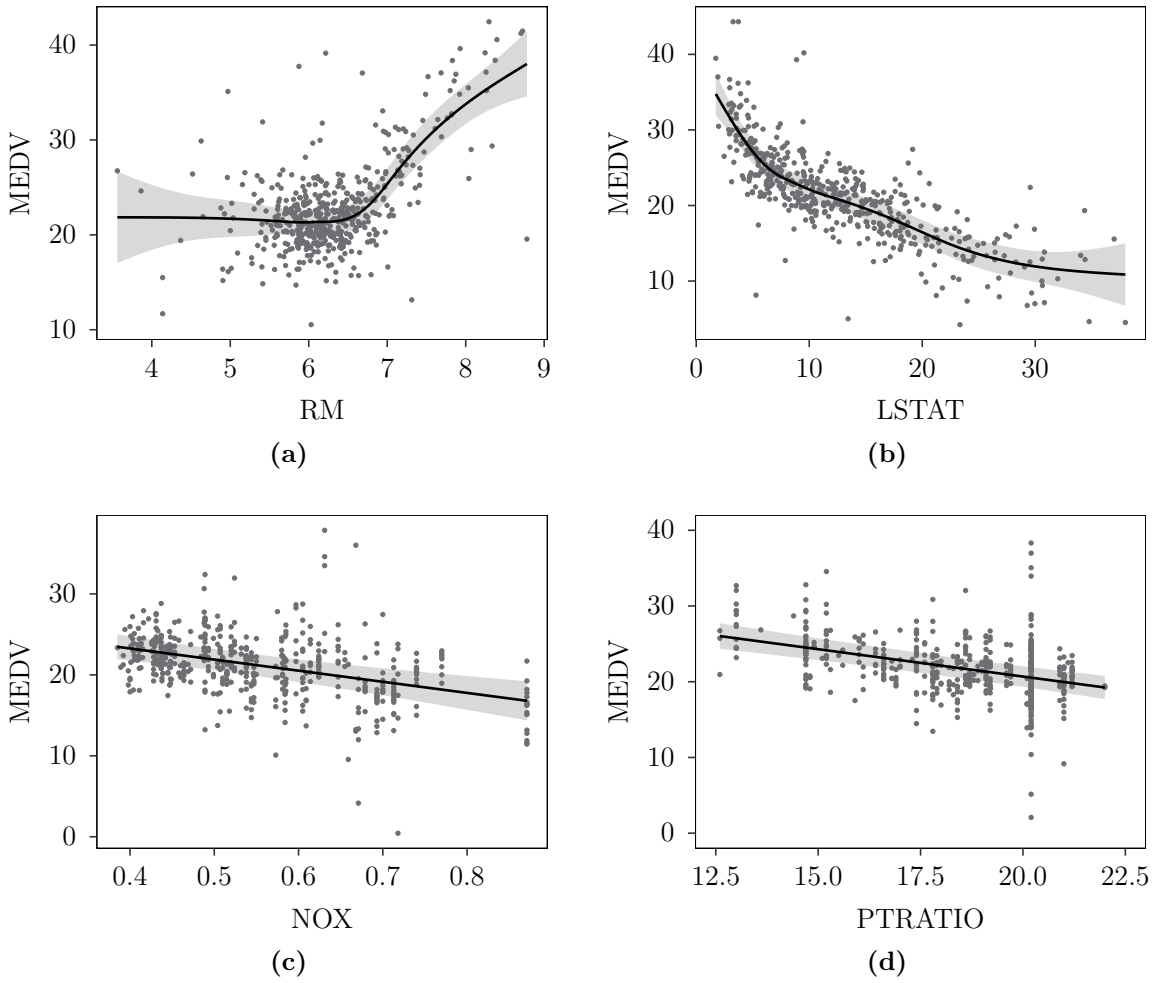

Figure 11: Plots of selected estimated functions in the Boston Housing data set

1 Generate initial population *pop* of 500 models;
2 **for** $g = 1$ **to** *no_gen* **do**
3 Estimate every model in *pop* and sort based on BIC;
4 Initialize $new\_pop = \emptyset$;
5 Place best 10 models from *pop* into *new_pop*;
6 Form random pairs from the best 10 models in *pop*;
7 Apply crossover to the pairs of models formed on line 6, add the results into *new_pop*;
8 Mutate only the nonparametric status of the best 10 models in *pop*, add the results into *new_pop*;
9 Initialize $pre\_mut = \emptyset$;
10 Place the best 240 models from *pop* into *pre_mut*;
11 Randomly select 190 models from the best 240 models in *pop*;
12 Form random pairs from the models selected on line 11;
13 Apply crossover to the pairs of models formed on line 12, add the results into *pre_mut*;
14 Form random pairs from the best 40 models in *pop*;
15 Apply crossover to the pairs of models formed on line 14, add the results into *pre_mut*;
16 Mutate the solutions in *pre_mut*, add the results to *new_pop*;
17 $pop = new\_pop$;
18 **end**
19 Estimate every model in *pop* and sort based on BIC;
20 Return the model from *pop* with the lowest BIC

Algorithm 1: Pseudocode for GAGAM

D Tables

	GAMSEL	GA	GA-R1	GA-R2	GA-R3	GAMSEL	GA	GA-R1	GA-R2	GA-R3
	Low correlation					High correlation				
Low noise										
Zero	18.88	0.78	0.04	0.04	0.78	14.82	0.68	0.02	0.02	0.68
Linear	1.22	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00
% of which zero	0%					0%				
% of which nonlinear	100%					100%				
Nonlinear	0.02	0.02	0.04	0.14	0.16	0.02	0.04	0.10	0.16	0.16
% of which zero	100%	100%	100%	29%	13%	100%	50%	100%	63%	13%
% of which linear	0%	0%	0%	71%	88%	0%	50%	0%	38%	88%
Zero vs. nonzero	18.90	0.80	0.08	0.08	0.80	14.84	0.70	0.12	0.12	0.70
RMSE	5.20	2.64	2.63	2.68	2.68	5.33	2.44	2.44	2.48	2.48
Time (min.)	N.A.		24.62			N.A.		23.89		
Medium noise										
Zero	10.34	0.74	0.04	0.04	0.74	12.42	0.52	0.02	0.02	0.52
Linear	0.66	0.02	0.02	0.00	0.00	1.02	0.00	0.00	0.00	0.00
% of which zero	0%	0%	0%			0%				
% of which nonlinear	100%	100%	100%			100%				
Nonlinear	0.66	0.94	0.96	1.02	1.02	0.34	0.98	1.00	1.00	1.00
% of which zero	52%	81%	100%	94%	75%	76%	82%	98%	98%	80%
% of which linear	48%	19%	0%	6%	25%	24%	18%	2%	2%	20%
Zero vs. nonzero	10.68	1.50	1.00	1.00	1.50	12.68	1.32	1.00	1.00	1.32
RMSE	7.78	6.40	6.36	6.48	6.48	7.25	5.79	5.76	5.88	5.88
Time (min.)	N.A.		19.32			N.A.		19.04		
High noise										
Zero	1.56	0.76	0.04	0.04	0.76	1.08	0.70	0.06	0.06	0.70
Linear	1.20	0.78	1.04	1.04	0.78	1.60	1.04	1.54	1.54	1.04
% of which zero	97%	100%	100%	100%	100%	100%	100%	100%	100%	100%
% of which nonlinear	3%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Nonlinear	2.82	2.90	2.96	2.96	2.96	2.96	2.84	2.92	2.96	2.96
% of which zero	95%	77%	86%	86%	76%	97%	83%	95%	94%	80%
% of which linear	5%	23%	14%	14%	24%	3%	17%	5%	6%	20%
Zero vs. nonzero	4.60	3.78	3.64	3.64	3.78	4.60	4.10	4.38	4.38	4.10
RMSE	25.76	25.02	24.97	25.48	25.49	23.49	22.87	22.88	23.33	23.32
Time (min.)	N.A.		14.30			N.A.		13.72		

Table 1: Results of simulation 1: $n = 400$, $p = 50$, 6 nonzero variables, 3 of which nonlinear.

	GAMSEL	GA	GA-R1	GA-R2	GA-R3	GAMSEL	GA	GA-R1	GA-R2	GA-R3
	Low correlation					High correlation				
Low noise										
Zero	12.80	0.52	0.02	0.02	0.52	12.04	0.38	0.06	0.06	0.38
Linear	8.68	3.00	3.32	0.82	0.46	8.54	3.10	3.64	1.38	0.76
% of which zero	1%	15%	25%	100%	100%	5%	25%	38%	100%	100%
% of which nonlinear	99%	85%	75%	0%	0%	95%	75%	62%	0%	0%
Nonlinear	0.32	1.94	2.20	2.52	2.62	0.30	2.04	2.24	2.40	2.54
% of which zero	38%	74%	92%	80%	55%	67%	85%	96%	90%	69%
% of which linear	62%	26%	8%	20%	45%	33%	15%	4%	10%	31%
Zero vs. nonzero	13.04	2.42	2.86	2.86	2.42	12.64	2.88	3.58	3.58	2.88
RMSE	8.70	6.06	6.09	6.21	6.21	9.20	5.81	5.89	6.00	6.00
Time (min.)	N.A.		76.01			N.A.		78.18		
Medium noise										
Zero	8.64	0.52	0.02	0.02	0.52	8.62	0.42	0.04	0.04	0.42
Linear	6.72	3.80	6.02	4.88	2.50	9.56	5.68	8.68	8.16	5.04
% of which zero	23%	66%	81%	100%	100%	38%	88%	94%	100%	100%
% of which nonlinear	77%	34%	19%	0%	0%	62%	12%	6%	0%	0%
Nonlinear	1.80	3.32	3.64	4.04	4.46	1.66	3.58	3.80	4.06	4.48
% of which zero	50%	78%	88%	80%	59%	61%	82%	89%	83%	66%
% of which linear	50%	22%	12%	20%	41%	39%	18%	11%	17%	34%
Zero vs. nonzero	11.08	5.60	8.12	8.12	5.60	13.24	8.40	11.58	11.58	8.40
RMSE	15.95	14.92	15.50	15.83	15.81	15.44	14.45	15.17	15.47	15.47
Time (min.)	N.A.		93.66			N.A.		85.50		
High noise										
Zero	1.68	0.52	0.02	0.02	0.52	1.14	0.26	0.06	0.06	0.26
Linear	14.42	14.24	17.08	17.08	14.18	16.84	15.88	17.32	17.72	15.88
% of which zero	98%	99%	100%	100%	100%	100%	100%	100%	100%	100%
% of which nonlinear	2%	1%	0%	0%	0%	0%	0%	0%	0%	0%
Nonlinear	4.88	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
% of which zero	90%	93%	99%	99%	93%	82%	90%	95%	95%	90%
% of which linear	10%	7%	1%	1%	7%	18%	10%	5%	5%	10%
Zero vs. nonzero	19.38	19.34	22.06	22.06	19.34	21.06	20.62	22.12	22.12	20.62
RMSE	58.55	57.87	58.72	59.91	59.87	56.47	55.41	55.66	56.70	56.70
Time (min.)	N.A.		17.01			N.A.		13.63		

Table 2: Results of simulation 2: $n = 400$, $p = 50$, 25 nonzero variables, 5 of which nonlinear.

	GAMSEL	GA	GA-R1	GA-R2	GA-R3
	Simulation 3 (high correlation, low noise)				
Zero	17.94	0.62	0.06	0.06	0.62
Linear	2.44	0.00	0.00	0.00	0.00
% of which zero	68%				
% of which nonlinear	32%				
Nonlinear	2.04	0.14	0.20	0.36	0.38
% of which zero	84%	57%	90%	50%	21%
% of which linear	16%	43%	10%	50%	79%
Zero vs. nonzero	21.32	0.70	0.24	0.24	0.70
RMSE	4.67	3.06	3.06	3.07	3.09
Time (min.)	N.A.		30.13		
	Simulation 4 (low correlation, low noise)				
Zero	40.58	12.40	3.24	3.24	12.40
Linear	0.12	0.32	0.32	0.00	0.00
% of which zero	0%	0%	0%		
% of which nonlinear	100%	100%	100%		
Nonlinear	0.76	0.04	0.04	0.08	0.04
% of which zero	34%	0%	0%	0%	0%
% of which linear	66%	100%	100%	100%	100%
Zero vs. nonzero	40.84	12.40	3.24	3.24	12.40
RMSE	5.19	2.93	2.71	2.71	2.93
Time (min.)	N.A.		84.60		
	Simulation 5 (low correlation, low noise)				
Zero	10.36	4.84	2.20	2.20	4.84
Linear	0.62	0.56	0.56	0.12	0.38
% of which zero	0%	0%	0%	0%	0%
% of which nonlinear	100%	100%	100%	100%	100%
Nonlinear	0.68	0.38	0.50	0.72	0.52
% of which zero	74%	68%	92%	64%	50%
% of which linear	26%	32%	8%	36%	50%
Zero vs. nonzero	10.86	5.10	2.66	2.66	5.10
RMSE	8.76	3.51	3.30	3.26	3.47
Time (min.)	N.A.		13.33		

Table 3: Results of simulations 3, 4, 5

Abbreviation	Variable
MEDV	Median value of owner-occupied homes in $1,000s.
CRIM	Per capita crime rate per town
ZN	Proportion of residential land zoned for lots over 25k sq. ft.
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (1 if area bounds river)
NOX	Nitrogen oxides concentration
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted mean of distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000s
PTRATIO	Pupil-teacher ratio by town
BLACK	$1000(\text{Bk} - 0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	Lower status of the population (percent)

Table 4: Variables included in the Boston Housing data set (Harrison & Rubinfeld 1978)

		ARMA	GAMSEL	Best GA	Worst GA
Industrial production	**RMSE**	0.45^{***}	0.45^{**}	**0.41**	0.42
	Time (min.)	N.A.	N.A.	30.54	23.84
Consumer expectations	**RMSE**	4.47^{**}	4.34	**4.23**	4.39
	Time (min.)	N.A.	N.A.	12.08	7.04
Unemployment rate	**RMSE**	0.17^{*}	0.16	**0.16**	0.19
	Time (min.)	N.A.	N.A.	34.47	49.75
Housing starts	**RMSE**	118.87^{*}	110.54	**109.35**	114.96
	Time (min.)	N.A.	N.A.	43.25	36.80
10Y Treasury rate	**RMSE**	**0.31**	0.32	0.32	0.35
	Time (min.)	N.A.	N.A.	24.37	22.77

Table 5: Results of using ARMA, GAMSEL, and the best and the worst runs of GAGAM for forecasting five macroeconomic time series. The lowest RMSE for each method is in bold. Stars denote the p-value from the Diebold-Mariano test comparing ARMA/GAMSEL with the best GA run ($^{*}p < 0.1$;$^{**}p < 0.05$;$^{***}p < 0.01$).