Computational Statistics (2024) 39:461–493
https://doi.org/10.1007/s00180-022-01292-7

ORIGINAL PAPER

Feature selection algorithms in generalized additive models under concurvity

László Kovács[1]

Received: 22 October 2021 / Accepted: 3 October 2022 / Published online: 3 November 2022

Abstract

In this paper, the properties of 10 different feature selection algorithms for generalized additive models (GAMs) are compared on one simulated and two real-world datasets under concurvity. Concurvity can be interpreted as a redundancy in the feature set of a GAM. Like multicollinearity in linear models, concurvity causes unstable parameter estimates in GAMs and makes the marginal effect of features harder interpret. Feature selection algorithms for GAMs can be separated into four clusters: stepwise, boosting, regularization and concurvity controlled methods. Our numerical results show that algorithms with no constraints on concurvity tend to select a large feature set, without significant improvements in predictive performance compared to a more parsimonious feature set. A large feature set is accompanied by harmful concurvity in the proposed models. To tackle the concurvity phenomenon, recent feature selection algorithms such as the mRMR and the HSIC-Lasso incorporated some constraints on concurvity in their objective function. However, these algorithms interpret concurvity as pairwise non-linear relationship between features, so they do not account for the case when a feature can be accurately estimated as a multivariate function of several other features. This is confirmed by our numerical results. Our own solution to the problem, a hybrid genetic–harmony search algorithm (HA) introduces constrains on multivariate concurvity directly. Due to this constraint, the HA proposes a small and not redundant feature set with predictive performance similar to that of models with far more features.

Keywords Generalized additive model · Feature selection · Regularization · Boosting · Genetic algorithm · Harmony search algorithm

✉ László Kovács
laszlo.kovacs2@uni-corvinus.hu

[1] Department of Statistics, Corvinus University of Budapest, Budapest, Hungary

1 Introduction

In supervised machine learning our aim is to predict a well-defined target variable as accurately as possible by utilizing the known values of several feature variables. Nowadays many complex algorithms are available to solve this task. Such as deep learning neural networks, random forests, support vector machines, etc. On the other hand, more and more authors, like Molnar (2020) and Du et al. (2019) draw attention to the fact that those algorithms that provide the most accurate estimates of the target variable are poor at determining marginal effects of the feature variables to the target. However, in certain practical applications, the most important result of supervised learning is not necessarily the accurate estimation of the target, but the discovery of each feature's marginal effect. For example, a bank has to offer a clear reasoning when declining a credit application. In cases like this, our focus should be on building an explainer model, not a predictive one.

In our current bid data environment, when the number of possible features is large, determining marginal effects can be challenging even for a linear regression model. One tool that can be utilized to make supervised learning models more interpretable is feature selection as proposed by Molnar (2020) and James et al. (2013).

The most important principle of feature selection as defined by Hall (1999) is to identify a feature set where each element correlates well with the target, but the features are uncorrelated with each other. In a linear case this means the avoidance of multicollinearity. This principle is similar to the preference of parsimonious models in Econometrics (Wooldridge 2016).

In this paper, we examine the performance of several feature selection algorithms that can be utilized in the context of generalized additive models (GAMs). We examine the case of GAMs as according to James et al. (2013), GAMs represent a balance between model interpretability and prediction accuracy. In the case of GAMs, marginal effect of the features can be determined, and we are not bound by pre-defined linear, logarithmic, squared, or other closed functional forms when representing the non-linear effect of features. However, the model does assume an additive structure, so we should apply features that are uncorrelated in a non-linear sense as well. The non-linear relationship between GAM features called concurvity makes GAM parameter estimates unstable similarly as multicollinearity does for linear models (Ramsay et al. 2003). A detailed description for concurvity is given Sect. 3.

Several feature selection algorithms can be applied in a GAM framework. These algorithms can be separated to four clusters. One is the cluster of the classical stepwise methods. The second cluster is for regularization methods such as the cosso (Lin and Zhang 2006) and the penalized thin plate splines (Marra and Wood 2011). The third cluster contains methods that are utilizing popular boosting techniques, like the GAMBoost algorithm (Schmid and Hothorn 2008) or the modified backfitting procedure (Belitz and Lang 2008). In the fourth cluster, the algorithms are based on the Hilbert–Schmidt independence criterion (HSIC) and aim to avoid selecting correlated features. The best examples are the mRMRe (De Jay et al. 2013) and the block HSIC-Lasso (Climente-González et al. 2019). However, these algorithms only examine pairwise independence of the features, so they cannot tackle a case where one feature can be accurately estimated by the combination of several other features. In other words, the algorithms do not control for multivariate concurvity. We propose a hybrid genetic-improved harmony search algorithm (hybrid algorithm, HA) that applies thin plate splines to produce a best subset feature selector that has a direct constraint on multivariate concurvity in its proposed feature subset. Furthermore, we also apply recursive feature elimination (RFE) combined with a random forest learner as a benchmark algorithm that is not based on GAM learners.

The performance of the algorithms examined in this paper is tested on one simulated and two real world datasets. In a smaller database with 8 features we investigate which non-redundant features are most important in predicting comprehensive strength of concrete girders. This dataset is mainly used to fine tune the parameters of the examined algorithms. Next, a more realistic case with 27 features is used. Here, we investigate which features are most significant in predicting the default of credit card clients. Both real world datasets contain serious multivariate concurvity among its features, but the feature selection algorithms are also evaluated on a simulated dataset, where the concurvity structure is known in advance.

The paper is structured as follows. In Sect. 2, a general mathematical model for GAMs is introduced. In this section, the feature selection task for GAMs is also defined. In Sect. 3, a general introduction into concurvity as a non-linear extension of multicollinearity is given. In Sect. 4, the examined feature selection algorithms are introduced. In Sect. 5, numerical results for the smaller concrete dataset are discussed and the parameter settings of the HA is given. In Sect. 6, the simulated dataset is introduced, and performance of the examined algorithms is evaluated in a known concurvity structure. In Sect. 7, practical issues regarding the implementation of some algorithms in the case of the larger credit card default dataset and numerical results for the same dataset are discussed. In Sect. 8, the conclusions regarding the performance of the examined algorithms are drawn based on the numerical results introduced in Sects. 5–7.

2 Generalized additive models (GAMs)

Let us consider an i.i.d. sample of size n. Let $Y = \left[y_1, y_2, \dots, y_n\right]^T \in \mathbb{R}^n$ be a vector of observed values of a random variable with a distribution from the exponential family. In a GAM, the expected value of Y can be estimated by (1) by utilizing the observed values of $X_j = \left[x_{j1}, x_{j2}, \dots, x_{jn}\right]^T$ feature variables (Hastie and Tibshirani 1990).

$$h(E(Y)) = \sum_{j=1}^{p} f_j\left(X_j\right), \tag{1}$$

where $h(\cdot)$ is a link function for the GAM and $f_j(\cdot)$ is a transformation function for the j th feature. The number of features is p.

The most important task to tackle in (1) is the representations of the $f_j(\cdot)$ transformation functions.

The f_j are represented as the linear combination of spline basis functions (e.g. Wood 2017). This way, (1) can be estimated as a GLM, but overfitting should be avoided. So, in practice, parameters in (1) are estimated by penalized iteratively reweighted least squares (P-IRLS), where the GAM is fitted by solving the minimization of (2). In (2) there is a penalty term that controls for the roughness of f_j s.

$$\min_{\boldsymbol{\beta}}\left(\left\|\sqrt{\boldsymbol{W}}(z-\boldsymbol{B\beta})\right\|^2+\sum_{j=1}^{p}\lambda_j\boldsymbol{\beta}^T\boldsymbol{S_j\beta}\right), \tag{2}$$

here, $z=\boldsymbol{B\beta}+\boldsymbol{G}(Y-\mu)$ and μ is the current model estimate of $E(Y)$ and $\boldsymbol{G}$ is a diagonal matrix such that $G_{ii}=h'(\mu_i)$. μ_i is the current model estimate of $E(y_i)$, where $i=1,2,\ldots,n$. $\boldsymbol{W}$ is a diagonal matrix of $W_{ii}=\left[G_{ii}-V(\mu)\right]^{-1}$, where $V(\mu)$ gives the variance of Y. The columns of $\boldsymbol{B}$ are representing the spline basis functions for the f_j s, while $\boldsymbol{\beta}$ contains all the smooth coefficient vectors, β_j. The $\boldsymbol{S_j}$ are matrices such that the terms in the sum from 1 to p can measure the roughness of the smooth functions. The λ_j s are smoothing parameters that control the trade-off between fit and smoothness and can be selected by restricted maximum likelihood (REML) estimation (Wood 2011). Technically, REML chooses λ_j that minimize the REML criterion, which is essentially an ML applied to the model residuals.

By fitting f_j s through solving problem (2), we choose to represent f_j s as thin plate spline functions that are equivalent with cubic splines in one dimension. The advantage of this representations is that the only hyperparameter to be determined is the maximum number of spline bases to use in each f_j. We denote these values as k_j for the j th feature. The optimal choice for each k_j should be the smallest integer that is large enough so the resulting spline function f_j can capture most of the variance in X_j. Fortunately, there is a statistical test proposed by Augustin et al. (2012) that test the null hypothesis that $Var\left(X_j\right)$ is not significantly different from $Var\left(f_j\left(X_j\right)\right)$. So, we look for the smallest k_j where the null hypothesis of this that cannot be rejected. If we choose a higher k_j value, than it does not cause our model to overfit because of the second term in (2). However, we should not choose too large k_j s as it would make the solution of (2) computationally longer.

Of course, the thin plate or cubic spline representation for f_j s is not the only choice. For example, the R package *gam* (Hastie 2018) utilizes natural splines or local likelihood and the cosso method (Lin and Zhang 2006) represents spline functions in reproducing kernel Hilbert spaces (RKHS). However, the R package *mgcv*, which is one of the most popular R packages for GAMs (Perperoglou et al. 2019) (Lai et al. 2019), builds on the representation introduced in this section. Our hybrid genetic-improved harmony search algorithm, introduced in Sect. 4.2, also builds on the f_j representations of the *mgcv* package.

2.1 Feature selection in GAMs

After obtaining the optimal estimate of the $\boldsymbol{\beta}$ spline coefficient matrix, we can tackle the feature selection problem for GAMs. Given a set of m possible features $X = \{X_1, X_2, \dots, X_m\}$, the task is to select a subset of $\tilde{X} = \{X_1, X_2, \dots, X_p\} \subseteq X$ features, where $p \leq m$, such that the resulting GAM has the best out-of-sample performance. To achieve the best out-of-sample performance, we need to compromise in using in-sample information. If we use too few in-sample information, we will fail to gain a good enough understanding of the relationship between features and the target. However, if we use too much in-sample information, we will overfocus our model and it will have poor out-of-sample performance.

To maximize out-of-sample performance for our GAM, it is advised to apply the principles of parsimony during feature selection. According to this principle, we should select the $\tilde{X} \subseteq X$ subset such as that the GAM we obtain has the best possible fit to Y in-sample, with the application of as few features as possible.

Out-of-sample performance for our model can be achieved in several ways. In our current paper, we utilize the adjusted McFadden pseudo R-squared measure. Description of the measure is based on McFadden (1974). The measure uses the concept of full model, which is a GAM where the number of spline parameters equals to the number of observations, so it provides a perfect estimate for all observations in-sample. Next. we define the deviance for each i observation. It is the gain in log-likelihood if the full model is applied instead of the current one: $D_i = 2\left(\ln L_i(\text{full}) - \ln L_i(\alpha)\right)$. The distribution for the likelihood function L_i is defined by the distribution of the target variable. Full deviance for the model in-sample is $D = \sum_{i=1}^{n} D_i$. Let D_0 be the full deviance for the null model, a model that contains only an intercept. With these notations $R^2 = 1 - \frac{D}{D_0}$. To avoid overfitting, the model in-sample, we should adjust this measure with the number of estimated model parameters ($\sum_{j=1}^{p} k_j$):

$$\overline{R}^2 = 1 - \frac{n-1}{n - \sum_{j=1}^{p} k_j}\left(1 - R^2\right).$$

Overfitting of a GAM can also be avoided by applying a cross-validated pseudo R^2. However, if our aim is to evaluate the out-of-sample performance of several $\tilde{X}$ subsets, then applying the adjusted R^2 is preferred for saving computational time.

During feature selection, $\tilde{X}$ should be selected such that $\overline{R}^2$ is maximal. On the other hand, evaluating every possible subset of X is a NP-hard problem as we have $2^m - 1$ possible subsets to examine, excluding the empty set (Huo and Ni 2007).

Furthermore, by applying thin plate or cubic splines as shown in Sect. 2, we should choose the value of the hyperparameters k_j for each feature. For example, a low initial value for each k_j can be set and increased it until the null hypothesis of the test proposed by Augustin et al. (2012) cannot be rejected. Choosing the initial value for a k_j too large, does not cause serious problems, as the second term in (2) protects us from overfitting f_j. It just causes an increase in computation time for parameter estimation.

3 The issue of concurvity

Several authors have warned that several methods of feature selection (mainly stepwise and regularization methods) can result in a non-consistent feature selection if multicollinearity, or in the non-linear case, concurvity is present in the set of m possible features: Chong and Jun (2005), Zhao and Yu (2006), Signoretto et al. (2008), Jia and Yu (2010), Gu et al. (2010). The lack of consistency in GAM feature selection means that during feature selection features with completely 0 basis function coefficients in the population can be selected into $\tilde{X}$, and features with non-zero basis function coefficients in the population can be estimated with 0 coefficients for all their basis functions (hence excluding the feature from the model). Furthermore, having concurvity among the features in $\tilde{X}$ can result in unstable GAM parameter estimates with variances biased downwards (Ramsay et al. 2003). For these reasons, the issue of concurvity should be addressed during GAM feature selection.

As concurvity can be considered as the non-linear extension of multicollinearity, first we examine multicollinearity more formally. Multicollinearity occurs when one of the X_j features in a linear model can be accurately approximated as a linear combination of the other $X_{c\neq j}$ features in the model.

$$X_j \approx \sum_{c\neq j} \alpha_c X_c,$$

if $\sum_{c\neq j} \alpha_c X_c$ can explain at least 50% from the variance of X_j, we consider the X_j a redundant feature in the linear model. In other words, X_j has harmful multicollinearity. (Mansfield and Helms 1982).

For defining concurvity as a non-linear extension of multicollinearity, we rely on the notations given in (2). Let $\boldsymbol{B_j}$ consist of the columns in the $\boldsymbol{B}$ matrix that are representing the spline basis functions for f_j. This way, $\boldsymbol{B_j}$ is of size $n \times k_j$ and β_j is the k_j dimensional coefficient vector for $\boldsymbol{B_j}$. This way, we have $f_j\left(X_j\right) = \boldsymbol{B_j}\beta_j$. If f_j can be accurately approximated by the linear combination $\boldsymbol{B_{c\neq j}}$ spline basis functions of other $X_{c\neq j}$ features, we have a concurvity problem in the feature X_j (3).

$$f_j\left(X_j\right) \approx \sum_{c\neq j} \boldsymbol{B_c}\alpha_c, \tag{3}$$

this definition of concurvity is already present in Buja et al. (1989) and in Hastie and Tibshirani (1990) with its possible effects on parameter estimation (upwardly biased parameters and downwardly biased variances). Furthermore, this redundancy between features makes interpreting feature effects on the target difficult since the effects are not clearly separable. Ramsay et al. (2003) gives practical examples of these harmful consequences in epidemiology.

Early measures and methods to measure concurvity in GAMs mainly focus on the pairwise non-linear relationships. Gu et al. (2010) applies mutual information measure and Amodio et al. (2014) utilize maximal local correlation statistics to detect the degree of concurvity. Both approaches consider the non-linear dependence of only two variables. The algorithms that attempt to control for concurvity in feature selection (mRMRe and HSIC-Lasso) also rely on concurvity measures that consider only pairwise relationships, namely the Hilbert–Schmidt independence criterion proposed by Gretton et al. (2005). These algorithms are discussed in detail in Sect. 4.

Wood (2017) proposes three indices of concurvity in GAMs built on the definition of concurvity given in (3). This way it can detect multivariate concurvity as well, not just pairwise relationships between features. All three indices are bounded between 0 and 1, with 0 indicating no problem, and 1 indicating total lack of identifiability for GAM parameter estimation.

The three indices are all based on the idea that a smooth function, f_j, in the model can be decomposed into a part, g_j, that is part of the basis spline representation of some other smooth function $f_{c \neq j}$ in the model. So, g_j is a part of f_j that can be expressed as a linear combination of some $\boldsymbol{B}_{\boldsymbol{c} \neq \boldsymbol{j}}$ spline basis functions. So, let $g_j = \sum_{c \neq j} \boldsymbol{B}_{\boldsymbol{c}} \alpha_c$ and let $\varepsilon_j = f_j(X_j) - g_j$. This way, we have $f_j(X_j) = g_j + \varepsilon_j$.

If g_j makes up a large part of f_j then there is a concurvity problem. Wood's three indices used are all based on $\frac{||g_j||}{||f_j||}$, that is the ratio of the simple Euclidean norms of f_j and g_j evaluated at the observed values of X_j.

The three measures differ in their approach to the α_c vectors in g_j:

- *Worst* In this measure the α_c vectors are given so that they maximize $\frac{||g_j||}{||f_j||}$. So, this is a pessimistic measure, as it looks at the worst case, without considering the data.
- *Observed* Here, we take the β_c coefficient vectors estimated during the GAM fitting for the α_c vectors and compute $\frac{||g_j||}{||f_j||}$ accordingly.
- *Estimate* This is the matrix F-norm of the basis for g_j ($\boldsymbol{B}_{\boldsymbol{c}}$) divided by the same norm of the basis for f_j ($\boldsymbol{B}_{\boldsymbol{j}}$). It is a measure of the extent to which the basis of f_j can be explained by the bases of the other features regardless of the α_c coefficient vectors.

These three indices can be computed for each X_j feature in the current model and the extent of concurvity can be given similarly as for multicollinearity in the linear case.

In our current paper, if the chosen concurvity measure is above 0.5 for a X_j feature, then we consider that feature redundant (has harmful concurvity). This cut-off is analogous to the cut-off for harmful multicollinearity in the linear case. There, at least half of the variance in X_j have to be explained by the rest of the features in the current model to have harmful multicollinearity.

4 Feature selection algorithms in GAM framework

The feature selection algorithms investigated in our current paper were selected based on the number of Google Scholar references for the scientific paper in which they were first introduced, and the availability of implementations in R or Python packages. Detailed description for each algorithm is not given in this paper, but reference of their original paper is always provided.

The algorithms that are investigated can be partitioned into four clusters. A summary of these clusters and the algorithms is given in Table 1.

The first cluster of algorithms consists of the classical stepwise methods first proposed by Efroymson (1960). In each step, a feature is considered for addition to or subtraction from the set of all possible features (X) based on some prespecified criterion. The criterion applied in the current paper is the $\overline{R}^2$ as defined in Sect. 2.1.

The second cluster is a family of algorithms that utilize boosting techniques, commonly applied in classifier algorithms, for GAM feature selection. Boosting was originally developed in the machine learning community as a means to improve classification procedures (e.g., Schapire 1990). The basic concept to use a classifier iteratively with differing weights on the observations and combine the results in a committee voting has been shown to reduce the misclassification error drastically. In Tutz et al. (2006), Binder and Tutz (2008) and Schmid and Hothorn (2008), it has been shown that boosting is a way of fitting an additive expansion in basis functions.

The third cluster is for the regularization methods. These procedures are the generalization of the Lasso algorithm proposed by Tibshirani (1996) to the non-linear case of GAMs. The basic concept of these algorithms is to apply extra regularization conditions, such as the L_1 norm of the basis function coefficients, during the coefficient estimation process (like the solution of (2) in the case of thin plate splines). These extra regularization conditions can result in completely zero coefficients for all basis functions of a X_j feature. Hence executing feature selection.

In the thin plate spline framework, regularization can be applied in two ways as proposed by Marra and Wood (2011).

Proposed by Lin and Zhang (2006), the component selection and smoothing operator (cosso) executes GAM feature selection in a reproducing kernel Hilbert space (RKHS) by applying the L_1 penalty of model parameters during parameter estimation like the Lasso method does in the linear case.

To identify the important smooth components of an additive model, Cantoni et al. (2011) suggest employing the Non-negative garrote estimator. The idea behind this is as follows. In a first step we obtain the original regression coefficient or smooth function estimates, depending on whether we are in a parametric or nonparametric context. We then shrink the model components by solving a constrained optimization problem.

It is important to note that none of the feature selection algorithms in the first three clusters have attempted to avoid concurvity in their final selected model.

To tackle the inconsistency in the feature selection process caused by the presence of concurvity among the possible features, we should examine a fourth cluster of feature selection algorithms. The algorithms in the fourth cluster are using the mutual information of the target variable and the features to execute feature selection. They are special in a way that they do not assume special model structure (like GAMs, or support vector machines, etc.) for feature selection. The most popular algorithms in this cluster are the mRMR (De Jay et al. 2013) and the block HSIC-Lasso (Climente-González et al. 2019).

Since our paper focuses on feature selection in datasets with concurvity, these two algorithms are described in a bit more detail in Sect. 4.1.

Table 1 Summary of the feature selection algorithms investigated in this paper

Cluster	Algorithm	Original paper	Software implementation
Stepwise	Forward selection	Efroymson (1960)	R package *gam* (Hastie 2018)
	Backward elimination		
	Bidirectional elimination		
Boosting	GAMBoost	Binder and Tutz (2008) and Schmid and Hothorn (2008)	R package *GAMBoost* (Binder and Tutz 2008)
	Modified backfitting	Belitz and Lang (2008)	R package a *R2BayesX* (Umaluf et al. 2015)
Regularization methods	Double penalty approach	Marra and Wood (2011)	R package *mgcv* (Wood 2017)
	Shrinkage approach		
	Cosso method	Lin and Zhang (2006)	R package *cosso* (Zhang and Lin 2013)
	Non-negative garrote component selection	Cantoni et al.(2011)	Own R script bases on the *ncvreg* package (Breheny and Huang 2011)
HSIC based methods	mRMR	De Jay et al. (2013)	R package *mRMRe* (De Jay et al. 2013)
	HSIC-Lasso	Climente-González et al. (2019)	Python package *pyHSICLasso* (Climente-González et al. 2019)

4.1 Feature selection algorithms with some control to concurvity

Both the mRMR and the HSIC-Lasso algorithms apply the Hilbert–Schmidt independence criterion (HSIC) in order to measure the mutual information of two random variables x and z. The HSIC is introduced by Gretton et al. (2005) and defined as in (4).

$$\begin{aligned} HSIC(x,z) = E_{x,x',z,z'}\left[K\left(x,x'\right)L\left(z,z'\right)\right] + E_{x,x'}\left[K\left(x,x'\right)\right]\mathbb{E}_{z,z'}\left[L\left(z,z'\right)\right] \\ - 2E_{x,z}\left[E_{x'}\left[K\left(x,x'\right)\right]E_{z'}\left[L\left(z,z'\right)\right]\right], \end{aligned} \tag{4}$$

where K and L $\mathbb{R}^{n\times n} \to \mathbb{R}$ are positive definite kernel functions. $E_{x,x',z,z'}$ is the expected value of the independently selected random pairs (x,z) and $\left(x',z'\right)$ from the joint distribution of x and z. If x and z are independent and otherwise positive. By examining (7), we can conclude that the HSIC measures id dependence of two random variables based on their joint distribution, hence accounting for non-linear relationships as well. In practical implementations Gauss kernel is used for continuous features: $K(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$. Where $\sigma = 1$ as the features are normalized during pre-processing. For categorical features, a Delta kernel is applied: $K(x,y) = 1/n_y, if \quad x = y; 0 \quad$ otherwise. Where n_y is the number of y values in the observed sample.

The mRMR (minimum Redundancy–Maximum Relevance) algorithms is a best subset feature selector that aims to select those X_j features to $\tilde{X} \subseteq X$, where $HSIC\left(Y,X_j\right)$ is high, but $HSIC\left(X_j,X_k\right)$ is low for $\forall k \neq j$ where is already in $\tilde{X}$. This principle for feature selection is implemented in a simple linear search in the mRMR. We a priori define p, which is the number of features in $\tilde{X}$. Then, we start from the empty set for $\tilde{X}$. In every iteration from 1 to p, we add a feature X_j to $\tilde{X}$ such that $j = \arg\max_{k\in X\setminus\tilde{X}} HSIC\left(Y,X_k\right)/\frac{1}{|\tilde{X}|}\sum_{i\in\tilde{X}} HSIC\left(X_i,X_k\right)$. We can see that our aim function increases if the mutual information between X_k and the target is high, but the aim function decreases if X_k has high mutual information with other X_i features.

The mRMR algorithm is implemented in the *mRMRe* R package, where the initial value for p is 6.

The HSIC-Lasso algorithm aims to solve the Lasso optimization task of (5) by assigning a linear coefficient γ_j for each $HSIC\left(X_j,Y\right)$.

$$\max_{\Gamma\geq 0} \sum_{j=1}^{m} \gamma_j HSIC\left(X_j,Y\right) - \frac{1}{2}\sum_{j,k=1}^{m} \gamma_j\gamma_k HSIC\left(X_j,X_k\right) - \lambda\Gamma_1, \tag{5}$$

where $\lambda > 0$ is a tuning parameter selected via cross validation and $\Gamma = \left[\gamma_0, \gamma_1, \dots, \gamma_m\right]$. Because of the L_1 penalty for the coefficient vector Γ, the features to be removed will have an estimated γ_j of 0. In (5) we also have a penalty term for high $HSIC\left(X_j,X_k\right)j \neq k$ values, which helps to avoid selecting redundant features. The main advantage of solving (5) instead of applying the mRMR algorithm, is that it does not require an a priori knowledge for the size of $\tilde{X}$.

Solving (5) can be rather memory intensive. To tackle this problem, Climente-González et al. (2019) proposed computing the HSIC measures on some $B \ll n$ sized blocks of the observed sample separately, then summing the HSIC measures of each block, we can obtain a HSIC for the whole sample. By utilizing these block-by-block calculations for the HSICs, the HSIC-Lasso algorithm can be effectively applied on larger datasets.

The HSIC-Lasso algorithm does not currently have an R implementation, so a Python implementation is applied in the online IDE, *replit.com*.

It is important to note that both the mRMR and HSIC-Lasso algorithms attempt to penalize for concurvity in their aim functions. However, the nature of the HSIC measure means that these algorithms can only penalize the relationship of feature pairs. The methods cannot account for concurvity that arises if a feature can be approximated by the non-linear combinations of several other features. This means that final models proposed by the mRMR and HSIC-Lasso algorithms can still have a harmful degree of concurvity which makes it harder to separate feature effects to the target.

4.2 The hybrid genetic–harmony search algorithm for GAM feature selection

We propose a hybrid genetic-improved harmony search algorithm (hybrid algorithm, HA) that applies thin plate splines to produce a best subset feature selector that is capable to find concurvity-free models. Our HA solution can tackle a multivariate case of concurvity which is not accounted for by the mRMR and HSIC-Lasso algorithms detailed in Sect. 4.1. The HA was introduced in Láng et al. (2017), but only for linear models.

The flow chart of the algorithm is given in Fig. 1. The objective function in our current paper is the $\overline{R}^2$.

The HA represents the possible solutions for $\tilde{X}$ as a binary vector of length m. So, the only decision point is whether to include a feature in the model or not. The HA is only applicable in a GAM framework if this does not change. Fortunately, if we apply thin plate splines for representing the f_j smooth functions, we can keep the binary decision point of the HA.

In the algorithm, we apply the following procedure. If a X_j feature is selected in the current GAM, then by default $k_j = 10$ as proposed by Wood (2017). If the value set of X_j is smaller than 10, then k_j equals to the number of unique values in X_j. If the p-value for the hypothesis test defined in Augustin et al. (2012) is below $\alpha = 0.01$, then k_j is increased by 5 until the null hypothesis can be accepted at $\alpha = 0.01$. This way we try to avoid more basis functions than necessary. Although too many basis functions do not cause an overfit because of the roughness penalty in (2). They can only be a small computational nuisance for one model, but since the HA is a best subset feature selector, these nuisances can add up and we try to avoid that as much as possible.

The most important advantage of the HA is that during feature selection it aims to eliminate all forms of concurvity from the models. This means that during a memory update, we only transfer solutions to the new memory where the concurvity

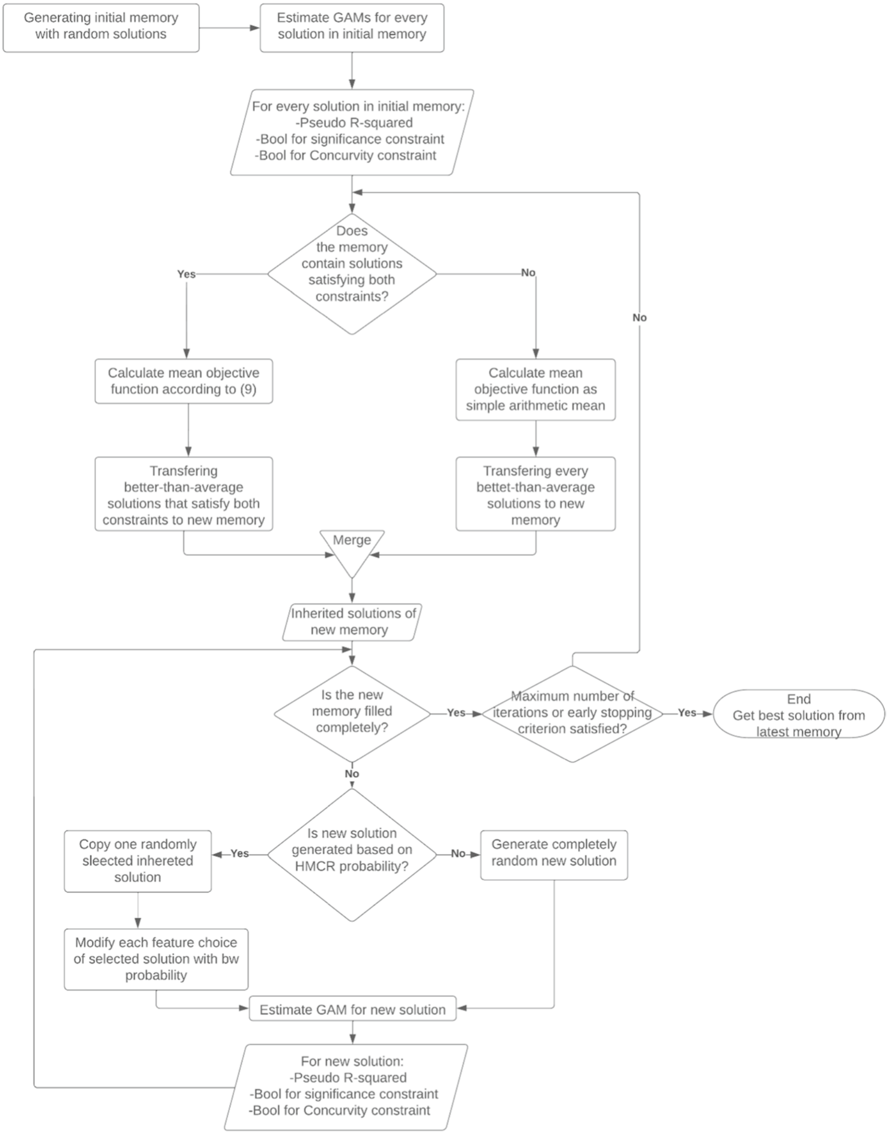
Generating initial memory with random solutions
Estimate GAMs for every solution in initial memory
For every solution in initial memory: -Pseudo R-squared -Bool for significance constraint -Bool for Concurvity constraint
Does the memory contain solutions satisfying both constraints?
Yes
No
No
Calculate mean objective function according to (9)
Calculate mean objective function as simple arithmetic mean
Transfering better-than-average solutions that satisfy both constraints to new memory
Transfering every bettet-than-average solutions to new memory
Merge
Inherited solutions of new memory
Is the new memory filled completely?
Yes
Maximum number of iterations or early stopping criterion satisfied?
Yes
End Get best solution from latest memory
No
Is new solution generated based on HMCR probability?
Yes
No
Copy one randomly sleected inhereted solution
Generate completely random new solution
Modify each feature choice of selected solution with bw probability
Estimate GAM for new solution
For new solution: -Pseudo R-squared -Bool for significance constraint -Bool for Concurvity constraint

Fig. 1 Flow chart for the Hybrid Genetical–Harmony Search Algorithm

measure defined in Sect. 3 is below 0,5 for all features. The HA can be parametrized in order to decide which concurvity measure to use: worst, observed or estimate. If the i th solution in a memory satisfies this constraint, then a binary variable, C_i takes a value of 1, and 0 otherwise.

Furthermore, it is also important that only those features should be preferred, where we can reject the null hypothesis that all its basis function coefficients are

0 ($\beta_j = 0$). Testing of this null hypothesis can be solved by a χ^2 test proposed by Marra and Wood (2011). If at a user defined significance level, the $\beta_j = 0$ null hypothesis can be rejected for all features in the i th solution in a memory, then a S_i binary variable takes a value of 1, and 0 otherwise.

Technically, incorporating these two conditions is solved in the memory update step of the HA. During this phase, when selecting the better than average solutions from the memory, the average objective function of the current memory is defined by (6). We only accept the i th solution in the memory as a better than average solution if it satisfies the condition $C_i = S_i = 1$.

$$\overline{R}_M^2 = \frac{\sum_{i=1}^{N} \overline{R}_i^2 \cdot C_i \cdot S_i}{\sum_{i=1}^{N} C_i \cdot S_i}, \tag{6}$$

where N is the memory size, $\overline{R}_i^2$ is the objective function of the i th solution in the memory. If $\sum\limits_{i=1}^{N} C_i \cdot S_i = 0$, then $\overline{R}_M^2$ is simply the arithmetic mean of $\overline{R}_i^2$ s, and every solution in the current memory can be accepted as a better than average solution in the memory.

It is important to notice that the $C_i and S_i$ constraints in the algorithm can cause that we will have no solutions in the initial random memory where $C_i = S_i = 1$. So, it can take several iterations to find solutions that satisfy every constraint. This implies that expected runtime of HA depends on the quality of the initial random memory.

Because of this sensitivity to the random initial memory, we should increase the variability of the initial memory. We solve this increase by borrowing an idea from the k-means clustering algorithm (Hartigan and Wong 1979). We choose a smaller memory size, which ensures a shot expected runtime, and run the algorithm several times with a small maximal generation number. This way several initial random memories can be tested, ensuring the desired variability.

For larger datasets parallelization of the HA is possible. This is done by solving (2) for each solution in the current memory simultaneously. This can be easily realized as the genetic algorithm part of the HA makes generation of solutions for the new memory easy to solve in parallel. In R the *foreach* (Weston 2019a) and *doParallel* (Weston 2019b) packages make the implementation of simultaneous parameter estimation for the solutions in the new memory easy to implement. We apply this parallelization technique in the numerical experiments.

5 Numerical results on the concrete compressive strength dataset

Our first examined real world dataset is proposed by Yeh (1998) and contains 9 variables of 1030 concrete girders. The task is to estimate the comprehensive strength of concrete material as a non-linear function of age and ingredients. As $m = 8$, the feature selection task is small in size, which means all of the possible subsets can be generated and the global optima is easily selected. The aim of our investigation for this dataset is to examine how effectively can the examined selection algorithms identify the known global optima.

Variables in the database are the following:

- Cement—in kg/m^3
- Blast Furnace Slag—in kg/m^3
- Fly Ash—in kg/m^3
- Water—in kg/m^3
- Superplasticizer—in kg/m^3
- Coarse Aggregate—in kg/m^3
- Fine Aggregate—in kg/m^3
- Age—days since manufacturing
- Compressive Strength—in MPa (target variable)

Some girders needed to be excluded from the numerical experiments as they proved too old. This way, we had 916 observations left. This dataset is randomly split into training and test sets in a 7:3 ratio. Cross validations steps of the examined algorithms are executed on the training set.

In this dataset the target is continuous and normally distributed (Yeh 1998), so the link function for the GAMs is the identity. Predictive performance of the final models from the algorithms is measured by the $R^2 = \text{cor}\left(Y, \hat{Y}\right)^2$ on the test set.

Global optimum is known for this dataset. From all possible feature subsets where $C_i = S_i = 1$, maximal R^2 is provided by the *Cement* + *Blast Furnace Slag* + *Water* + *Age* model.

Every numerical experiment and the data cleansing steps are executed in R version 3.5.3. with the help of the packages mentioned in Sect. 4. The only exception is the HSIC-Lasso algorithm which is run in Python as described in Sect. 4.1. For the mRMR algorithm, we exploit the a priori knowledge that the number of features in the optimal subset is $p = 4$.

For the HA, the chosen significance level for testing the $\beta_j = 0$ null hypothesis for each feature is 5%. Memory size is 15 and maximal number of iterations is also 15. Early stopping is possible if the objective function for the best solution in the memory does not change for 5 iterations. The reason for these parameter choices comes from the fact that the number of all feature subsets is $2^8 - 1 = 255$. So, in the worst case we examine $15 \times 15 = 225$ separate models that is still fewer than the total number of all subsets. The rest of the parameters are optimised numerically. We optimise the parameters shown in Table 1 in a ceteris paribus way. We measure the efficiency of a parameter setting by running the HA 30 times and see how many times the algorithm identifies the known global optima or the second-best solution.[1] Based on these numerical experiments, the optimal parameters of the HA are given in Table 2.

[1] In the second-best solution $R^2 = 81.56\%$ and the maximal concurvity measure is 0.461. Compared with the rest of the models proposed by other algorithms, this solution is still acceptable: R^2 is not significantly lower and there is no violation of concurvity constraints.

Table 2 Optimal parameters for the HA

Parameter	Optimal value (%)
Initial *HMCR* probability	5
Initial mutation (*bw*) probability	90
Maximal *HMCR* probability	35
Minimal mutation (*bw*) probability	10

Table 1 shows that the HA prefers the random generation of new solutions, but inheritance from the previous population should not be neglected: the *HMCR* should be increased from 5 to 35% during the iterations. Decreasing the mutation *bw* probability from 90 to 10% during the HA iterations also supports the idea.

As several of the examined algorithms have a heuristic nature, all algorithms are run 30 times to examine the stability of the final models for each algorithm, the expected runtimes, and the standard deviation of the runtimes. In the case of those algorithms, where the results are not the same for all 30 trials, the best model is shown in the results, so every algorithm is represented with its best performance. Each algorithm is run on the same training set and evaluated on the same test set for each trial. This way, we try to ensure the differences we see in performance are due to the different characteristics of each algorithm.

5.1 Discussion of numerical results

The results of the algorithms described in Sect. 4 for the concrete compressive strength dataset are summarised in Table 3.

The results of HA shown in Table 3 is found by 19/30 times and it is the global optimum for the feature selection task when the constraints for basis function significance and concurvity are considered.

Most of the examined feature selection algorithms prefer the full model or eliminate only one feature. As a result of this, every feature in the final model of these algorithms violates the concurvity constraints, except for Age. The cosso, modified backfitting and HSIC-Lasso algorithms are preferring models with less features. There is no relevant difference in the predictive performance of these algorithms. Only the performance of the non-negative garrote is somewhat worse than the rest. This can be a consequence of the fact that this algorithm estimates all the smooth functions (f_j s) as b-splines before executing feature selection as a Lasso task.

The mRMR and the HA models are the only ones with a smaller and concurvity-free feature set of only 4 features (out of 8) with predictive performance on the test set that is not significantly different than the rest of the examined algorithms. This suggests that most of the feature selectors tend to pick up noise features in their final models.

Table 3 Numerical Results of the Examined Algorithms on the concrete comprehensive strength dataset

Algorithm	Selected features	Features that violate the concurvity constraints	R^2(test Set) (%)	Expected runtime (sec)	St. D. of runtimes (sec)
Full model	Full model	All, except Age	88.12	2.663	0.046
Cosso	Cement, BlastFurnaceSlag, FlyAsh, Water, Age	All, except Age	85.05	3.983	0.515
Thin plate spline with double penalty	Full model	All, except Age	88.08	9.902	0.253
Thin plate spline with shrinkage	Full model	All, except Age	87.69	3.313	0.114
Non-negative garrote	Full model	All, except Age	78.20	0.270	0.120
Stepwise	Full model	All, except Age	88.12	2.794	0.064
GAMBoost	Full model	All, except Age	89.24	507.104	18.506
Modified backfitting	Cement, BlastFurnaceSlag, Superplasticizer, CoarseAggregate, Age	Cement, Superplasticizer	87.95	0.835	0.527
mRMR	Cement, BlastFurnaceSlag, Superplasticizer, Age	–	81.58	0.068	0.037
HSIC-Lasso	Cement, BlastFurnaceSlag, Water, Superplasticizer, Age	Water Superplasticizer	86.38	0.404	0.095
Hybrid algorithm	*Cement, Blast Furnace Slag, Water, Age*	–	*84.36*	*111.026*	*17.460*

Italics mark the results of the algorithm that's our own development and not one implemented from literature

Detailed examination of the modified backfitting and HSIC-Lasso models can highlight what causes the harmful concurvity between the chemical components of the concrete girders. In the HSIC-Lasso model, concurvity constraints are violated by the Water and Superplasticizer variables. While in the Modified backfitting model, the Cement and Superplasticizer features do the same. The reason behind this phenomenon is that the amount of a superplasticizer applied in concrete manufacturing is the function of the water/cement ratio. So, the Superplasticizer feature can be accurately approximated by the Cement and Water features. This phenomenon is disregarded even by the HSIC-Lasso algorithm as it only controls for pairwise relationship between the features during feature selection.

The HA controls for concurvity directly. Therefore, it can ensure that the selected feature set is completely free from concurvity. In this current dataset the mRMR model is also free of concurvity, but we utilized the a priori knowledge that $p = 4$. Otherwise, mRMR is only capable of handling pairwise relationship of features, similarly to the HSIC-Lasso. So, with some a priori information, the mRMR can provide an acceptable solution along with the HA: predictive performance not significantly different from other algorithms with more parsimonious and concurvity-free feature set.

The greatest advantage of the HA model is that the marginal effects of the selected features are interpretable based on the fitted smooth functions as they are completely free from concurvity. An interpretation like this can be done from the mRMR model as well in this dataset since we had the a priori information of $p = 4$. However, poorer, though not significantly poorer predictive performance on the test set could mean that the mRMR did not identify the true features of comprehensive strength. This is something that can be confirmed with experiments on a simulated dataset in Sect. 6.

Based on the HA model we can state that comprehensive strength is mainly influenced by their cement, blast furnace slag, water contents and their age. On Fig. 2, we can observe that the cement contents are almost linearly influencing the comprehensive strength. In the case of blast furnace slag and water, we can determine a global level that maximizes comprehensive strength: about 250 kg/m^3 for furnace slag and about 140 kg/m^3 for water. We can also observe that higher age causes comprehensive strength to increase on a logarithmic scale: after about 55 days, comprehensive strength does not increase significantly.

Figure 2 also shows some wide confidence bands at the higher values of some features. This suggests that these higher sparse values might be too influential in GAM estimation. We applied Cook's distance to see the influence of each observation and found that we have no Cook's distance greater than 1. This indicates no influential observations for parameter estimation (Altman and Krzywinski 2016).

Expected runtime for the HA is almost 2 min compared to the 1–2 s runtimes for the other methods. The reason for this is that the HA examines several subsets of features and needs to estimate a GAM for each subset. It is a significantly slower process, even if it is done simultaneously than doing a stepwise search that examines much less feature subsets for example. The HSIC-Lasso and the mRMR algorithms are not even estimating a GAM during feature selection, so expected runtime for the HA is noticeably slower compared to these methods. However, the HA is

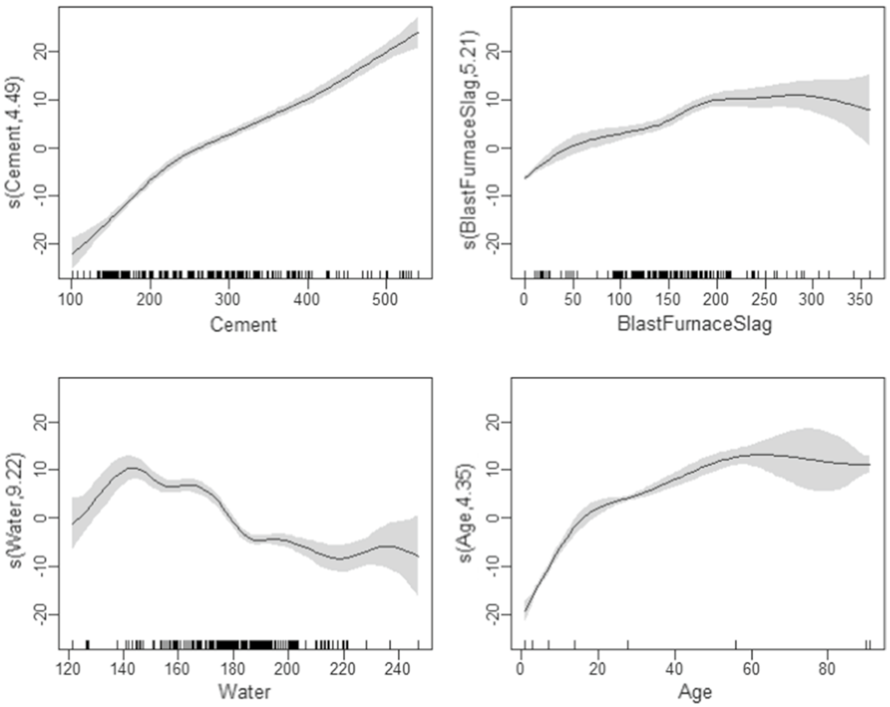

Fig. 2 The fitted smooth functions in the HA model for the concrete comprehensive strength dataset with 95% confidence-intervals

significantly better with its 2-min expected runtime than the GAMBoost algorithm with its 8-min expected runtime.

All in all, we can conclude that if our aim is to build more of an explainer model without concurvity influencing the results, then the HA is a preferred method to the HSIC-Lasso and mRMR models if time is not a scarce resource and we do not have a priori information for the mRMR. The rest of the algorithms propose a more extended feature set without significant improvements in predictive performance on the test set.

6 Numerical results on simulated dataset

As a next step, numerical results on a simulated dataset of 1000 observations and 7 feature variables with a Gaussian target are presented. In this example we aim to demonstrate how well each examined selection algorithm control for a known concurvity structure.

The definition of our 8 variables is the following with $\sigma_1 = 0.05$ and $\sigma_2 = 0.5$.

$$X_1 \sim X_2 \sim X_3 \sim U(0, 1)$$

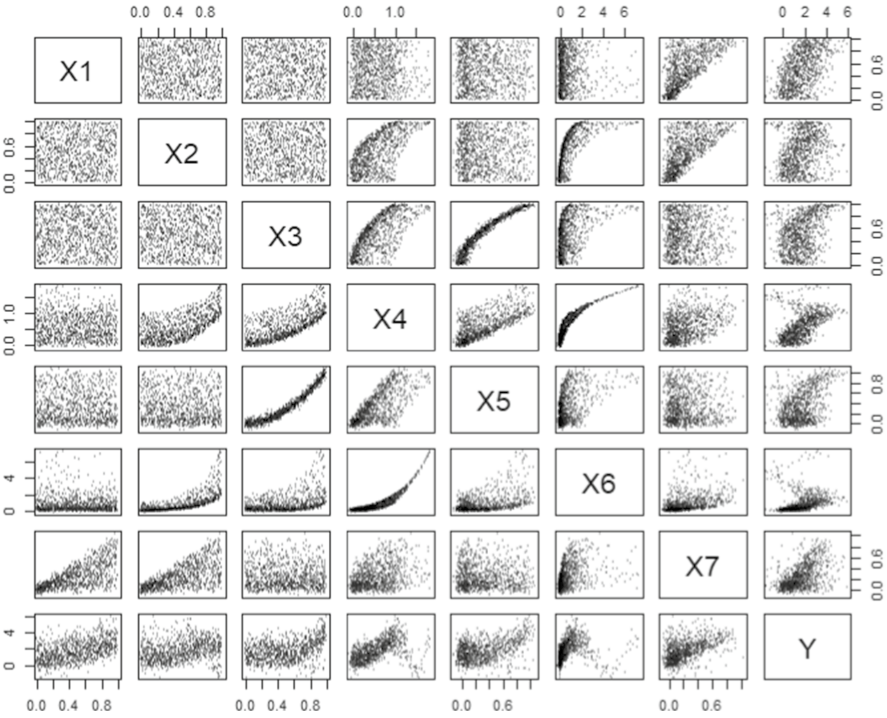

Fig. 3 Scatterplot matrix of the simulated variables

$$X_4 = X_2^3 + X_3^2 + N(0, \sigma_1)$$

$$X_5 = X_3^2 + N(0, \sigma_1)$$

$$X_6 = X_2^2 + X_4^3 + N(0, \sigma_1)$$

$$X_7 = X_1 \times X_2 + N(0, \sigma_1)$$

$$Y = 2X_1^2 + X_5^3 + 2\sin X_6 + N(0, \sigma_2)$$

With the definitions above, we have generated severe concurvity among the features (as σ_1 is relatively low), while only X_1, X_5 and X_6 have direct effect on the target Y. This concurvity is visible on the scatterplot matrix of the simulated dataset given in Fig. 3.

As $m = 7$, the feature selection task is small, which means all the possible subsets can be generated and the global optima is easily selected. The main aim of this investigation is to find out how well can each examined feature selection algorithm detects the concurvity that we know is present among the features and identify the optimal feature subset of $\left\{X_1, X_5, X_6\right\}$.

The simulated dataset is randomly split into training and test sets in a 7:3 ratio. Cross validations steps of the examined algorithms are executed on the training set.

The target is continuous, so the link function for the GAMs is the identity. Predictive performance of the final models from the algorithms is measured by the $R^2 = \mathrm{cor}\left(Y, \hat{Y}\right)^2$ on the test set. The HA parameters are the same as the optimal settings for the concrete comprehensive strength dataset given in Table 1. Similarly, all algorithms are run 30 time, so, the expected runtimes and the standard deviation of the runtimes is given. In the case of those algorithms, where the results are not the same for all 30 trials, the best model is shown in the results, so every algorithm is represented with its best performance. Each algorithm is run on the same training set and evaluated on the same test set for each trial. This way, we try to ensure the differences we see in performance are due to the different characteristics of each algorithm.

6.1 Discussion of numerical results

A summary on the performance of each feature selection algorithm is given in Table 4.

The results of HA shown in Table 4 is found by 21/30 times and it is the global optimum for the feature selection task when the constraints for basis function significance and concurvity are considered.

The picture is very similar to what was experienced in the case of the concrete comprehensive strength dataset. Most of the examined feature selection algorithms prefer the full model, or only exclude one feature. These models do not differ in test set performance significantly. One exception is the GAMBoost that has noticeably poorer performance compared to the other model preferring the full model or a model close to it. On the other hand, the performance of the non-negative garrote has improved quite well compared to its performance of the concrete dataset. So, estimating the smooth functions before a Lasso selection seems beneficial in cases of complex concurvity from the perspective of predictive performance. However, since almost all models have quite similar predictive performances, the main differences are again in the selected feature sets.

All the algorithms preferring something close to the full model have harmful concurvity in all their features. One exception is the Modified backfitting where X_7 is omitted that is partly a function of X_1. All in all, these methods select models with convincing predictive performance that do not differ significantly, but they use almost all available feature to achieve this, not revealing the true predictors of the target. They result in good predictive models but poor explanatory models since they pick up features that are essentially just noise in respect to our target variable. This is similar to what we experienced in the results on the concrete dataset.

The cosso and stepwise methods retain the same 4 features out of 7, with predictive performance that is essentially the same as for the methods that prefer something close to the full model. However, they still retain the X_4 feature that has no

Table 4 Numerical Results of the Examined Algorithms on a Simulated Dataset

Algorithm	Selected Features	Features that Violate the Concurvity Constraints	R^2(Test Set) (%)	Expected Runtime (sec)	St. D. of Runtimes (sec)
Full model	Full model	All	84.99	1.01	0.06
Cosso	X_1, X_4, X_5, X_6	All, except X_1	85.51	5.16	0.72
Thin plate spline with double penalty	X_1, X_4, X_5, X_6, X_7	All	85.15	3.55	0.03
Thin plate spline with shrinkage	X_1, X_3, X_4, X_5, X_6, X_7	All	85.20	1.08	0.01
Non-negative garrote	X_1, X_2, X_5, X_6, X_7	All	85.50	0.21	0.01
Stepwise	X_1, X_4, X_5, X_6	All, except X_1	85.11	2.04	0.08
GAMBoost	Full model	All	79.74	352.24	6.78
Modified backfitting	X_1, X_2, X_3, X_4, X_5	All, except X_1	85.26	0.56	0.03
mRMR	X_1, X_5, X_7	–	47.89	0.02	0.01
HSIC-Lasso	X_1, X_4, X_7	–	76.87	0.55	0.06
Hybrid algorithm	X_1, X_5, X_6	–	*85.31*	*38.44*	*7.60*

Italics mark the results of the algorithm that's our own development and not one implemented from literature direct effect on the target but causes harmful concurvity in the models. So, these models contain all the true features of Y, but still pick up noise in feature X_4 that only has an indirect effect on the target through X_6.

The algorithms that attempt to control for concurvity (mRMR, HSIC-Lasso and HA) each select 3 features (in case of the mRMR this information was given to the method a priori) that have no concurvity among them. While for the HA these features are the 3 true features of Y, the other two algorithm omits X_6 in favour of X_7 that only mediates partial effects of X_1. This causes a rather significant drop in predictive performance in the models proposed by the mRMR and HSIC-Lasso. These two methods only control for pairwise concurvity and while this control results in models that are technically free from concurvity, they miss a true feature from the proposed model due to the multivariate relationship among the candidate features.

The HA algorithm with its direct control on multivariate concurvity can correctly identify the 3 true features of Y and this results in a predictive performance that is not significantly different from the methods that propose a model with more features. So, the HA could propose the true model under completely multivariate concurvity among the candidate features.

Of course, the best subset selector nature of the HA results again in higher expected runtime when compared to the other methods. This higher expected runtime is not as substantial as the expected runtime of the GAMBoost. The result is like what we experienced in the concrete dataset. Furthermore, the GAMBoost has no benefits on the side of predictive performance similarly to the other algorithms without concurvity constraints.

7 Numerical results on the credit card default dataset

In our second experiment, the task is to estimate for clients in a Taiwanese bank if they are to report default on their credit card loans in one month from now. This dataset consists of 30,000 observations and 26 possible features after applying a dummy coding for categorical variables. Source of the dataset is Yeh and Lien (2009). The feature selection task in this case is not solvable via examining all possible feature subsets and selecting the best one. Best models proposed by each algorithm are only comparable to each other, there is no reference point.

Meaning of the variables in the dataset is the following:

- LIMIT_BAL—Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- SEX—Gender
  - (1 = male; 2 = female)
- EDUCATION – Education
  - (1 = graduate school; 2 = university; 3 = high school; 4 = others)
- MARRIAGE—Marital status

  - (1 = married; 2 = single; 3 = others).

- AGE—Age (year)
- PAY_*X*—Payment records *X* months before data collection $X \in \{0, 2, 3, 4, 5, 6\}$
  - (− 1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; …)
- BILL_AMT*X*—Amount of bill statement (NT dollar) *X* months before data collection $X \in \{1, 2, 3, 4, 5, 6\}$
- PAY_AMT*X*—Amount of previous payment (NT dollar) *X* months before data collection $X \in \{1, 2, 3, 4, 5, 6\}$
- default_nextMonth—Target Variable: binary variable, default payment one month after data collection
  - (Yes = 1, No = 0)

Based on Yeh and Lien (2009), we can determine that there are some clients with false encodings in the categorical variables, like cases with 0 in their MARRIAGE variable. These observations are removed from the dataset, and 22,165 observations are left. This reduced dataset is randomly split to training and test sets in a 7:3 ratio, similarly to the case of the concrete comprehensive strength dataset.

The target variable has a Bernoulli distribution, so the link function for GAMs is the logit and to evaluate predictive performance, we examine the Area Under the ROC Curve (*AUC*) on the test set.

7.1 Addressing some computational issues

Global optima for the feature selection task on the credit card default dataset cannot be determined. Examining all feature subsets is computationally infeasible. If we take 100 random feature subsets and estimate their corresponding GAM, we found that the lower bound for the 95% confidence-interval of the expected runtime of one model is 0.35 min. This means, generating all the 2^{26} subsets would take $\frac{0{,}35 \cdot 2^{26}}{60 \cdot 24 \cdot 365{,}25} = 44.72$ years.

In the cosso and GAMBoost algorithms for this dataset, we encountered a RAM shortage problem. The cosso algorithm tries to fit a smooth function from a RKHS for every feature and it uses all values from the values set of the feature as knots (where a thin plate spline would only use k_j knots). This results in a 46 GB of RAM requirement to store the representational matrices for each f_j. In case of the GAMBoost, RAM requirement of 9.8 GB comes from the need to test several values for the number of maximum iterations via cross validation. A RAM requirement of 9.8 GB is not necessarily problematic for a better PC nowadays, but the configuration used for our numerical experiments only has 8 GB of RAM available.

A solution for the RAM problems of the cosso and GAMBoost algorithms can be found in the original paper for the cosso algorithm (Lin and Zhang 2006). In this paper, the authors suggest taking an i.i.d. sample from the original training set and use it as a training set for the cosso. Sample size should be significantly smaller than the size of the original training set, so the computations are feasible in RAM, but it should be large enough, so the model parameters are not significantly different from the values that would be estimated from the complete training set. For this experiment, we apply $n = 5000$. In this case, we simulated 1000 i.i.d. sample and estimated the parameters of the full GAM model (with thin plate splines) in each sample. Sampling distribution of the McFadden pseudo R-squared can be considered $N(0.217;0.011)$. The p-value of the Shapiro–Wilk test for normality is 0.9125. On the complete training set $\overline{R}^2 = 0,217$ for the full GAM model. So, the maximum difference of a sample $\overline{R}^2$ from the one calculated on the complete training set with a 95% probability is $2 \cdot 0.011 = 0.022$. With a sample of $n = 5000$, the lower bound for the 95% confidence-interval of the expected computation time for one model is 0.207 min according to our simulations. This means, generating all the 2^{26} feature subsets would take about 26.46 years. This is a noticeable decrease from the initial 44.72 years. If we also allow the simultaneous estimation of the GAMs from the 1000 samples (as it is proposed for the HA in Sect. 4.2), then expected runtime for generating all feature subsets would decrease to 4.18 years.

An i.i.d. sampling of $n = 5000$ is also applied for the HA in this experiment because of the huge decrease in expected runtime for the estimation in a GAM. As HA is a best subset feature selection algorithm, decreased computation time for a solution in the memory is always preferred. Furthermore, based on our simulations, this technique causes the sample $\overline{R}^2$ to differ from the one computed on the complete training set only by a maximum of 0.022 with a 95% probability. So, there is no relevant loss in predictive performance due to the sampling. For this dataset, we apply a memory size of 60 and in each trial the HA is run 6 times as suggested in Sect. 4.2 to ensure a "good" initial random memory. Every other HA parameter is the same as the ones suggested in Table 2.

7.2 Discussion of numerical results

For the credit card default dataset, some benchmark models are also applied as Yang and Zhang (2018) has tested the performance of several supervised learners on this dataset. The investigation of Yang and Zhang (2018) only concerned with predictive performance, so feature selection is not applied there. Therefore, we examined our own benchmark models as well. A decision tree trained by CART algorithm with the Gini-index as a splitting measure, as it has implicit feature selection during the tree training process, and it can handle non-linear effects. Furthermore, a random forest (RF) ensemble learner with recursive feature elimination (RFE) is also investigated in this paper. The decision tree is implemented in R with the help of the *rpart* package (Therneau and Atkinson 2018), and the RF with RFE is implemented via the *caret* package (Kuhn et al. 2019).

The RFE algorithm looks for a subset of features by starting with all features in the model and removes features until a predefined number of features remains. This is achieved by fitting the current machine learning model (in our case the RF), ranking features by their importance and omitting the least important features, then refitting the model. The process is repeated until a specified number of features remains.

We tested the RFE with the RF algorithm for feature subsets with a size of 26, 25, 20, 25, 10, 5, 3, 1 and selected the one with the best cross-validated misclassification rate score on the training set.

All algorithms are again run 30 times to examine the stability of the final models for each algorithm, the expected runtimes, and the standard deviation of the runttimes. In the case of those algorithms, where the results are not the same for all 30 trials, the best model is shown in the results, so every algorithm is represented with its best performance. Each algorithm is run on the same training set and evaluated on the same test set for each trial. This way, we try to ensure the differences we see in performance are due to the different characteristics of each algorithm.

The results of the algorithms described in Sect. 4 for the credit card defaults dataset are summarised in Table 5. From Yang and Zhang (2018) only the best (lightGBM) and worst (GLM) models are shown.

The most important takeaway from Table 5 is quite similar to the results of the simulated and concrete comprehensive strength datasets. Models proposed by the mRMR, HSIC-Lasso and HA are using relatively few features (only 3–6) and have no significant differences in predictive performance on the test set than the rest of the proposed models. Even if they are compared to the best model in Yang and Zhang (2018) (lightGBM) or to the random forest model combined with RFE. The rest of the feature selection algorithms again prefer larger models with at least 15 features. Based on the experiences from the simulated example, in the cases of these algorithms, several of the selected features can be considered noise and not true features of credit card default. The only exception is the model proposed by the cosso that prefers a sparser solution, but still uses significantly more features than the three algorithms with concurvity constraints. So, again, the main difference in the proposed models lies in the quality and size of the proposed feature set, not in predictive performance.

It is important to see that feature PAY_0 is selected by every algorithm but selecting an earlier version of the feature (like PAY_2 or PAY_6) causes the violation of concurvity constraints. This is because features that describe payment history of the clients are autocorrelated, so payment status in the month of data collection can be accurately estimated based by the client's previous payment statuses. This phenomenon is the main reason that in the full model all PAY, BILL_AMT and PAY_AMT features violate concurvity constraints. However, based on the GAMBoost model (that does not use all the payment history features), this autocorrelation does not last for the longer lags. As PAY_AMT6 does not violate the concurvity constraints. So, a model could benefit if its feature set contains a longer lag for the BILL_AMT, PAY_AMT features next to PAY_0, as these longer lags contain new and not redundant information on the default probability. However, based on the cosso model, we can see that putting a longer lag of the PAY feature like PAY_6 in the model next to PAY_0 is not advisable as it causes harmful concurvity. So, PAY_0 should only be combined with longer lags of BILL_AMT, PAY_AMT.

These redundancies between the features describing payment history are not completely handled even by the mRMR and the HSIC-Lasso models as they only examine for pairwise relationships between features. For example, in the mRMR

Table 5 Numerical Results of the Examined Algorithms on the credit card default dataset

Algorithm	Selected features	Features that Violate the Concurvity Constraints	AUC (Test Set)	Expected Runtime (min)	St. D. of Runtimes (min)
Full Model	Full Model	All, except AGE, SEX, EDUCATION, MARRIAGE	0.769	0.305	0.041
Cosso	LIMIT_BAL, EDUCATION, PAY_0+2+3+4+6	PAY_0+2+3+4+6	0.701	64.272	20.979
Thin plate spline with double penalty	Full Model, except BILL_AMT5+6, PAY_AMT5	All, except AGE, SEX, EDUCATION, MARRIAGE	0.768	6.624	0.105
Thin plate spline with shrinkage	Full Model, except BILL_AMT6, PAY_AMT5	All, except AGE, SEX, EDUCATION, MARRIAGE	0.767	5.183	0.065
Non-negative garrote	Full Model, except PAY_6, BILL_AMT5	All, except AGE, SEX, EDUCATION, MARRIAGE	0.767	0.670	0.003
Stepwise	Full Model	All, except AGE, SEX, EDUCATION, MARRIAGE	0.769	0.309	0.045
GAMBoost	LIMIT_BAL, SEX, MARRIAGE, AGE, PAY_0+2+3+4+6. PAY_AMT1+4+5+6	All, except LIMIT_BAL, AGE, SEX, MARRIAGE, PAY_AMT6	0.761	692.237	9.497
Modified backfitting	LIMIT_BAL, AGE, SEX, EDUCATION, MARRIAGE, PAY_0+2+3+4+5+6, BILL_AMT1, PAY_AMT1+2+4+6	All, except LIMIT_BAL, AGE, SEX, EDUCATION, MARRIAGE	0.765	0.443	0.214
mRMR	AGE, PAY_0+2+5, PAY_AMT_1+6	PAY_2	0.744	1.629	0.035
HSIC-Lasso	AGE, PAY_0+2+3+4	PAY_0+2+3	0.741	0.446	0.048
Hybrid algorithm	*LIMIT_BAL, PAY_0, PAY_AMT3*	–	*0.749*	*147.257*	*15.079*
Decision tree	PAY_0	–	0.641	0.0127	0.0004

Table 5 (continued)

Algorithm	Selected features	Features that Violate the Concurvity Constraints	AUC (Test Set)	Expected Runtime (min)	St. D. of Runtimes (min)
Random forest with RFE	Full Model	All, except AGE, SEX, EDUCATION, MARRIAGE	0.756	132.543	0.758
GLM	There is no feature selection in Yang and Zhang (2018), so the full feature set is used in all cases		0.723	There are no numerical results for runtimes in Yang and Zhang (2018)	
LightGBM			0.790		

Italics mark the results of the algorithm that’s our own development and not one implemented from literature

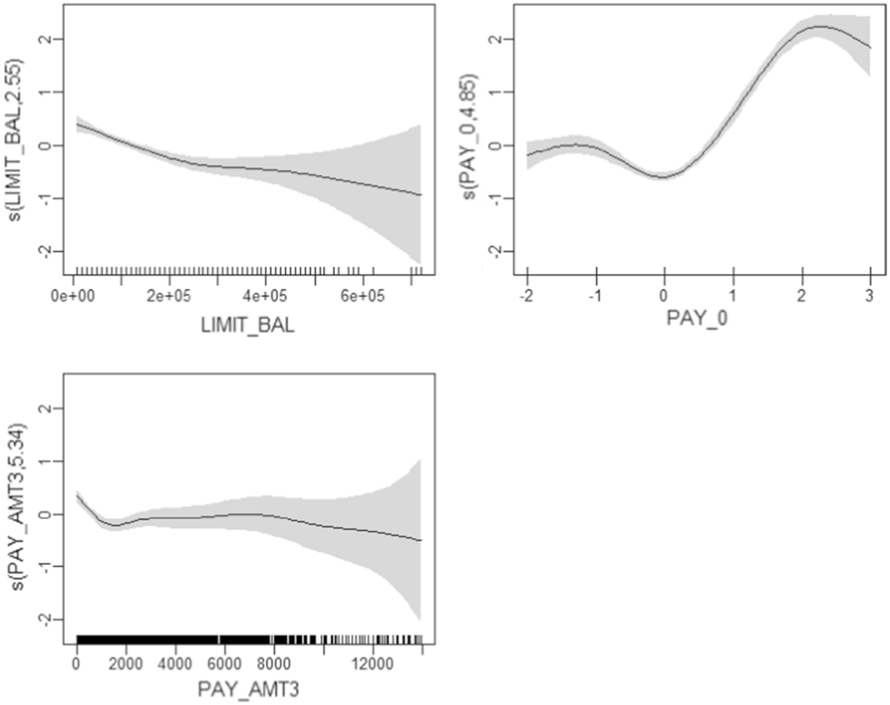

Fig. 4 The fitted smooth functions in the HA model for the credit card default dataset with 95% confidence-intervals

model PAY_2 can be accurately estimated as a function of PAY_0 and PAY_5. On the other hand, the HA clearly shows that selecting PAY_AMT3 next to PAY_0 and LIMIT_BAL adds new, not redundant information to the model on credit card default. This behaviour is similar to what we experienced on the simulated dataset.

Comparing feature effects in the decision tree and the HA GAM model can result in interesting discoveries. Fitted smooth functions of the features in the HA models are shown in Fig. 4. During the detailed diagnostic examinations of the GAM proposed by the HA, we found 1856 observations with Cook's distances greater than 1. Figure 4 is created from a GAM where these influential observations are omitted during parameter estimation. This way the model *AUC* did not decrease significantly. It became 0.745 instead of 0.749.

Poor decision tree predictive performance on the test set can be attributed to the fact that the resulting tree is too simple. The only rule of the tree is that if a client has a payment delay for at least two months, then the client is predicted to default next month and otherwise not.

Inclusion of two extra features (LIMIT_BAL and PAY_AMT3) and increasing the odds of default for negative PAY_0 amounts, results in significantly higher predictive performance on the test set.

Negative values in the PAY_0 feature mean overpayment. This can cause an increased risk of credit card default as banks usually compensate these overpayments with refunds.[2] Clients tend to overestimate the value of extra credits on their cards, and this can lead to serious overspending and then default.

In case of expected runtimes, best subset algorithms like RFE and HA have a clear disadvantage to regularization estimate, stepwise or linear search algorithms, not to mention the simple decision tree. However, the GAMBoost algorithm has the longest expected runtime, like in the case of the two other datasets, due to the cross validation necessary to find the optima for the maximum number of iterations. Expected runtime of the algorithm is above 11 h, despite the application of random sampling to significantly decrease the number of observations in the training set.

Summarizing the results obtained from the numerical experiments on the credit card default dataset, we can repeatedly conclude that if our aim is to build a truly parsimonious model with no harmful concurvity among the features without considerable runtime constraints, then the HA can be a viable alternative to the rest of the examined algorithms. These algorithms have very similar predictive performances, but the HA can show us a model where feature effects are clearly interpretable due to the smaller number of selected features and the lack of concurvity. Based on experiences from the simulated dataset, we can suggest that algorithms with no concurvity constraint are again picking up features that are just noise in the model. The algorithms with only pairwise control for concurvity can select features with redundant information since they violate the multivariate concurvity constraints.

However, if our main goal is purely prediction, then feature selection seems unnecessary and the analyst should go with the fastest solution, since there are no significant differences in the predictive performances of the algorithms proposing the full model. Only application of the decision tree is not advised since it proposes a too simple model here.

8 Summary

In our current paper, we examined and compared the performance of 10 different feature selection algorithms for GAMs on simulated and two real-world datasets. Based on the general properties of these algorithms and found that they can be separated into four clusters: stepwise, boosting, regularization and concurvity controlled methods.

After running numerical experiments on all three datasets, we found that algorithms with no constraints on concurvity tend to prefer large models where only a few features are removed from the final model. This causes harmful concurvity in the proposed models, which makes marginal effect of features uninterpretable due to the redundancy among the selected features. Results from the simulated data suggest that most of the features selected are just noise for the target variable. Furthermore, predictive performance of these algorithms does not differ significantly from each

[2] https://www.creditcards.com/credit-card-news/credit-balance-overpay-refund-1282.php Downloaded: 2020. 11. 04.

other apart from some rare exceptions (e.g. the non-negative garrote on the concrete dataset and the GAMBoost on the simulated data).

To tackle the concurvity phenomenon, some recent feature selection algorithms such as the mRMR and the HSIC-Lasso incorporated some constraints on concurvity in their objective function. These algorithms interpret concurvity as pairwise non-linear relationship between features, so they do not account for the case when a feature can be accurately estimated as a multivariate function of several other features. Therefore, during our numerical experiments the models proposed by the mRMR, and HSIC-Lasso algorithms had violated the concurvity constraints in some features. In the simulated example they excluded a true feature from their proposed models.

Our own solution to the problem, the HA introduces constraints on multivariate concurvity directly. Due to this constraint, the HA usually applied very few features in its final models but achieved a predictive performance that rivalled that of models with far more features. Lack of concurvity in models proposed by the HA means that we can take advantage of the additive structure of GAMs and interpret each smooth function separately. So, feature effects on the target are distinguishable and interpretable in these models. In the simulated example, the HA could correctly identify the true model. Expected runtime of the HA on larger datasets can be long but expected runtime of the GAMBoost algorithm is much longer. Furthermore, on larger datasets, the expected runtime of the HA is similar to that of the RFE feature selection algorithms combined with a random forest learner.

Overall, results from the three numerical experiments show that while there is no significant difference in the predictive performance of most algorithms, usually there is a difference in the number of features used. Algorithms without concurvity control usually select the whole feature set (or something very close to it), while the three concurvity controlled solutions usually achieve the same level of performance in the test set, but with significantly less features selected. Algorithms with pairwise concurvity constraints can still violate the multivariate concurvity constraint, despite using a small number of features, but due to its direct constraint on multivariate concurvity, the HA never does. So, models proposed by the HA have similar predictive performance than the rest of the examined algorithms, but with a more parsimonious and concurvity-free feature set.

Application of the HA needs further studies on more real-world datasets. Due to the strict and direct concurvity constraints, models proposed by the HA can be affected by omitted variable bias (OVB). OVB occurs when a statistical model leaves out one or more relevant variables. The bias results in the model attributing the effect of the missing features to those that were included. Further numerical experiments are required to examine how the strictness of the concurvity constraints in HA affects the subset of selected features and whether OVB can be present in the final models.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00180-022-01292-7.

Funding Open access funding provided by Corvinus University of Budapest. Supported by the ÚNKP-20-3-II New National Excellence Program of the Ministry For Innovation and Technology from the source of the National Research, Development and Innovation Fund.

Availability of data and material The results generated in this research are available at https://github.com/KoLa992/Hybrid-algorithm-for-GAMs.

Code availability The code developed in this research is available at https://github.com/KoLa992/Hybrid-algorithm-for-GAMs. The Python code is available at https://replit.com/@LaszloKovacs5/HSICLassoTest#main.py.

Declarations

Conflict of interest We declare that there are no financial interests or personal relationships that could have influenced this work.

References

Altman N, Krzywinski M (2016) Analyzing outliers: Influential or nuisance? Nat Methods 13(4):281–283

Amodio S, Aria M, D'Ambrosio A (2014) On concurvity in nonlinear and nonparametric regression models. Statistica 74(1):85–98

Augustin NH, Sauleau EA, Wood SN (2012) On quantile quantile plots for generalized linear models. Comput Stat Data Anal 56(8):2404–2409. https://doi.org/10.1016/j.csda.2012.01.026

Belitz C, Lang S (2008) Simultaneous selection of variables and smoothing parameters in structured additive regression models. Comput Stat Data Anal 53(1):61–81. https://doi.org/10.1016/j.csda.2008.05.032

Binder H, Tutz G (2008) A comparison of methods for the fitting of generalized additive models. Stat Comput 18(1):87–99. https://doi.org/10.1007/s11222-007-9040-0

Breheny P, Huang J (2011) Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann Appl Stat 5(1):232–253

Cantoni E, Flemming JM, Ronchetti E (2011) Variable selection in additive models by non-negative garrote. Stat Model 11(3):237–252. https://doi.org/10.1177/1471082X1001100304

Chong IG, Jun CH (2005) Performance of some variable selection methods when multicollinearity is present. Chemom Intell Lab Syst 78(1–2):103–112. https://doi.org/10.1016/j.chemolab.2004.12.011

Climente-González H, Azencott CA, Kaski S, Yamada M (2019) Block HSIC Lasso: model-free biomarker detection for ultra-high dimensional data. Bioinformatics 35(14):i427–i435. https://doi.org/10.1093/bioinformatics/btz333

De Jay N, Papillon-Cavanagh S, Olsen C, El-Hachem N, Bontempi G, Haibe-Kains B (2013) mRMRe: an R package for parallelized mRMR ensemble feature selection. Bioinformatics 29(18):2365–2368. https://doi.org/10.1093/bioinformatics/btt383

Du M, Liu N, Hu X (2019) Techniques for interpretable machine learning. Commun ACM 63(1):68–77. https://doi.org/10.1145/3359786

Efroymson MA (1960) Multiple regression analysis. In: Ralston A, Wilf HS (eds) Mathematical methods for digital computers. John Wiley, New York, pp 191–203

Gretton A, Bousquet O, Smola A, Schölkopf B (2005) Measuring statistical dependence with Hilbert-Schmidt norms. In: International conference on algorithmic learning theory. Springer, Berlin, pp 63–77

Gu H, Kenney T, Zhu M (2010) Partial generalized additive models: an information-theoretic approach for dealing with concurvity and selecting variables. J Comput Graph Stat 19(3):531–551. https://doi.org/10.1198/jcgs.2010.07139

Hall MA (1999) Correlation-based feature selection for machine learning. Dissertation, University of Waikato.

Hartigan JA, Wong MA (1979) Algorithm AS 136: A k-means clustering algorithm. J R Stat Soc Ser C Appl Stat 28(1):100–108. https://doi.org/10.2307/2346830

Hastie TJ, Tibshirani RJ (1990) Generalized additive models. Chapman and Hall, London

Hastie TJ (2018) gam: generalized additive models. R package version 1.16. https://CRAN.R-project.org/package=gam

Huo X, Ni X (2007) When do stepwise algorithms meet subset selection criteria?. Ann Stat. pp 870–887. https://www.jstor.org/stable/25463581

James G, Witten D, Hastie TJ, Tibshirani R (2013) An introduction to statistical learning: with applications in R. Springer, New York

Jia J, Yu B (2010) On model selection consistency of the elastic net. Stat Sin 20:595–611

Kuhn M, Wing J, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, Kenkel B, the R Core Team, Benesty M, Lescarbeau R, Ziem A, Scrucca L, Tang Y, Candan C, Tyler H (2019) caret: Classification and Regression Training. R package version 6.0–84. https://CRAN.R-project.org/package=caret

Lai J, Lortie CJ, Muenchen RA, Yang J, Ma K (2019) Evaluating the popularity of R in ecology. Ecosphere 10(1):e02567. https://doi.org/10.1002/ecs2.2567

Láng B, Kovács L, Mohácsi L (2017) Linear regression model selection using a hybrid genetic – Improved harmony search parallelized algorithm. SEFBIS J 11(1):2–9

Lin Y, Zhang HH (2006) Component selection and smoothing in multivariate nonparametric regression. Ann Stat 34(5):2272–2297. https://doi.org/10.1214/009053606000000722

Mansfield ER, Helms BP (1982) Detecting multicollinearity. Am Stat 36(3a):158–160

Marra G, Wood SN (2011) Practical variable selection for generalized additive models. Comput Stat Data Anal 55(7):2372–2387. https://doi.org/10.1016/j.csda.2011.02.004

McFadden D (1974) Conditional logit analysis of qualitative choice behaviour. In: Zarembka P (ed) Frontiers in econometrics. Academic Press, New York, pp 105–142

Molnar C (2020) Interpretable machine learning. Leanpub, Victoria

Perperoglou A, Sauerbrei W, Abrahamowicz M, Schmid M (2019) A review of spline function procedures in R. BMC Med Res Methodol 19(1):1–16. https://doi.org/10.1186/s12874-019-0666-3

Ramsay TO, Burnett RT, Krewski D (2003) The effect of concurvity in generalized additive models linking mortality to ambient particulate matter. Epidemiology 14(1):18–23

Schapire RE (1990) The strength of weak learnability. Mach Learn 5:197–227

Schmid M, Hothorn T (2008) Boosting additive models using component-wise P-splines. Comput Stat Data Anal 53(2):298–311. https://doi.org/10.1016/j.csda.2008.09.009

Signoretto M, Pelckmans K, Suykens JA (2008) Functional ANOVA Models: Convex-concave approach and concurvity analysis (No. 08–203). Internal Report.

Therneau T, Atkinson B (2018) rpart: recursive partitioning and regression trees. R package version 4.1–13. https://CRAN.R-project.org/package=rpart

Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc: Ser B (methodol) 58(1):267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tutz G, Binder H (2006) Generalized additive modeling with implicit variable selection by likelihood-based boosting. Biometrics 62(4):961–971. https://doi.org/10.1111/j.1541-0420.2006.00578.x

Weston S (2019a) foreach: provides foreach looping construct. R package version 1.4.7. https://CRAN.R-project.org/package=foreach

Weston S (2019b) doParallel: Foreach Parallel Adaptor for the 'parallel' Package. R package version 1.0.15. https://CRAN.R-project.org/package=doParallel

Wood SN (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. J R Stat Soc Ser B Stat Methodol 73(1):3–36. https://doi.org/10.1111/j.1467-9868.2010.00749.x

Wood SN (2017) Generalized additive models: an introduction with R, 2nd edn. Chapman and Hall/CRC, London

Wooldridge JM (2016) Introductory econometrics: a modern approach. Nelson Education, Toronto

Yang S, Zhang H (2018) Comparison of several data mining methods in credit card default prediction. Intell Inf Manag 10(05):115–122. https://doi.org/10.4236/iim.2018.105010

Yeh IC (1998) Modeling of strength of high-performance concrete using artificial neural networks. Cem Concr Res 28(12):1797–1808. https://doi.org/10.1016/S0008-8846(98)00165-3

Yeh IC, Lien CH (2009) The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Syst Appl 36(2):2473–2480. https://doi.org/10.1016/j.eswa.2007.12.020

Zhang HH, Lin CY (2013) cosso: fit regularized nonparametric regression models using COSSO penalty. R package version 2.1–1. https://CRAN.R-project.org/package=cosso

Zhao P, Yu B (2006) On model selection consistency of Lasso. J Mach Learn Res 7:2541–2563

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.