

Sparse Hierarchical Regression with Polynomials

Dimitris Bertsimas^{*1} and Bart Van Parys^{†1}

¹*Operations Research Center, Massachusetts Institute of Technology*

Abstract

We present a novel method for exact hierarchical sparse polynomial regression. Our regressor is that degree r polynomial which depends on at most k inputs, counting at most ℓ monomial terms, which minimizes the sum of the squares of its prediction errors. The previous hierarchical sparse specification aligns well with modern big data settings where many inputs are not relevant for prediction purposes and the functional complexity of the regressor needs to be controlled as to avoid overfitting. We present a two-step approach to this hierarchical sparse regression problem. First, we discard irrelevant inputs using an extremely fast input ranking heuristic. Secondly, we take advantage of modern cutting plane methods for integer optimization to solve our resulting reduced hierarchical (k, ℓ) -sparse problem exactly. The ability of our method to identify all k relevant inputs and all ℓ monomial terms is shown empirically to experience a phase transition. Crucially, the same transition also presents itself in our ability to reject all irrelevant features and monomials as well. In the regime where our method is statistically powerful, its computational complexity is interestingly on par with **Lasso** based heuristics. The presented work fills a void in terms of a lack of powerful disciplined nonlinear sparse regression methods in high-dimensional settings. Our method is shown empirically to scale to regression problems with $n \approx 10,000$ observations for input dimension $p \approx 1,000$.

Keywords: Nonlinear Regression, Sparse Regression, Integer Optimization, Polynomial Learning

1 Introduction

We consider the problem of high-dimensional nonlinear regression. Given input $X = (x_1, \dots, x_n) \in \mathbb{R}^{n \times p}$ and response data $Y = (y_1, \dots, y_n) \in \mathbb{R}^n$, we set out to find an unknown underlying nonlinear relationship

$$y_t = g(x_t) + e_t, \quad \forall t \in [n],$$

where $E := (e_1, \dots, e_n)$ in \mathbb{R}^n is the error term. High-dimensional regression is a problem at the core of machine learning, statistics and signal processing. It is clear that if we are to carry any hope of success, some structure on the nature of the unknown nonlinear relationship g between input and response data must be assumed. Classical statistical learning theory (Vapnik, 2013) indeed requires the complexity of the set of considered functions to be bounded in some way. Polynomial regression has a long history (Smith, 1918) and is mentioned in almost any standard work on machine learning. We will consider in this paper all nonlinear relationships in the form of polynomials in p variables of total degree at most r . We denote with \mathcal{P} the set of polynomials of total degree r . Typically the polynomial which best explains data is defined as the minimizer to the abstract optimization problem

$$\min_{g \in \mathcal{P}} \frac{1}{2} \sum_{t \in [n]} \|y_t - g(x_t)\|^2 + \frac{1}{2\gamma} \|g\|^2, \quad (1)$$

^{*}dbertsim@mit.edu

[†]vanparys@mit.edu

over polynomial functions g in \mathcal{P} . The squared norm $\|g\|^2$ of a polynomial g is taken here to mean the sum of squares of its coefficients. The best polynomial in formulation (1) minimizes a weighted combination of the sum of its squared prediction errors and its coefficient vector. This latter Ridge regularization (Tikhonov, 1943; Hoerl and Kennard, 1970) term stabilizes its solution and avoids overfitting. An alternative interpretation of the regularization term as a precaution against errors in the input data matrix X has been given for instance in (Bertsimas and Copenhaver, 2017). Nevertheless, the value of the hyperparameter γ must in practice be estimated based on historical data using for instance cross validation.

By allowing for nonlinear dependence between input and response data polynomial regression can discover far more complex relationships than standard linear regression. For a sufficiently large degree r in fact, any continuous functional dependence can be discovered up to arbitrary precision (Stone, 1948). The previous observation leads to the fact that polynomial regression is a hybrid between parametric and nonparametric regression. Depending on the degree r of the polynomials considered, it falls between textbook parametric regression ($r = 1$) which assumes the functional dependence g between input and response data to be linear and completely nonparametric regression ($r \rightarrow \infty$) where nothing beyond continuity of the dependence g between input and response data is assumed. Although nonparametric approaches are very general and can unveil potentially any continuous relationship between input and response data, they nonetheless seriously lack in statistical power. Indeed, nonparametric methods such as kernel density estimation (Turlach, 1993) need a huge number of samples in order to return statistically meaningful predictions. This curse of dimensionality is especially harmful in high-dimensional settings $p \gg n$ commonly found in modern data sets.

The polynomial regressor g in the regression problem (1) is a sum of at most $f := \binom{p+r}{r}$ monomial features. A seminal result due to Vapnik (1998) states that the high-dimensionality of the unconstrained regression problem (1) does not pose an obstacle to its numerical solution. Indeed, the feature dimensionality f can be avoided in its entirety using the now classical kernel representation of polynomials put forward by Mercer (1909). Regression formulations amendable to such a kernel reformulation are typically referred to a kernel learning methods. It is thanks to both the flexibility of the regression formulation (1) and its computational tractability that polynomial and even more general kernel methods have experienced a lot of interest in the learning community (Suykens and Vandewalle, 1999). Polynomial regression using kernel learning has indeed been used with success in many applications such as character recognition, speech analysis, image analysis, clinical diagnostics, person identification, machine diagnostics, and industrial process supervision. Today, efficient and mature software implementations of these so called polynomial kernel regressors are widely available, see c.f. (Schölkopf and Smola, 2002; Pelckmans et al., 2002). Unfortunately in high-dimensional settings ($f \gg n$), the previously discussed curse of dimensionality and overfitting phenomena do pose a formidable obstacle to the recovery of the correct nonlinear relationship between input and response data. That is, in settings where we have many more monomial input features f than observations n , it becomes unlikely that we recover a statistically meaningful regressor by solving (1).

Here we will work to address the previous issue by providing a sparse counterpart to the polynomial regression problem (1). Sparse regression has recently been identified in the works of Hastie, Tibshirani, and Wainwright (2015) and Candès, Romberg, and Tao (2006) as an excellent antidote to the malignant phenomena of both dimensionality and overfitting. Interestingly, very few mature machine learning methods seem to have been developed which can deal reliably with sparse nonlinear regressors in a high-dimensional settings despite the obvious relevance of this problem class. One notable exception of direct relevance here is the **SPORE** algorithm by Huang et al., 2010 which uses an approach based on ℓ_1 -regularization. We subsequently describe a hierarchical sparse regression problem which controls both the dependence and functional complexity of the regression polynomials considered.

1.1 Hierarchical (k, ℓ) -Sparsity

The popularity and effectiveness of sparse regression can from a practical perspective be explained by the following two observations. In the digital age obtaining and processing vast amounts of input data is increasingly less of a burden. Nevertheless, we expect only a small number k of all p recorded inputs to be meaningfully correlated to the response data Y . The trouble is that we can not tell the relevant features

from the obfuscating bulk of data ahead of time. Sparsity hence firstly describes the limited functional dependence between input and response data. As only a finite amount of data can be recorded, one must avoid overfitting by limiting the complexity of the considered functional relationships. Of the potential f monomials making up the polynomial regressor g , the considered relationships should only depend on a select few of them. We use sparsity to enforce simplicity among the considered relationships. As both these described sparsity considerations are quite different in nature, we believe it is paramount not to conflate them.

We will say that the function $g \in \mathcal{P}_{k,\ell}$ is so called (k, ℓ) -sparse if it is the sum of ℓ monomials in at most k inputs. For instance the regressor $g(x) = x_1^2 + x_2x_3$ would be $(3, 2)$ sparse as it depends on the three inputs x_1, x_2 and x_3 , and is made up of two monomials x_1^2 and x_2x_3 . The resulting problem of hierarchical sparse regression can be cast as the regression problem

$$\min_{g \in \mathcal{P}_{k,\ell}} \frac{1}{2} \sum_{t \in [n]} \|y_t - g(x_t)\|^2 + \frac{1}{2\gamma} \|g\|^2. \quad (2)$$

The previous regression formulation is a structured sparsity constrained version of (1). Using this novel notion of hierarchical sparse regressors, we hope to keep the statistical power of parametric regression while simultaneously allowing highly nonlinear relationships between input and response data as well. Although structured hierarchical sparsity patterns were studied already by Zhao, Rocha, and Yu, 2009, they were never considered in our polynomial regression context directly. A related hierarchical kernel learning approach to a convex proxy of problem (2) is studied in Bach, 2009. The regression problem (2) carries the additional benefit of automatically yielding highly interpretable regressors with only a few nonlinear terms and input dependencies. By explicitly controlling both the dependence complexity k of used inputs as well as the functional complexity ℓ of the regression polynomials, the hierarchical sparse regression problem (2) promises to deliver nonlinear regressors with significant statistical power even in high-dimensional settings.

Unfortunately, solving the hierarchical sparse regression problem (2) can be challenging. Bringing to bear the power of modern integer optimization algorithms combined with smart heuristics, we will nevertheless show that many hierarchical sparse regression problems can nevertheless be handled.

1.2 Exact Scalable Algorithms

The problem of sparse linear regression has been studied extensively in the literature. Despite being provably hard in the sense of complexity theory’s NP hardness, in practice many successful algorithms are available. Historically, the first heuristic methods for sparse approximation seem to have arisen in the signal processing community (c.f. the work of Mallat and Zhang (1993) and references therein) and typically are of an iterative thresholding type. More recently, one popular class of sparse regression heuristics solve the convex surrogate

$$\min_{\|g\|_1 \leq \ell} \frac{1}{2} \sum_{t \in [n]} \|y_t - g(x_t)\|^2 + \frac{1}{2\gamma} \|g\|^2. \quad (3)$$

to the sparse regression formulation (2). Here the norm $\|g\|_1$ of the polynomial g is meant to denote the sum of the absolute values of its coefficients. The convex proxy reformulation (3) is a direct adaptation of the seminal **Lasso** method of Hastie, Tibshirani, and Wainwright (2015) to the polynomial regression problem (1). The discussed **SPORE** algorithm by Huang et al., 2010 provides an implementation of this idea specific to the polynomial regression context considered here. Where the convex heuristic (3) does not incorporate the hierarchical sparsity structure of our exact formulation (2), more refined methods such as **Group Lasso** (Zhao, Rocha, and Yu, 2009; Bach, 2008) could in principle do so by considering structured norm constraints. There is an elegant theory for convex proxy schemes promising large improvements over the more myopic iterative thresholding methods. Indeed, a truly impressive amount of high-quality work (Bühlmann and Geer, 2011; Hastie, Tibshirani, and Wainwright, 2015; Wainwright, 2009) has been written on characterizing when exact solutions can be recovered, albeit through making strong probabilistic assumptions on the data.

The problem of exact sparse nonlinear regression however seems, despite its importance, not to have been studied extensively. Although they are well studied separately, combining nonlinear with sparse regression

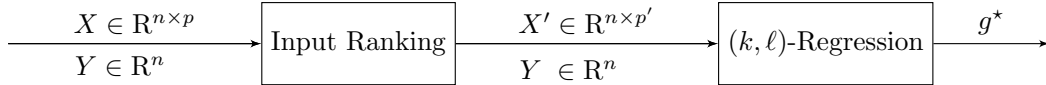


Figure 1: Our two-step approach to (k, ℓ) -sparse regression for observations Y in \mathbb{R}^n and regressors X in $\mathbb{R}^{n \times p}$. In a first step we select the $p' \ll p$ most promising inputs out of the p candidates. The second step then performs exact (k, ℓ) -sparse regression on these most promising candidates. We set out to show that this combination of a smart ranking heuristics and exact sparse regression goes a long way to solve hierarchical sparse regression problems (2) of practical size.

never received much attention. Our recent work (Bertsimas and Van Parys, 2017) (and earlier in (Bertsimas, King, and Mazumder, 2016)) has revealed that despite complexity results, exact sparse linear regression is not outside the realm of the possible even for very high-dimensional problems with a number of features f and samples n in the 100,000s. Contrary to traditional complexity theory which suggests that the difficulty of a problem increases with size, the sparse regression problems seem to have the property that for a small number of samples n , exact regression is not easy to accomplish, but most importantly its solution does not recover the truth. However, for a large number of samples n , exact sparse regression can be done extremely fast and perfectly separates the true features in the data from the obfuscating bulk. These results warrant also the possibility of nonlinear feature discovery in regression tasks.

1.3 Triage Heuristic

Unfortunately the dimension of hierarchical (k, ℓ) -sparse regression problems quickly becomes problematic for all but midsize problem instances. The effective number of regressors f is indeed combinatorial and hence scales quite badly in both the number of regressors p as well of the degree r of the considered polynomials. In order to provide a scalable algorithm to the problem of hierarchical kernel regression it is clear that the dimensionality of the problem needs to be reduced.

Our key insight in this paper will be to use polynomial kernel regression to rank the potential inputs. This heuristic method is very helpful in rejecting many irrelevant candidate inputs without missing out on the actual underlying nonlinear relationship between input and response data. Exact sparse hierarchical kernel regression described before will then be used to identify the relevant nonlinearities from among the most promising candidate inputs; see Figure 1. In this paper we set out to show that a combination of smart heuristics and exact sparse regression goes a long way to solve hierarchical sparse regression problems (2) of practical size.

1.4 Contributions

In this paper, we first and foremost want to promote a novel notion of hierarchical (k, ℓ) -sparsity that rhymes well with the challenges of the big data era. Capturing limited functional dependence and complexity in big data problems is crucial to allow statistically meaningful regression in high-dimensional and nonlinear settings. Hierarchical (k, ℓ) -sparse regression definition is in this regard a first step in the direction of lifting nonlinear regression into high-dimensional settings as well. In particular, we hope that the method presented here will show a more disciplined approach to nonlinear discovery than some more black box methods such as artificial neural networks.

Secondly, we also offer scalable algorithms able to solve these hierarchical (k, ℓ) -sparse regression problems using modern optimization techniques. In accordance with previous results (Bertsimas and Van Parys, 2017), we show that exact sparse regression is not outside the realm of the possible even for very high-dimensional problems with a number of features f and samples n in the 100,000s. We will indeed show that we can reliably discover nonlinear relationships using a combination of smart heuristics and exact sparse regression using a cutting plane approach for convex integer optimization.

In order to judge the quality of a proposed regressor g^* , we must measure on the one hand to what extent all the relevant monomial features are discovered. In order to do so, while at the same time avoiding notational clutter, we need to establish a way to refer to each of the monomials of degree r in p variables in an efficient fashion. Let $m_j : \mathbb{R}^p \rightarrow \mathbb{R}$ for each j in $[f]$ denote a distinct monomial in p inputs of degree at most r . We define the accuracy of a regressor g^* as

$$A\% := \frac{|\text{supp}(g^*) \cap \text{supp}(g_{\text{true}})|}{|\text{supp}(g_{\text{true}})|}, \quad (4)$$

where $\text{supp}(g)$ represents all j such that monomial m_j contributes to the polynomial g . This accuracy measure $A\%$ thus represents the proportion of true underlying monomial features discovered by the proposed polynomial regressor g^* . On the other hand, we can use

$$F\% := \frac{|\text{supp}(g^*) \setminus \text{supp}(g_{\text{true}})|}{|\text{supp}(g^*)|} \quad (5)$$

to quantify how many irrelevant features were wrongly included in the process. Perfect recovery occurs when the method gives the whole truth ($A\% = 100$) and nothing but the truth ($F\% = 0$). In practice, however, machine learning methods must inevitably make a choice between both desirables.

We intend to illustrate that exact sparse regression methods have an inherent edge over proxy based sparse heuristics. Proxy based methods such as **Lasso** do indeed have several well documented shortcomings. First and foremost, as argued by Bertsimas, King, and Mazumder (2016) they do not recover very well the sparsity pattern. Furthermore, the **Lasso** leads to biased regression regressors, since the ℓ_1 -norm penalizes both large and small coefficients uniformly. The ability of our method to identify all relevant features is shown empirically to experience a phase transition. There exists a critical number of data samples n_0 such that when presented sufficient data $n > n_0$ our method recovers the ground truth ($A\% \approx 100$) completely, whereas otherwise its accuracy $A\%$ tends to zero. Crucially, the same number of samples n_0 also enables our method to reject most irrelevant features ($F\% \approx 0$) as well. We thus show that we significantly outperform **Lasso** in terms of offering regressors with a larger number of relevant features (bigger $A\%$) for far fewer nonzero coefficients (smaller $F\%$) enjoying at the same time a marginally better prediction performance. In the regime $n > n_0$ where our method is statistically powerful ($A\% \approx 100$, $F\% \approx 0$), its computational complexity is furthermore on par with sparse heuristics such as **Lasso**. This last observation takes away the main propelling justification for most heuristic based sparsity approaches.

Notation

The knapsack set S_k^p denotes here the binary set $S_k^p := \{s \in \{0, 1\}^p : \sum_{j \in [p]} s_j \leq k\}$, which contains all binary vectors s selecting k components out of p possibilities. Assume that (y_1, \dots, y_p) is a collection of elements and suppose that s is an element of S_k^p , then $y_s \in \mathbb{R}^{|s|}$ denotes the sub-collection of y_j where $s_j = 1$. Similarly, we use $\text{supp}(x) = \{s \in \{0, 1\}^p : s_j = 1 \iff x_j \neq 0\}$ to denote those indices of a vector x which are nonzero. We denote by S_+^n (S_{++}^n) the cone of $n \times n$ positive semidefinite (definite) matrices. Given a matrix $K \in \mathbb{R}^{n \times n}$, we denote its element-wise r th power or Hadamard power as K^{or} , i.e., we have that

$$K^{or} := \begin{pmatrix} K_{11}^r & K_{12}^r & \dots & K_{1n}^r \\ K_{21}^r & K_{22}^r & \dots & K_{2n}^r \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1}^r & K_{n2}^r & \dots & K_{nn}^r \end{pmatrix}.$$

2 Hierarchical (k, ℓ) -Sparse Polynomial Regression

In this section, we discuss two formulations of the hierarchical sparse regression problem (2) through a standard integer optimization lens. In the last twenty plus years the computational power of integer optimization solves has increased at a dramatic speed (Bertsimas, King, and Mazumder, 2016). Where twenty years ago

integer optimization for statistics was branded impossible, recent work (Bertsimas and Van Parys, 2017; Bertsimas, King, and Mazumder, 2016) has shown convincingly that this position needs to be revisited. The position that exact sparse regression is an unattainable goal only to be striven for via admittedly elegant convex heuristics such as **Lasso** should not be held any longer.

We considered in the sparse regression problem (2) as features the set of all monomials of degree at most r . It is clear that this sparse hierarchical regression problem over polynomials can equivalently stated as an optimization problem over their coefficients in the monomial basis m_j for j in $[f]$. We will avoid the need to explicitly order each these monomials as follows. Define the dependents $D(i)$ of any data input i as the set of indices j such that the monomial m_j depends on input i . Similarly, we define the ancestors $A(j)$ of any j as the multiset of inputs making up the monomial m_j . Instead of using the classical monomial basis, we consider a scaled variant in which we take $m_j(\mathbb{1}_p)$ to coincide with the square root of the number of distinct ways to order the multiset $A(j)$. This rescaling of the monomials comes in very handy when discussing the solution of the regression problem (1) in Section 3. In fact, the same scaling is implicitly made by the Mercer kernel based approach as well.

To make the discussion more concrete, we consider a data problem with $p = 3$ inputs and associated data $(y_t, x_{t,1}, x_{t,2}, x_{t,3})$ for $t \in [n]$. We consider all monomials on the three inputs of degree at most two, i.e., we consider the monomials and their corresponding indices as given below.

Monomial m_j	1	$\sqrt{2}x_1$	$\sqrt{2}x_2$	$\sqrt{2}x_3$	x_1^2	$\sqrt{2}x_1x_2$	$\sqrt{2}x_1x_3$	x_2^2	$\sqrt{2}x_2x_3$	x_3^2
Index j	1	2	3	4	5	6	7	8	9	10

The set $D(i)$ corresponds to the set of indices of the monomials in which input x_i participates. In our example above we have $D(1) = \{2, 5, 6, 7\}$ corresponding to the monomials $\{\sqrt{2}x_1, x_1^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3\}$. The set $A(j)$ corresponds to all inputs that are involved in the monomial with index j . Again as an illustration, in our example $A(7) = \{1, 3\}$ corresponding to the inputs x_1 and x_3 .

2.1 Mixed Integer Formulation

With the help of the previous definitions, we can now cast the hierarchical (k, ℓ) -sparse regression problem (2) as a standard optimization problem. The problem of hierarchical sparse regression can indeed be cast as the following Mixed Integer Optimization (MIO) problem

$$\begin{aligned}
\min \quad & \frac{1}{2} \sum_{t \in [n]} \|y_t - \sum_{j \in [f]} w_j \cdot m_j(x_t)\|^2 + \frac{1}{2\gamma} \|w\|^2 \\
\text{s.t.} \quad & w \in \mathbb{R}^f, \quad s \in \mathbb{S}_\ell^f, \quad h \in \mathbb{S}_k^p, \\
& s_j \leq h_i \quad \forall i \in A(j), \quad \forall j \in [f], \\
& -\mathcal{M}s_j \leq w_j \leq \mathcal{M}s_j \quad \forall j \in [f],
\end{aligned} \tag{6}$$

using a big- \mathcal{M} formulation. Its optimal solution w^* gives the coefficients of the polynomial $g^*(x) = \sum_{j \in [f]} w_j^* m_j(x)$ in (2) best describing the relationship between input and observations in our monomial basis $\{m_j\}$. The coefficient w_j of any monomial m_j is only then nonzero when $s_j = 1$ as per the first constraint in (6) for a sufficiently large constant \mathcal{M} . This constant \mathcal{M} needs to be estimated from data. Although nontrivial, this can be done using the results found in (Bertsimas, King, and Mazumder, 2016). In any case, the binary variable $s \in \mathbb{S}_\ell^f$ represents the sparsity pattern in the monomials, i.e., which ℓ monomials are used out of f potential candidates. The ultimate constraint of formulation (6) encodes the hierarchical nature of our (k, ℓ) -sparsity requirement. Only those monomials m_j such that the input $h_i = 1$ is selected for all its ancestors $i \in A(j)$ are considered as potential regressors. In all, the binary constraint

$$(s, h) \in \mathbb{S}_{\ell, k}^{f, p} := \{s \in \mathbb{S}_\ell^f, h \in \mathbb{S}_k^p : s_j \leq h_i, \forall i \in A(j), \forall j \in [f]\}$$

hence represents the postulated hierarchical (k, ℓ) -sparsity pattern.

To use the example discussed before in which we have three inputs and monomials of order two, the monomial $m_7(x) = \sqrt{2}x_1x_3$ can only be included as a regressor if the variable $s_7 = 1$. The variable s_7 can only then be nonzero if both inputs x_1 and x_3 are selected which requires that the variables $h_1 = 1$ and $h_3 = 1$. The resulting optimal regressor polynomial $h^*(x) = \sum_{j \in [f]} w_j^* \cdot m_j(x)$ thus counts at most ℓ monomials depending on at most k regressor inputs.

Although the direct formulation (6) of the hierarchical (k, ℓ) -sparse regression problem results in a well posed MIO problem, the constant \mathcal{M} needs to be chosen with extreme care as not to impede its numerical solution. The choice of this data dependent constant \mathcal{M} indeed affects the strength of the MIO formulation (6) and is critical for obtaining solutions quickly in practice (Bertsimas, King, and Mazumder, 2016). Furthermore, as the regression dimension p grows, explicitly constructing the MIO problem (6), let alone solving it, becomes burdensome. In order to develop an exact scalable method a different perspective on sparse regression is needed. In the subsequent sections we develop an exact method which avoids using a big- \mathcal{M} formulation while at the same time avoiding explicit construction of the problem at solve time.

2.2 Convex Integer Formulation

We next establish that the sparse regression problem (6) can in fact be represented as a pure binary optimization problem. By doing so we will eliminate the dependence of our formulation on the data dependent constant \mathcal{M} . Our results here closely resemble those presented in Bertsimas and Van Parys (2017). We will need the help of the following supporting lemma regarding linear regression.

Lemma 1 (The regression loss function c). *The least-squares regression cost $c(ZZ^\top) := \min_w \frac{1}{2} \|Y - Zw\|^2 + \frac{1}{2\gamma} \|w\|^2$ admits the following explicit characterization*

$$c(ZZ^\top) = \frac{1}{2} Y^\top (\mathbb{1}_n + \gamma ZZ^\top)^{-1} Y. \quad (7)$$

Proof. As the regression problem over w in \mathbb{R}^p is an unconstrained **qo!** (**qo!**) problem, the optimal value w^* satisfies the linear relationship $(\mathbb{1}_p/\gamma + Z^\top Z)w^* = Z^\top Y$. Substituting the expression for the optimal linear regressor w^* back into optimization problem, we arrive at

$$c(ZZ^\top) = 1/2 Y^\top Y - 1/2 Y^\top Z (\mathbb{1}_p/\gamma + Z^\top Z)^{-1} Z^\top Y.$$

The final characterization can be derived from the previous result with the help of the matrix inversion lemma found stating the identity $(\mathbb{1}_n + \gamma ZZ^\top)^{-1} = \mathbb{1}_n - Z (\mathbb{1}_p/\gamma + Z^\top Z)^{-1} Z^\top$. \square \square

Lemma 1 will enable us to eliminate the continuous variable w out of the MIO sparse regression formulation (6). The following result provides a different pure integer approach to hierarchical sparse regression. It will form the basis to our attempts to solve hierarchical regression problems.

Theorem 1 (Hierarchical (k, ℓ) -sparse regression). *The hierarchical (k, ℓ) -sparse regression problem (2) can be reformulated as the pure Convex Integer Optimization (CIO) problem*

$$\begin{aligned} \min \quad & \frac{1}{2} Y^\top \left(\mathbb{1}_n + \gamma \sum_{j \in [f]} s_j K_j \right)^{-1} Y \\ \text{s.t.} \quad & s \in S_\ell^f, \quad h \in S_k^p, \\ & s_j \leq h_i \quad \forall i \in A(j), \quad \forall j \in [f], \end{aligned} \quad (8)$$

where the micro kernel matrices K_j in S_\pm^n are defined as the dyadic outer products $K_j := m_j(X) \cdot m_j(X)^\top$.

Proof. We start the proof by separating the optimization variable w in the sparse regression problem (6) into its support $s := \text{supp } w$ and the corresponding non-negative entries w_s . Evidently, we can now write the sparse regression problem (6) as the bilevel minimization problem

$$\min_{s, h} \left[\min_{w \in \mathbb{R}^k} \frac{1}{2\gamma} \|w\|^2 + \frac{1}{2} \sum_{t \in [n]} \|y_t - \sum_{\{j \in [f] : s_j=1\}} w_j \cdot m_j(x_t)\|^2 \right]. \quad (9)$$

It now remains to be shown that the inner minimum can be found explicitly as the objective function of the optimization problem (8). Using Lemma 1, the minimization problem can be reduced to the minimization problem $\min\{c(m_s(X) \cdot m_s(X)^\top) : (s, h) \in S_{k, \ell}^{p, r}\}$. We finally remark that the outer product can be decomposed as the sum $m_s(X) \cdot m_s(X)^\top = \sum_{j \in [p]} s_j \cdot m_j(X) \cdot m_j(X)^\top$, thereby completing the proof. $\square \square$

Bertsimas and Van Parys (2017) provide an algorithm which can solve a related sparse linear regression problem up to dimensions f and n in the order of 100,000s based on a cutting plane formulation for integer optimization. Contrary to traditional complexity theory which suggests that the difficulty of a problem increases with size, there exists a critical number of observations n_0 such that the sparse regression problems seem to have the property that for a small number of samples $n < n_0$, an exact regressor is not easy to obtain, but most importantly its solution does not recover the truth ($A\% \approx 0$ and $F\% \approx 100$). For a large number of samples $n > n_0$ however, exact sparse regression can be done extremely fast and perfectly separates ($A\% \approx 100$ and $F\% \approx 0$) the true monomial features from the obfuscating bulk. These results warrant the possibility of nonlinear feature discovery for regression task of practical size as well.

Despite the previous encouraging results, for all but midsize problems hierarchical sparse regression quickly becomes problematic. The effective number of regression features f is indeed combinatorial in the number of inputs p and degree r of the considered polynomials. Our key insight is to triage the inputs first heuristically using an efficient input ranking method described in the subsequent section. Later in Section 5 we will show that this two-step procedure outlined in Figure 1 goes a long way to solve practical hierarchical sparse regression problems.

3 Polynomial Kernel Input Ranking

The objective in this section is to present an efficient method which can address the high-dimensional nature of exact sparse regression by ignoring irrelevant regression inputs and working with promising candidates only. The reader might wonder at this point whether any such attempt would not entail the solution of the original hierarchical sparse regression problem. Here, however, we do not claim to triage the inputs optimally, but rather aim for a simple approximate yet fast method. We will attempt to do so by leveraging the fact that the nonlinear regression problem (1) without sparse constraints can be solved efficiently.

A seminal result due to Vapnik (1998) states that the feature dimensionality f of the unconstrained regression problem (1) surprisingly does not play any role in its numerical solution. Indeed, the feature dimensionality f can be done away with in its entirety using the now classical Mercer (1909) kernel representation theorem. We can state the polynomial regression problem (1) as an optimization problem in terms of coefficients in the monomial basis

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{t \in [n]} \|y_t - \sum_{j \in [f]} w_j \cdot m_j(x_t)\|^2 + \frac{1}{2\gamma} \|w\|^2 \\ \text{s.t.} \quad & w \in \mathbb{R}^f. \end{aligned} \tag{10}$$

We state the Mercer kernel representation in Theorem 2 for the sake of completeness regarding the dual of the regression problem (10). It should be noted that surprisingly the dimension f does not play a role but instead the number of samples n is of importance. This previous observation is what has propelled kernel learning algorithms as viable nonlinear regression methods (Schölkopf and Smola, 2002).

Theorem 2 (Mercer Kernel Representation (Vapnik, 1998)). *The polynomial regression problem (10) can equivalently be formulated as the unconstrained maximization problem*

$$\begin{aligned} c(K) = \max \quad & -\frac{\gamma}{2} \alpha^\top K \alpha - \frac{1}{2} \alpha^\top \alpha + Y^\top \alpha \\ \text{s.t.} \quad & \alpha \in \mathbb{R}^n, \end{aligned} \tag{11}$$

where the positive semidefinite kernel matrix $K := m(X) \cdot m(X)^\top$ allows for an efficient characterization as the Hadamard power $K = (XX^\top + \mathbb{1}_{n \times n})^{\circ r}$.

Theorem 2 uses the Mercer kernel representation which establishes that the outer product $m(X) \cdot m(X)^\top$ can be characterized as the element-wise Hadamard power $(XX^\top + \mathbb{1}_{n \times n})^{\circ r}$ for our specific polynomial bases.

Indeed, for any t and t' in $[n]$ we have

$$\begin{aligned}
K(t, t') &:= m(x_t)^\top \cdot m(x_{t'}) \\
&= [\sqrt{\|A(1)\|} \cdot m_1(x_t), \dots, \sqrt{\|A(f)\|} \cdot m_f(x_t)]^\top \cdot [\sqrt{\|A(1)\|} \cdot m_1(x_{t'}), \dots, \sqrt{\|A(f)\|} \cdot m_f(x_{t'})] \\
&= \sum_{j \in [f]} \|A(j)\| m_j(x_t) \cdot m_j(x_{t'}) \\
&= \sum_{j \in [f]} \|A(j)\| m_j([x_{t,1} \cdot x_{t',1}, \dots, x_{t,p} \cdot x_{t',p}]) \\
&= (1 + \sum_{i \in [p]} x_{t,i} \cdot x_{t',i})^r \\
&= (1 + x_t^\top x_{t'})^r
\end{aligned}$$

where $\|A(j)\|$ denotes here the number of distinct ways to order the multiset $A(j)$. The penultimate equality is recognized as the binomial expansion theorem. Note that for the Mercer kernel representation to hold, the monomial basis had indeed to be properly normalized using $\sqrt{\|A(j)\|}$ as explained in the beginning of Section 2. This well known but crucial observation seems to have been made first by Poggio (1975).

The size of the kernelized regression problem (11) scales only with the number of data points n rather than the feature dimension f . It could be remarked that as the kernelized regression problem is unconstrained it admits a closed form solution in the form of the linear system $(\mathbb{1}_n + \gamma K)\alpha^* = Y$. The optimal regression coefficients w^* in formulation (10) are linearly related to the optimal dual variable α^* in formulation (11) via the complementarity conditions which here read

$$w_j^* = \gamma \cdot m_j(X)^\top \alpha^*. \quad (12)$$

Although this last relationship is linear, computing the coefficients in the monomial basis might still prove a daunting task merely because of the sheer number of them. In the following, we show that we can nevertheless compute the Euclidean norm of the optimal coefficients w_j^* in front of the monomials depending on a certain input i efficiently.

Proposition 1. *The Euclidean norm of the coefficients w_j^* in front of all monomials m_j depending on input i is related to the dual optimal variable α^* in (11) as*

$$\|w_{D(i)}^*\|^2 = \sum_{j \in D(i)} \|w_j^*\|^2 = \gamma^2 \cdot \alpha^{*\top} K_i \alpha^*, \quad (13)$$

where the kernel matrix K_i can be characterized explicitly as $K - (XX^\top - X_i X_i^\top + \mathbb{1}_{n \times n})^{\text{or}}$.

Proof. From the linear relationship (12) between the optimal coefficients w^* and dual variable α it follows immediately that $\|w_j\|^2 = \gamma^2 \cdot \alpha^{*\top} (\sum_{j \in D(i)} K_j) \alpha^*$. Through simple expansion it is quite easy to see that $(XX^\top - X_i X_i^\top + \mathbb{1}_{n \times n})^{\text{or}}$ coincides exactly with $\sum_{j \notin D(i)} K_j$. Hence, the kernel matrix $K_i := \sum_{j \in D(i)} K_j$ is found as its complement $K - (XX^\top - X_i X_i^\top + \mathbb{1}_{n \times n})^{\text{or}}$. \square \square

Hence, despite the fact that the size of the optimal coefficients w_j in front of the monomials depending on a certain input i consists of the sum of squares of as many as $\binom{p+r-1}{r-1}$ components, it can nevertheless be computed without much effort. Unfortunately, the optimal regressors coefficients w^* in (10) are not expected to be sparse. Nevertheless, the optimal regressors coefficients can be used to provide a ranking of the importance of the p data inputs. The Euclidean norm of the coefficients of the monomials which depend on input i can indeed be used as a proxy for the relevance of the input of interest. Fortunately, the quantities (13) are very efficient to compute once the optimal dual variable α^* has been found. Indeed, each quantity can be computed in $\mathcal{O}(n^2)$ time independent of the dimension f of the polynomials considered.

The complete computation is given in Algorithm 1. Though not exact, it gives a good indication of the significance of each of the p inputs. In fact it is very closely related the backward elimination wrapper methods discussed in (Guyon and Elisseeff, 2003).

An alternative way of looking at our input ranking algorithm is through the subgradients of the convex regression loss function c defined in the dual problem (11). We note that the subgradient of the function c can be computed explicitly using its dual characterization given in (11) as well.

Algorithm 1: Input Ranking

input : $Y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$ and $r \in \mathbb{N}$ **output:** $r \in \mathbb{R}^p$ $K \leftarrow (XX^\top + \mathbb{1}_{n \times n})$ $\alpha^* = (\mathbb{1}_n + \gamma K^{\circ r})^{-1} Y$ **for** i *in* $[p]$ **do**

$K_i \leftarrow K^{\circ r} - (K - X_i X_i^\top)^{\circ r}$
$r_i \leftarrow \gamma^2 \cdot \alpha^{*\top} K_i \alpha^*$

Proposition 2 (Derivatives of the optimal regression loss function). *We have that the subgradient of the regression loss function c as a function of the kernel K can be stated as*

$$\nabla c = -\frac{\gamma}{2} \cdot \alpha^{*\top} \nabla K \alpha^*,$$

where α^ maximizes (11).**Proof.* From the dual definition of the regression loss function c it is clear that we have the inequality

$$c(\bar{K}) \geq -\frac{\gamma}{2} \cdot \alpha^{*\top} \bar{K} \alpha^* - \frac{1}{2} \alpha^{*\top} \alpha^* + Y^\top \alpha^* \tag{14}$$

for all \bar{K} . From the very definition of α^* as the maximizer of (11) it follows that the previous inequality becomes tight for $\bar{K} = K$. This proves that the left hand side of (14) is a subgradient to the regression loss function c at the point K . □ □

When comparing the previous proposition with the result in Theorem 1, it should be noted that the derivatives of c at K agree up to the constant -2γ with the sum of squares of the coefficients w^* of the optimal polynomial. In essence thus, our Algorithm 1 ranks the inputs according to the linearized loss of regression performance ∇c_i caused by ignoring the inputs using the polynomial regression (1). The quantity r_i characterized up to first-order the loss in predictive power when not using the input i . The higher this caused loss, the more importance is assigned to including input i as a regressor.

As was noted in the beginning of the section, the input ranking method presented in Algorithm 1 does not aspire to find all k relevant inputs exactly. Rather, it is only meant to eliminate the most unpromising inputs and keep p' high potential candidates as illustrated in Figure 1. Among those inputs which are deemed promising we will then solve the hierarchical (k, ℓ) sparse regression problem (2) exactly as explained in the subsequent section.

4 A Cutting Plane Algorithm for Hierarchical Sparse Regression

We point out again that our pure integer formulation (8) of the hierarchical (k, ℓ) -sparse regression problem (2) circumvents the introduction of a big- \mathcal{M} constant which is simultaneously hard to estimate and crucial for its numerical efficacy. Nevertheless, explicitly constructing the optimization problem (8) results in the integer semidefinite optimization problem

$$\min_{(s,h) \in S_{\ell,k}^{f,p}} c\left(\sum_{j \in [f]} s_j K_j\right)$$

which might prove daunting. The regression loss function c is indeed a semidefinite representable function (Nesterov and Nemirovskii, 1994) to be optimized over the discrete set $S_{\ell,k}^{f,p}$. Without even taking into account the discrete nature of the optimization problem (8), solving a **sdo!** (**sdo!**) of size the number of samples might even prove in a convex case impractical for medium size problems with $n \approx 1,000$. In order

to solve our CIO formulation (8), we take here an alternative route using the outer approximation approach introduced by Duran and Grossmann (1986).

The outer approximation algorithm proceeds to find a solution to the CIO problem (8) by constructing a sequence of piecewise affine approximations c^a to the loss function c based on cutting planes. From the pseudocode given in Algorithm 2, the outer approximation Algorithm 2 can be seen to construct an increasingly better piece-wise affine lower approximation to the convex regression loss function c using the subgradients defined in Proposition 2. At each iteration indeed, the cutting plane added $\eta \geq c(s) + \nabla c(s^a)(s - s^a)$ in the outer approximation Algorithm 2 cuts off the current binary solution s^a unless it happened to be optimal in (8). As the algorithm progresses, the outer approximation function c^a thus constructed

$$c^a(\sum_{j \in [f]} s_j K_j) := \max_{u \in [a]} c(\sum_{j \in [f]} s_j^u K_j) + \nabla c(\sum_{j \in [f]} s_j^u K_j)(\sum_{j \in [f]} (s - s_j^u) K_j)$$

becomes an increasingly better approximation to our regression loss function of interest. Unless the current binary solution s^a is optimal in (8), a new distinct cutting plane will refine the approximation. The main advantage of working with the approximations c^a instead of c is that the former results in linear integer optimization instead of the much more tedious semidefinite integer optimization problem over $S_{\ell,k}^{f,p}$.

Theorem 3 (Exact Sparse Regression (Fletcher and Leyffer, 1994)). *Algorithm 2 returns the exact sparse solution w^* of the hierarchical (k, ℓ) -sparse regression problem (6) in finite time.*

Algorithm 2: The outer approximation process

input : $Y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$ and $k \in [1, p]$

output: $s^* \in S_k^p$ and $w^* \in \mathbb{R}^p$

$s_1 \leftarrow$ warm start

$\eta_1 \leftarrow 0$

$a \leftarrow 1$

while $\eta_a < c(s_a)$ **do**

$s_{a+1}, \eta_{a+1} \leftarrow \arg \min_{s, \eta} \{ \eta \in \mathbb{R}_+ \text{ s.t. } (s, h) \in S_{\ell,k}^{f,p}, \eta \geq c(s^u) + \nabla c(s^u)(s - s^u), \forall u \in [a] \}$
 $a \leftarrow a + 1$

$s^* \leftarrow s^a$

$w^* \leftarrow 0, \quad w_{s^*}^* \leftarrow (\mathbb{I}_p / \gamma + X_{s^*}^\top X_{s^*})^{-1} X_{s^*}^\top Y$

Despite the previous encouraging corollary, it nevertheless remains the case that from a theoretical point of view we may need to compute exponentially many cutting planes in the worst-case, thus potentially rendering our approach impractical. Indeed, in the worst-case Algorithm 2 considers all integer point in $S_{\ell,k}^{f,p}$ forcing us to minimize the function so constructed

$$\bar{c}(s) := \max_{(\bar{s}, \bar{h}) \in S_{\ell,k}^{f,p}} c(\bar{s}) + \nabla c(\bar{s})(s - \bar{s})$$

over the hierarchical binary constraint set $S_{\ell,k}^{f,p}$. As the number of integer points in the constraint set $S_{\ell,k}^{f,p}$ is potentially extremely large, the previous full explicit construction should evidently be avoided. In practice usually very few cutting planes need to be considered making the outer approximation method an efficient approach.

Furthermore, at each of the iterations in Algorithm 2 we need to solve the convex integer optimization problem $\min \{ c^a(\sum_{j \in [f]} s_j K_j) : (s, h) \in S_{\ell,k}^{f,p} \}$. This can be done by constructing a branch-and-bound tree, c.f. Lawler and Wood, 1966, which itself requires a potential exponential number of its leaves to be explored. This complexity behavior is however to be expected as exact sparse regression is known to be an NP-hard problem. Surprisingly, the empirical timing results presented in Section 5 suggest that the situation is much more interesting than what complexity theory might suggest. In what remains of this section, we briefly discuss a technique to carry out the outer approximation algorithm more efficiently than a naive implementation would.

In general, outer approximation methods such as Algorithm 2 are known as *multi-tree* methods because every time a cutting plane is added, a slightly different integer optimization problem is to be solved anew by constructing a branch-and-bound tree. Consecutive integer optimization problems $\min\{c^a(\sum_{j \in [f]} s_j K_j) : (s, h) \in S_{\ell, k}^{f, p}\}$ in Algorithm 2 differ only in one additional cutting plane. Over the course of our iterative cutting plane algorithm, a naive implementation would require that multiple branch and bound trees are built in order to solve the successive integer optimization problems. We implement a *single tree* way of solving the iteration algorithm 2 by using dynamic constraint generation, known in the optimization literature as either a lazy constraint or column generation method. Lazy constraint formulations described in Barnhart et al., 1998 dynamically add cutting planes to the model whenever a binary feasible solution is found. This saves the rework of rebuilding a new branch-and-bound tree every time a new binary solution is found in Algorithm 2. Lazy constraint callbacks are a relatively new type of callback. To date, the only commercial solvers which provide lazy constraint callback functionality are CPLEX, Gurobi and GLPK.

5 Numerical results

To evaluate the effectiveness of hierarchical sparse polynomial regression discussed in this paper, we report its performance first on synthetic sparse data and subsequently on real data from the UCI Machine Learning Repository as well. All algorithms in this document are implemented in Julia and executed on a standard Intel(R) Xeon(R) CPU E5-2690 @ 2.90GHz running CentOS release 6.7. All optimization was done with the help of the commercial mathematical optimization distribution Gurobi version 6.5 interfaced through the JuMP package developed by Lubin and Dunning (2015).

5.1 Benchmarks and Data

In the first part we will describe the performance of our cutting plane algorithm for polynomial sparse regression on synthetic data. We first describe the properties of the synthetic data in more detail.

Synthetic data: The synthetic observations Y and input data X satisfy the linear relationship

$$\begin{aligned} Y &= g_{\text{true}}(X) + E \\ &= m(X) \cdot w_{\text{true}} + E. \end{aligned}$$

The unobserved true regressor w_{true} has exactly ℓ nonzero components at indices j selected uniformly at random without replacement from \mathcal{J} . The previous subset \mathcal{J} is itself furthermore constructed randomly as $\cup_{i \in \mathcal{I}} D(i)$ where the k elements in \mathcal{I} are uniformly selected out of $[p]$. The previous discussed construction thus guarantees that the ground truth w_{true} is (k, ℓ) sparse. Additionally, the nonzero coefficients in w_{true} are drawn uniformly at random from the set $\{-1, +1\}$. The observation Y consists of the signal $S := Xw_{\text{true}}$ corrupted by the noise vector E . The noise components e_t for t in $[n]$ are drawn independent identically distributed (i.i.d.) from a normal distribution and scaled such that the signal-to-noise ratio equals

$$\sqrt{\text{SNR}} := \|S\|_2 / \|E\|_2.$$

Evidently as the signal-to-noise ratio SNR increases, recovery of the unobserved true regressor w_{true} from the noisy observations can be done with higher precision. We have yet to specify how the input matrix X is chosen. We assume here that the input data samples $X = (x_1, \dots, x_n)$ are drawn from an i.i.d. source with Gaussian distribution. Although the columns of the data matrix X are left uncorrelated, the features $m_j(X)$ will be correlated. For instance, it is clear that the second and forth power of the first input can only be positively correlated.

We will compare the performance of hierarchical sparse regression with two other benchmark regression approaches. These two approaches were chosen as to investigate the impact of both sparsity and nonlinearity on the performance of our method. Next we describe the particularities of these two benchmarks more closely.

Polynomial Kernel Regression: As a first benchmark we consider polynomial regression defined explicitly in (1). The primary advantage of this formulation stems from the fact that the optimal polynomial regressor in (1) can be found efficiently using

$$g_2^*(x) = \gamma \sum_{t \in [n]} \alpha_t^* (x^\top x_t + 1)^r$$

where α^* is the maximizer of (11). As this formulation does not yield sparse regressors, this benchmark will show us the merit of sparsity in terms of prediction performance. Classical Ridge regression is found as a special case for $r = 1$, enabling us to see what fruits nonlinearity brings us.

ℓ_1 -Heuristic Regression: In order to determine the effect of exact sparse regression in our two step procedure, we will also use a close variant of the **SPORE** algorithm developed by Huang et al. (2010) as a benchmark. Using the input ranking method discussed in Section 3, we determine first the p' most relevant inputs heuristically. Using the remaining inputs $X' \in \mathbb{R}^{n \times p'}$ and response data $Y \in \mathbb{R}^n$ we then consider the maximizer g_1^* of (3) as a heuristic sparse regressor. This two-step regression procedure hence shares the structure outlined in Figure 1 with our hierarchical exact sparse regression algorithm. As to have a comparable number of hyper parameters as our two-step approach, we finally perform Ridge regression on the thus selected features using a Tikhonov regularization parameter γ selected using cross validation.

Theoretical considerations (Bühlmann and Geer, 2011; Hastie, Tibshirani, and Wainwright, 2015; Wainwright, 2009) and empirical evidence (Donoho and Stodden, 2006) suggests that the ability to recover the support of the correct regressor w_{true} from noisy data using the **Lasso** heuristic experiences a phase transition. While it is theoretically understood (Gamarnik and Zadik, 2017) that a similar phase transition must occur in case of exact sparse regression, due to a lack of scalable algorithms such a transition was never empirically reported. The scalable cutting plane algorithm developed in Section 4 offers us the means to do so however. Our main observation is that exact regression is significantly better than convex heuristics such as **Lasso** in discovering all true relevant features ($A\% \approx 100$), while truly outperforming their ability to reject the obfuscating ones ($F\% \approx 0$).

5.2 Phase Transitions

In Figure 2 we depict the performance of the hierarchical sparse regression procedure outlined in Figure 1 in its ability to discover all relevant regression features ($A\%$) and the running time T in seconds as a function of the sample size n for a regression problem with $p' = p = 25$ inputs expanded with the help of all cubic monomials into $f = 3276$ possible features. As $p' = p$ the input ranking heuristic is irrelevant here and instead all inputs are considered by the exact sparse regression procedure. The reported results are the average of 20 independent sparse synthetic data sets where the error bars visualize the inter data set variation. For the purpose of this section, we assume that we know that only $\ell = 20$ features are relevant but do not know which. In practice though, the parameter ℓ must be estimated from data. In order to play into the ballpark of the **Lasso** method, no hierarchical structure ($k = p$) is imposed. This synthetic data is furthermore lightly corrupted by Gaussian noise with $\sqrt{SNR} = 20$. Furthermore, if the optimal sparse regressor was not found by the outer approximation Algorithm 2 within two minutes, the best solution found up to that point is considered.

It is clear that our ability to uncover all relevant features ($A\% \approx 100$) experiences a phase transition at around $n_0 \approx 600$ data samples. That is, when given more than n_0 data points, the accuracy of the sparse regression method is perfect. With fewer data points our ability to discover the relevant features quickly diminishes. For comparison, we also give the accuracy performance of the **Lasso** heuristic described in (3). It is clear that exact sparse regression dominates the **Lasso** heuristic and needs fewer samples for the same accuracy $A\%$.

Especially surprising is that the time T it takes Algorithm 2 to solve the corresponding (k, ℓ) -sparse problems exactly experiences the same phase transition as well. That is, when given more than n_0 data points, the accuracy of the sparse regression method is not only perfect but easy to obtain. In fact, in case $n > n_0$ our method is empirically as fast as the **Lasso** based heuristic. This complexity transition can be characterized

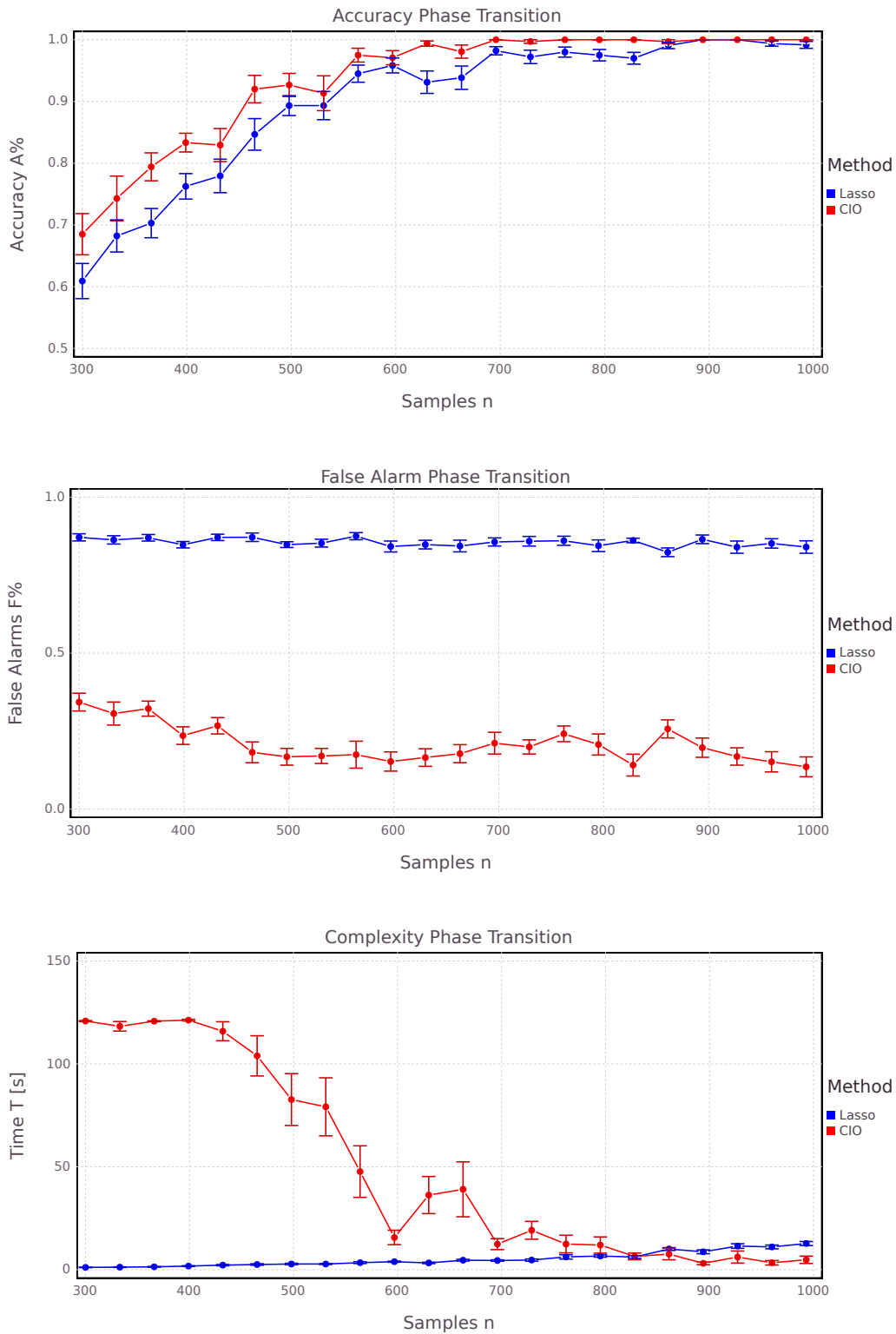


Figure 2: The performance of exact sparse regression and the Lasso heuristic on synthetic data in terms of accuracy $A\%$, false alarm rate $F\%$ and time T in seconds.

Samples n	300	400	500	600	700	800	900	1000
Cutting planes	> 300	> 300	298	200	70	59	25	31

Table 1: Number of cutting planes considered in the outer approximation Algorithm 2 as a function of the sample size n . For the smallest sample sizes n the optimal solution could not be computed within the allocated maximum solution time.

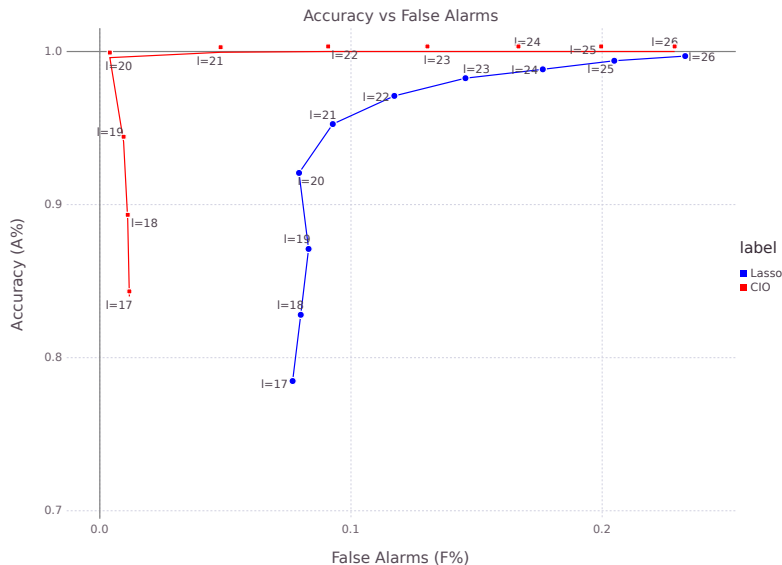


Figure 3: The performance of exact sparse regression and the **Lasso** heuristic on synthetic data in terms of accuracy $A\%$ and false alarms $F\%$.

equivalently in terms of the number of cutting planes necessary for our outer approximation Algorithm 2 to return the optimal hierarchical sparse regressor. While potentially exponentially many ($|S_{\ell,k}^{f,p}|$ in fact) cutting planes might be necessary in the worst-case, Table 1 list the actual average number of cutting planes considered on the twenty instances previously discussed in this section. When $n > n_0$ only a few cutting planes suffice, whereas for $n < n_0$ an exponential number seem to be necessary.

5.3 The whole truth, and nothing but the truth

In the previous section we demonstrated that exact sparse regression is marginally better in discovering all relevant features $A\%$. Nevertheless, the true advantage of exact sparse regression in comparison to heuristics such as **Lasso** is found to lie elsewhere. Indeed, both our method and the **Lasso** heuristic are with a sufficient amount of data eventually able to discover all relevant ($A\% \approx 100$) features. In this section, we will investigate the ability of both methods to reject irrelevant features. Indeed in order for a method to tell the truth, it must not only tell the whole truth ($A\% \approx 100$), but nothing but the truth ($F\% \approx 0$). It is in the latter aspect that exact sparse regression truly shines.

In Figure 3 we show the performance of the (p, ℓ) -sparse regressors found with our exact method and the **Lasso** heuristic on the same synthetic data discussed in the previous section in terms of both their accuracy $A\%$ and false alarm rate $F\%$ in function of ℓ . Again, the reported results are averages over 20 distinct synthetic data sets. Each of these data sets consisted of $n = 560$ observations. Among all potential third degree monomial features, again only 20 were chosen to be relevant for explaining the observed data Y . Whereas in previous section, the true number of features was treated as a given, in practical problems ℓ must also be estimated from data. As we vary ℓ over the regression path $[f]$, we implicitly trade lower false

Dimension p'	$n = 2 \cdot 10^3$	$n = 4 \cdot 10^3$	$n = 6 \cdot 10^3$	$n = 8 \cdot 10^3$	$n = 10 \cdot 10^3$
$p = 200$	72	41	36	54	27
$p = 400$	117	109	75	80	62
$p = 600$	222	166	128	68	104
$p = 800$	361	230	96	95	102
$p = 1000$	286	264	142	103	78

Table 2: Average size p' necessary for our input ranking heuristic to identify all relevant features.

alarm rates for higher accuracy. We have indeed a choice between including too many features in our model resulting in a high false alarm rate but hopefully discovering many relevant features, or limiting the number of features thus keeping the false alarm rate low but at the cost of missing features. One method is better than another when it makes this tradeoff better, i.e., obtains higher accuracy for a given false alarm rate or conversely a lower false alarm rate for the same accuracy. It is clear from the results shown in Figure 3 that exact sparse regression dominates the **Lasso** heuristic in terms of keeping a smaller false alarm rate while at the same time discovering more relevant monomial features.

Hence although both exact sparse regression and the **Lasso** heuristic are eventually capable to find all relevant monomial features, only the exact method finds a truly sparse regressor by rejecting most irrelevant monomials from the obfuscating bulk. In practical situations where interpretability of the resulting regression model is key, the ability to reject irrelevant features can be a game changer. While all the results so far are demonstrated on synthetic data, we shall argue in the next section that also for real data sets similar encouraging conclusions can be drawn.

5.4 Polynomial Input Ranking

In the preceding discussions, the first step of our approach outlined in Figure 1 did not come into play as it was assumed that $p' = p$. Hence, no preselection of the inputs using the ranking heuristic took place. Because of the exponential number of regression features f as a function of the input dimension p , such a direct approach may not be tractable when p becomes large. In this part we will argue that the performance of the input ranking heuristic discussed in Section 3 is sufficient in identifying the relevant features while ignoring the obfuscating bulk.

To make our case, we consider synthetic data with hierarchical sparsity $(k, \ell) = (20, 40)$. That is, only 20 inputs in 40 relevant degree $r = 3$ monomial features are relevant for the purpose of regression. In Table 2 we report the average reduced input dimension p' necessary for the input ranking heuristic to recover all relevant inputs. That is, all relevant k inputs are among the top p' ranked inputs. For example, for $n = 2,000$ and $p = 1,000$ the input ranking heuristic needs to include $p' = 286$ to cover all 20 true features, while for $n = 10,000$ and $p = 1,000$ the input ranking heuristic needs to include only $p' = 78$. We note that as n increases the input ranking heuristic needs a smaller number of features to identify the relevant ones. Note, however, that the false alarm rate of the input ranking heuristic remains high, that is, among the top p' inputs many inputs were in fact irrelevant ($p' > k$). However, we do not aspire here to find all k relevant inputs exactly, rather we hope to reduce the dimension to p' without missing out any relevant inputs. Reducing the false alarm rate is done by means of exact hierarchical sparse regression in the second step of our overall algorithm.

5.5 Real Data Sets

In the final part of the paper we report the results of the presented methods on several data sets found in the **UCI Machine Learning Repository** found under <https://archive.ics.uci.edu/ml/datasets.html>. Each of the datasets was folded ten times into 80% training data, 10% validation data and 10% test data \mathcal{T} . No preprocessing was performed on the data besides rescaling the inputs to have a unit norm.

We report the prediction performance of four regression methods on each of the test data sets. The first

			RR	SVM		PL			CIO			
Problem	n	p	TE	r^*	TE	r^*	ℓ^*	TE	r^*	k^*	ℓ^*	TE
Brooks, Pope, and Marcolini	1502	5	4.62	3	4.04	4	65	3.61	4	5	100	3.53
Yeh	1029	8	10.2	3	6.00	4	94	6.11	3	8	100	6.27
Tsanas and Xifara I	768	8	2.79	4	2.40	4	49	2.52	4	8	100	2.37
Tsanas and Xifara II	768	8	3.12	4	1.72	4	50	2.97	4	6	70	2.79
Cortez et al.	1599	11	0.65	2	0.70	4	70	0.69	3	7	95	0.63
Cortez et al.	4898	11	0.74	2	0.72	3	95	0.70	3	11	95	0.70
Zhou, Claire, and King I	1058	68	16.4	2	20.3	2	30	16.5	4	18	65	18.5
Zhou, Claire, and King II	1058	68	44.0	2	50.4	2	37	44.8	2	20	25	45.8

Table 3: Out-of-sample performance of Ridge regression (RR), polynomial kernel regression (SVM), polynomial lasso (PL) and exact hierarchical regression (CIO) on several UCI Data sets. We give the optimal hyper parameters for each of these methods as well.

regression method we consider is ordinary Ridge regression. Ridge regression allows us to find out whether considering nonlinear regressors has merit. The second method we consider is polynomial kernel regression with degree $r > 1$ polynomials as in (1). This nonlinear but non-sparse method on its part will allow us to find out the merits of sparsity for the purpose of out-of-sample prediction performance. The third method is the ℓ_1 -heuristic described before and which will allow us to see whether exact sparse formulations bring any benefits. We used the input ranking algorithm described in Section 3 to limit the number of potentially relevant inputs to at most $p' = 20$. Each of these methods is compared to our hierarchical exact regressor in terms of out-of-sample test error

$$\text{TE} := \frac{\sum_{t \in \mathcal{T}} \|y_t - h(x_t)\|_2}{\sqrt{|\mathcal{T}|}}.$$

The hyperparameters of each method were chosen as those best performing on the validation data from among $k \in [p']$, $\ell \in [100]$ and polynomial degree ranging in $r \in [4]$. The average out-of-sample performance on the test data and sparsity of the obtained regressors using the **Lasso** heuristic and exact hierarchical sparse regression is shown in Table 3.

As one could expect, considering nonlinear monomial features is not beneficial for prediction in all data sets. Indeed, in two data sets ordinary Ridge regression provides the best out-of-sample performance. A similar remark can be made with regards to sparsity. That is, for three data sets adding sparsity does not immediately yield any benefits in terms of prediction power. Nevertheless, in those situations where nonlinear and sparse regression is beneficial the results again point out that there is a benefit to exact sparse regression rather than heuristic approaches.

6 Conclusions

We discussed a scalable hierarchical sparse regression method based on a smart heuristic and modern integer optimization for nonlinear regression. We consider as the best regressor that degree r polynomial of the input data which depends on at most k inputs counting at most ℓ monomial terms which minimizes the sum of squares prediction errors with a Tikhonov loss. This hierarchical sparse specification aligns well with big data settings where many inputs are not relevant for prediction purposes and the functional complexity of the regressor needs to be controlled to avoid overfitting. Using a modern cutting plane algorithm, we can use exact sparse regression on regression problems of practical size. The ability of our method to identify all k relevant inputs as well as all ℓ relevant monomial terms and reject all others was shown empirically to experience a phase transition. In the regime where our method is statistically powerful, the computational complexity of exact hierarchical regression was empirically on par with **Lasso** based heuristics taking away their main propelling justification. We have empirically shown that we can outperform heuristic methods in both finding all relevant nonlinearities as well as rejecting obfuscating ones.

Acknowledgements

The second author is generously supported by the Early Postdoc.Mobility fellowship P2EZP2 165226 of the Swiss National Science Foundation.

References

- Bach, F.R. (2008). “Consistency of the group Lasso and multiple kernel learning”. In: *Journal of Machine Learning Research* 9.Jun, pp. 1179–1225.
- Bach, F.R. (2009). “Exploring large feature spaces with hierarchical multiple kernel learning”. In: *Advances in neural information processing systems*, pp. 105–112.
- Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance (1998). “Branch-and-price: Column generation for solving huge integer programs”. In: *Operations Research* 46.3, pp. 316–329.
- Bertsimas, D. and M.S. Copenhaver (2017). “Characterization of the equivalence of robustification and regularization in linear and matrix regression”. In: *European Journal of Operational Research*.
- Bertsimas, D., A. King, and R. Mazumder (2016). “Best subset selection via a modern optimization lens”. In: *Annals of Statistics* 44.2, pp. 813–852.
- Bertsimas, D. and B. Van Parys (2017). “Sparse high-dimensional regression: Exact scalable algorithms and phase transitions”. In: *Submitted to the Annals of Statistics*.
- Brooks, T.F., D.S. Pope, and M.A. Marcolini (1989). *Airfoil self-noise and prediction*. Tech. rep. 19890016302. NASA.
- Bühlmann, P. and S. van de Geer (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Science & Business Media.
- Candès, E.J., J. Romberg, and T. Tao (2006). “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information”. In: *IEEE Transactions on Information Theory* 52.2, pp. 489–509.
- Cortez, P., A. Cerdeira, F. Almeida, T. Matos, and J. Reis (2009). “Modeling wine preferences by data mining from physicochemical properties”. In: *Decision Support Systems* 47.4, pp. 547–553.
- Donoho, D. and V. Stodden (2006). “Breakdown point of model selection when the number of variables exceeds the number of observations”. In: *International Joint Conference on Neural Networks*. IEEE, pp. 1916–1921.
- Duran, M.A. and I.E. Grossmann (1986). “An outer-approximation algorithm for a class of mixed-integer nonlinear programs”. In: *Mathematical Programming* 36.3, pp. 307–339.
- Fletcher, R. and S. Leyffer (1994). “Solving mixed integer nonlinear programs by outer approximation”. In: *Mathematical Programming* 66.1, pp. 327–349.
- Gamarnik, David and Ilias Zadik (2017). “High-Dimensional Regression with Binary Coefficients. Estimating Squared Error and a Phase Transition”. In: *arXiv preprint arXiv:1701.04455*.
- Guyon, I. and A. Elisseeff (2003). “An introduction to variable and feature selection”. In: *Journal of Machine Learning Research* 3.Mar, pp. 1157–1182.

- Hastie, T., R. Tibshirani, and M. Wainwright (2015). *Statistical Learning with Sparsity: the Lasso and Generalizations*. CRC Press.
- Hoerl, A.E. and R.W. Kennard (1970). “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1, pp. 55–67.
- Huang, L., J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik (2010). “Predicting execution time of computer programs using sparse polynomial regression”. In: *Advances in neural information processing systems*, pp. 883–891.
- Lawler, E.L. and D.E. Wood (1966). “Branch-and-bound methods: A survey”. In: *Operations Research* 14.4, pp. 699–719.
- Lubin, M. and I. Dunning (2015). “Computing in Operations Research Using Julia”. In: *INFORMS Journal on Computing* 27.2, pp. 238–248.
- Mallat, S.G. and Z. Zhang (1993). “Matching pursuits with time-frequency dictionaries”. In: *IEEE Transactions on Signal Processing* 41.12, pp. 3397–3415.
- Mercer, J. (1909). “Functions of positive and negative type, and their connection with the theory of integral equations”. In: *Philosophical Transactions of the Royal Society of London* 209, pp. 415–446.
- Nesterov, Y. and A. Nemirovskii (1994). *Interior-point polynomial algorithms in convex programming*. SIAM.
- Pelckmans, K., J.A.K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, and J. Vandewalle (2002). *LS-SVMlab: a Matlab/C toolbox for least squares support vector machines*. Tech. rep. K.U.Leuven.
- Poggio, T. (1975). “On optimal nonlinear associative recall”. In: *Biological Cybernetics* 19.4, pp. 201–209.
- Schölkopf, B. and A.J. Smola (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.
- Smith, K. (1918). “On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations”. In: *Biometrika* 12.1/2, pp. 1–85.
- Stone, M.H. (1948). “The generalized Weierstrass approximation theorem”. In: *Mathematics Magazine* 21.5, pp. 237–254.
- Suykens, J.A.K. and J. Vandewalle (1999). “Least squares support vector machine classifiers”. In: *Neural Processing Letters* 9.3, pp. 293–300.
- Tikhonov, A.N. (1943). “On the stability of inverse problems”. In: *Doklady Akademii Nauk SSSR* 39.5, pp. 195–198.
- Tsanas, Athanasios and Angeliki Xifara (2012). “Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools”. In: *Energy and Buildings* 49, pp. 560–567.
- Turlach, B.A. (1993). *Bandwidth Selection in Kernel Density Estimation: A Review*. Tech. rep. CORE discussion paper.
- Vapnik, V. (1998). “The support vector method of function estimation”. In: *Nonlinear Modeling*. Springer, pp. 55–85.
- Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.

- Wainwright, M.J. (2009). “Sharp thresholds for high-dimensional and noisy sparsity recovery using-constrained quadratic programming (Lasso)”. In: *IEEE Transactions on Information Theory* 55.5, pp. 2183–2202.
- Yeh, I.-C. (1998). “Modeling of strength of high-performance concrete using artificial neural networks”. In: *Cement and Concrete research* 28.12, pp. 1797–1808.
- Zhao, P., G. Rocha, and B. Yu (2009). “The composite absolute penalties family for grouped and hierarchical variable selection”. In: *The Annals of Statistics*, pp. 3468–3497.
- Zhou, F., Q. Claire, and R.D. King (2014). “Predicting the geographical origin of music”. In: *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, pp. 1115–1120.