Journal of Machine Learning Research 20 (2019) 1-39 Submitted 6/17; Revised 10/18; Published 1/19

Forward-Backward Selection with Early Dropping

Giorgos Borboudakis BORBUDAK@GMAIL.COM
Ioannis Tsamardinos TSAMARD.IT@GMAIL.COM
Computer Science Department, University of Crete
Gnosis Data Analysis

Editor: Isabelle Guyon

Abstract

Forward-backward selection is one of the most basic and commonly-used feature selection algorithms available. It is also general and conceptually applicable to many different types of data. In this paper, we propose a heuristic that significantly improves its running time, while preserving predictive performance. The idea is to temporarily discard the variables that are conditionally independent with the outcome given the selected variable set. Depending on how those variables are reconsidered and reintroduced, this heuristic gives rise to a family of algorithms with increasingly stronger theoretical guarantees. In distributions that can be faithfully represented by Bayesian networks or maximal ancestral graphs, members of this algorithmic family are able to correctly identify the Markov blanket in the sample limit. In experiments we show that the proposed heuristic increases computational efficiency by about 1-2 orders of magnitude, while selecting fewer or the same number of variables and retaining predictive performance. Furthermore, we show that the proposed algorithm and feature selection with LASSO perform similarly when restricted to select the same number of variables, making the proposed algorithm an attractive alternative for problems where no (efficient) algorithm for LASSO exists.

Keywords: Feature Selection, Forward Selection, Markov Blanket Discovery, Bayesian Networks, Maximal Ancestral Graphs

1. Introduction

The problem of feature selection (a.k.a. variable selection) in supervised learning tasks can be defined as the problem of selecting a minimal-size subset of the variables that leads to an optimal, multivariate predictive model for a target variable (outcome) of interest (Tsamardinos and Aliferis, 2003). Thus, the feature selection's task is to filter out irrelevant variables and variables that are superfluous given the selected ones (that is, weakly relevant variables, see (John et al., 1994; Tsamardinos and Aliferis, 2003)).

Solving the feature selection problem has several advantages. Arguably, the most important one is knowledge discovery: by removing superfluous variables it improves intuition and understanding about the data-generating mechanisms. This is no accident as solving the feature selection problem has been linked to the data-generating causal network (Tsamardinos and Aliferis, 2003). In fact, *it is often the case that the primary goal of data analysis is feature selection* and not the actual resulting predictive model. This is particularly true in medicine and biology where the features selected may direct future experiments and studies. Feature selection is also employed to reduce the cost of measuring the features to make op-

erational a predictive model; for example, it can reduce the monetary cost or inconvenience to a patient of applying a diagnostic model by reducing the number of medical tests and measurements required to perform on a subject for providing a diagnosis. Feature selection also often improves the predictive performance of the resulting model in practice, especially in high-dimensional settings. This is because a good-quality selection of features facilitates modeling, particularly for algorithms susceptible to the curse of dimensionality. There has been a lot of research on feature selection methods in the statistical and machine learning literature. An introduction to the topic, as well as a review of many, prominent methods can be found in (Guyon and Elisseeff, 2003), while the connections between feature selection, the concept of relevancy, and probabilistic graphical models is in (John et al., 1994; Tsamardinos and Aliferis, 2003).

We will focus on forward and backward selection algorithms, which are specific instances of stepwise methods (Kutner et al., 2004; Weisberg, 2005). These methods are some of the oldest, simplest and most commonly employed feature selection methods. An attractive property of stepwise methods is that they are very general, and are applicable to different types of data. For instance, stepwise methods using conditional independence tests can be directly applied to (a) mixed continuous and categorical predictors, (b) cross-sectional or time course data, (c) continuous, nominal, ordinal or time-to-event outcomes, among others, (d) with non-linear tests, such as kernel-based methods (Zhang et al., 2011), and (e) to heteroscedastic data using robust tests; many of the aforementioned tests, along with others have been implemented in the MXM R package (Lagani et al., 2017). The main drawback of forward selection is its computational cost. In order to select k variables, it performs $O(pk)$ tests for variable inclusion, where p is the total number of variables in the input data. This is acceptable for low-dimensional datasets, but becomes unmanageable with increasing dimensionality. Another issue is that forward selection suffers from multiple testing problems and thus may select a large number of irrelevant variables (Flom and Cassell, 2007).

In computer science, forward-backward selection has re-appeared in the context of Markov blanket discovery and Bayesian network learning (Margaritis and Thrun, 2000; Tsamardinos et al., 2003b; Margaritis, 2009), and has been shown to be optimal for distributions that can be faithfully represented by causal graphs. In the signal processing community forward selection is known as orthogonal least squares (Chen et al., 1989). Other algorithms, such as LASSO (Tibshirani, 1996), least-angle regression (LARS) (Efron et al., 2004), forward stagewise regression (FSR) (Efron et al., 2004) and orthogonal matching pursuit (OMP) (Pati et al., 1993; Davis et al., 1994) are all variations of the basic stepwise selection algorithm, while information-theoretic feature selection methods are all approximations of the forward phase using discrete data (Brown et al., 2012). A detailed comparison between LASSO, LARS and FSR is given in (Efron et al., 2004), a comparison between LARS and OMP can be found in (Hameed, 2012), while a comparison between OMP and forward selection (called orthogonal least squares) can be found in (Blumensath and Davies, 2007). We proceed with a brief high-level comparison of the above with the forward selection algorithm. All of the above methods select the next feature using some selection criterion and are equipped with a stopping criterion. Intuitively, they all select the feature that provides the most information for the errors (residuals) of the current model. Forward selection on the other hand, selects the feature that leads to a model providing

the most additional information, given all selected variables. In LASSO, both forward and backward steps can be performed at each iteration. After a feature is selected, forward selection and OMP create a new unrestricted model that also contains the newly selected feature. LASSO, LARS and FSR create a new model by updating the previous one, constraining the coefficients of the new model. LASSO has a stopping criterion based on the L1-norm of the coefficients of the current variables. *Given this synthetic view and connections between the algorithms, we would like to note that any extension to stepwise methods, such as the one proposed in this work, can be translated and directly applied with any of the above feature selection algorithms.*

In this work we extend the forward selection algorithm to deal with the problems above. In Section 3 we propose **early dropping**, a simple heuristic to speed-up forward selection, without sacrificing its quality and maintaining its theoretical guarantees. The idea is, in each iteration of the forward search, to filter out variables that are deemed conditionally independent of the target given the current set of selected variables. After termination, the algorithm is allowed to run up to K additional times, every time initializing the set of selected variables to the ones selected in the previous run. Finally, backward selection is applied on the result of the forward phase. We call this algorithm **Forward-Backward selection with Early Dropping** (FBED^K)[1]. This heuristic is inspired by the theory of Bayesian networks and maximal ancestral graphs (Spirtes et al., 2000; Richardson and Spirtes, 2002), and similar ideas have been successfully applied for feature selection (Aliferis et al., 2010). In Section 3.2 we show that (a) FBED^0 returns a superset of the adjacent nodes in any Bayesian network or maximal ancestral graph that faithfully represents the data distribution (if there exists one and assuming perfect statistical independence tests), (b) FBED^1 returns the Markov blanket of the data distribution, provided the distribution is faithful to a Bayesian network, and (c) FBED$^\infty$ returns the Markov blanket of the data distribution provided the distribution is faithful to a maximal ancestral graph, or equivalently, it is faithful to a Bayesian network where some variables are unobserved (latent). In the experimental evaluation presented in Section 4, we show that FBED^K is 1-2 orders of magnitude faster than FBS, while at the same time selecting a similar number of features and having similar predictive performance. In a comparison between different members of the FBED^K family and FBS we show that FBED^0 and FBED^1 also reduce the number of false variable selections, when the data consist of irrelevant variables only. We also investigated the behavior of FBED^K with increasing number of runs K, showing that a relatively small K is sufficient in most cases to reach optimal predictive performance. Afterwards, we compare FBED^K to FBS, feature selection with LASSO (Tibshirani, 1996) and to the Max-Min Parents and Children algorithm (MMPC) (Tsamardinos et al., 2003a) and show that it often has comparable predictive performance while selecting the fewest variables overall. Finally, we compare FBED^K to feature selection with LASSO (Tibshirani, 1996) when both algorithms are limited to select the same number of variables, showing that both algorithms perform similarly. This, along with the generality of FBED^K makes it an interesting alternative to LASSO, especially for problems where LASSO requires specialized algorithms (like the group LASSO algorithm for logistic regression (Meier et al., 2008), LASSO for mixed-

1. The early dropping heuristic has also used by an extension of FBED for Big Data settings and map-reduce architectures (Tsamardinos et al., 2018a). However, the main idea and motivation behind it, its theoretical properties and a thorough experimental evaluation are presented in this work.

effects linear models (Schelldorfer et al., 2011), and the LASSO and group LASSO methods for functional Poisson regression (Ivanoff et al., 2016)), which may be non-convex (as is the case for mixed-effects linear models (Schelldorfer et al., 2011) or for temporal-longitudinal data (Groll and Tutz, 2014; Tsagris et al., 2018)) and computationally demanding (taking several days to terminate on datasets with just 1000 predictors using Cox's proportional hazards model (Fan et al., 2010)).

2. Notation and Preliminaries

We start by introducing the notation and terminology used throughout the paper. We use upper-case letters to denote single variables (for example, X), and bold upper-case letters to denote sets of variables (for example, $\mathbf{Z}$). The terms variable, feature or predictor will be used interchangeably. We will use p and n to refer to the number of variables and samples in a dataset $\mathcal{D}$, respectively. The set of variables in $\mathcal{D}$ will be denoted as $\mathbf{V}$. The target variable (also called outcome) will be referred to as T. Next, we proceed with the basics about stepwise feature selection methods (Kutner et al., 2004; Weisberg, 2005).

2.1. Stepwise Feature Selection

Stepwise methods start with some set of selected variables and try to improve it in a greedy fashion, by either including or excluding a single variable at each step. There are various ways to combine those operations, leading to different members of the stepwise algorithmic family. Two popular members of the stepwise family are the **forward selection** and **backward selection** (also known as backward elimination) algorithms. Forward selection starts with a (usually empty) set of variables and adds variables to it, until some stopping criterion is met. Similarly, backward selection starts with a (usually complete) set of variables and then excludes variables from that set, again, until some stopping criterion is met. Typically, both methods try to include or exclude the variable that offers the highest performance increase. We will call each step of selecting (removing) a variable a forward (backward) **iteration**. Executing forward (backward) iterations until termination will be called a forward (backward) **phase** respectively. An instance of the stepwise family, which we focus on hereafter, is the **Forward-Backward Selection** algorithm (FBS), which first performs a forward phase and then a backward phase on the selected variables. This algorithm is not new; similar algorithms have appeared in the literature before (Margaritis and Thrun, 2000; Tsamardinos et al., 2003b; Margaritis, 2009).

FBS is shown in Algorithm 1. The function Perf evaluates a set of variables and returns their performance relative to some statistical model. Examples are the log-likelihood for logistic regression, the partial log-likelihood for Cox regression and the F-score for linear regression, or their AIC (Akaike, 1973) or BIC (Schwarz et al., 1978) penalized variants. The **selection criterion** C compares the performance of two sets of variables as computed by Perf. For instance, in the previous example C could perform a likelihood ratio test and use a predetermined significance level α to make a decision [2]; we will describe such selection criteria in the next subsection. We will use the predicates $\underset{\text{C}}{>}$, $\underset{\text{C}}{\geq}$ and $\underset{\text{C}}{=}$ to compare two sets

2. In general, the type of criteria used in practice are not limited to that. For example, one may also stop after a fixed number of variables have been selected.

Algorithm 1 Forward-Backward Selection (FBS)

Input: Dataset $\mathcal{D}$, Target T
Output: Selected Variables $\mathbf{S}$

$\mathbf{S} \leftarrow \emptyset$ *//Set of selected variables*
$\mathbf{R} \leftarrow \mathbf{V}$ *//Set of remaining candidate variables*

//Forward phase: iterate until $\mathbf{S}$ *does not change*
while $\mathbf{S}$ changes **do**
  //Identify the best variable V_{best} *out of all remaining variables* $\mathbf{R}$*, according to* Perf
  $V_{best} \leftarrow \underset{V \in \mathbf{R}}{\text{argmax}}\ \textsc{Perf}(\mathbf{S} \cup V)$
  //Select V_{best} *if it increases performance according to criterion* C
  if $\textsc{Perf}(\mathbf{S} \cup V_{best}) \underset{\text{C}}{>} \textsc{Perf}(\mathbf{S})$ **then**
    $\mathbf{S} \leftarrow \mathbf{S} \cup V_{best}$
    $\mathbf{R} \leftarrow \mathbf{R} \setminus V_{best}$
  end if
end while

//Backward phase: iterate until $\mathbf{S}$ *does not change*
while $\mathbf{S}$ changes **do**
  //Identify the worst variable V_{worst} *out of all selected variables* $\mathbf{S}$*, according to* Perf
  $V_{worst} \leftarrow \underset{V \in \mathbf{S}}{\text{argmax}}\ \textsc{Perf}(\mathbf{S} \setminus V)$
  //Remove V_{worst} *if it does not decrease performance according to criterion* C
  if $\textsc{Perf}(\mathbf{S} \setminus V_{worst}) \underset{\text{C}}{\geq} \textsc{Perf}(\mathbf{S})$ **then**
    $\mathbf{S} \leftarrow \mathbf{S} \setminus V_{worst}$
  end if
end while
return S

of variables; they are true if the left-hand-side value is greater, greater or equal, or equal than the right-hand-side value respectively, according to the criterion C.

2.2. Criteria for Variable Selection

Next we will briefly describe some performance functions and selection criteria that are employed in practice; for more details see (Kutner et al., 2004; Weisberg, 2005). The most common choices are statistical tests, information criteria and cross-validation. We will focus on statistical tests, as we use them in the remainder of the paper. We will also describe information criteria and contrast them to statistical tests, but will not further consider cross-validation, mainly because of its high computational cost.

2.2.1. Statistical Tests

Since the models tested at each iteration are nested, one can employ a likelihood-ratio (LR) test (or asymptotically equivalent approximations thereof such as score tests and Wald

tests) for nested models as a selection criterion. We next describe the likelihood-ratio test in more depth. For the LR test, the performance Perf is related to the log-likelihood (LL) and the criterion C tests the hypothesis that both models are equivalent with respect to some pre-specified significance level α. Let $\text{Dev}(T|\mathbf{X}) \equiv -2 \cdot \text{LL}(T|\mathbf{X})$ and $\text{Par}(T|\mathbf{X})$ be the deviance and number of parameters respectively of a model for target T using variables $\mathbf{X}$. Then, the test statistic of a nested test for models with variables $\mathbf{X}$ (null model) and $\mathbf{X} \cup \mathbf{Y}$ (alternative model) is computed as the difference in deviance of both models, that is, $\text{Dev}(T|\mathbf{X}) - \text{Dev}(T|\mathbf{X} \cup \mathbf{Y})$, and follows asymptotically a χ^2 distribution with $\text{Par}(T|\mathbf{X} \cup \mathbf{Y}) - \text{Par}(T|\mathbf{X})$ degrees of freedom (Wilks, 1938) [3].

Tests for nested models are essentially **conditional independence tests**, relative to some statistical model (for example, using linear regression without interaction terms tests for linear dependence), and assuming that the model is correctly specified. If the null model contains the predictors $\mathbf{X}$ and the alternative model contains $\mathbf{X} \cup \mathbf{Y}$, the nested test tests the hypothesis that the coefficients of $\mathbf{Y}$ are zero, or equivalently, that $\mathbf{Y}$ is conditionally independent of the target T given $\mathbf{X}$. We denote **conditional independence** of two non-empty sets $\mathbf{X}$ and $\mathbf{Y}$ given a (possibly empty) set $\mathbf{Z}$ as $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$. Finally, we note that one is not limited to likelihood-ratio based conditional independence tests, but can use any appropriate conditional independence test, such as a kernel-based test (Zhang et al., 2011).

A problem when using statistical tests for feature selection is that, due to multiple testing, the test statistics do not have the claimed distribution (Hastie et al., 2009) and the resulting p-values are too small (Harrell, 2001; Flom and Cassell, 2007), leading to a high false discovery rate. Approaches to deal with problem include methods that dynamically adjusting significance levels (Hwang and Hu, 2015), or methods that directly deal with the problem of sequential testing of stepwise procedures (G'Sell et al., 2016; Tibshirani et al., 2016). In order to perform tests on the model returned by stepwise selection, one can use resampling-based procedures to correct the p-values (Finos et al., 2010). In addition to the above problems, we note that the model returned by stepwise selection is sub-optimal, as it will have inflated coefficients (Flom and Cassell, 2007), reducing its predictive ability. If the main focus is to obtain a predictive model, methods performing regularization (like L1, L2 or elastic net) are more appropriate. In any case, procedures like cross-validation should be used to estimate out-of-sample predictive performance of the final model. We will not consider the above in this paper; we note however that our proposed algorithm is orthogonal to those methods and could be used in conjunction with them.

2.2.2. Information Criteria

Another way to compare two (or more) competing models is to use information criteria, such as the Akaike information criterion (AIC) (Akaike, 1973) or the Bayesian information criterion (BIC) (Schwarz et al., 1978). Information criteria are based on the fit of a model but additionally penalize the model by its complexity (that is, the number of parameters).

3. This result assumes that the larger hypothesis is correctly specified. In case of model misspecification, the statistic follows a different distribution (Foutz and Srivastava, 1977). Methods to handle model misspecification have been proposed by White (1982) and Vuong (1989). A method for dealing with model misspecification in model selection with information criterion is presented in (Lv and Liu, 2014). As this problem is out of this paper's scope, we did not further consider it.

The AIC and BIC scores of a model for T based on $\mathbf{X}$ are defined as follows:

$$\text{AIC}(T|\mathbf{X}) \equiv \textsc{Dev}(T|\mathbf{X}) + 2 \cdot \textsc{Par}(T|\mathbf{X})$$
$$\text{BIC}(T|\mathbf{X}) \equiv \textsc{Dev}(T|\mathbf{X}) + \log(n) \cdot \textsc{Par}(T|\mathbf{X})$$

where n is the number of samples. In the framework described above, information criteria can be applied by using the information criterion value as the performance function Perf, and a selection criterion C that simply compares the performance of two models, giving preference to the one with the lowest value. Alternatively, one could check that the difference in scores is larger than some threshold.

Neither AIC nor BIC are designed for cases where the number of predictors p is larger than the number of samples n (Chen and Chen, 2008), and thus also suffer from a high false discovery rate, similar to statistical tests. There have been several extensions to handle this problem, like the extended Bayesian information criterion (EBIC) (Chen and Chen, 2008), the generalized information criterion (GIC) (Fan and Tang, 2013; Kim et al., 2012), and the corrected risk information criterion (RIC_c) (Zhang and Shen, 2010), to name a few.

Compared to statistical tests, information criteria are somewhat limited as they can only be computed for models where the model complexity is known (like generalized linear models). An example where information criteria are not applicable are kernel-based tests (Zhang et al., 2011). Thus, statistical tests are inherently more general than information criteria. We will show next how, in case of nested models, using BIC directly corresponds to a likelihood-ratio test for some significance level α; the same reasoning can be applied to AIC and all information criteria that are computed based on the model likelihood and a penalty term. Let $\mathbf{X}$ and $\mathbf{X} \cup \mathbf{Y}$ be two candidate variables sets. $\mathbf{X} \cup \mathbf{Y}$ is selected (that is, the null hypothesis is rejected) if $\text{BIC}(T|\mathbf{X} \cup \mathbf{Y}) < \text{BIC}(T|\mathbf{X})$, or equivalently if $\textsc{Dev}(T|\mathbf{X}) - \textsc{Dev}(T|\mathbf{X} \cup \mathbf{Y}) > \log(n) \cdot (\textsc{Par}(T|\mathbf{X} \cup \mathbf{Y}) - \textsc{Par}(T|\mathbf{X}))$. Note that the left-hand side term equals the statistic of a likelihood-ratio test, whereas the right-hand size corresponds to the critical value. The statistic follows a χ^2 distribution with $k = \textsc{Par}(T|\mathbf{X} \cup \mathbf{Y}) - \textsc{Par}(T|\mathbf{X})$ degrees of freedom, and thus, the significance level equals $\alpha = 1 - F(\log(n) \cdot k; k)$, where $F(v; k)$ is the χ^2 cdf with k degrees of freedom at value v.

2.3. Markov Blankets in Bayesian Networks and Maximal Ancestral Graphs

The proposed algorithm is inspired by the theory of Markov blankets in Bayesian networks and maximal ancestral graphs. Next, we will provide a brief introduction of them; more details can be found in Appendix A. For a comprehensive introduction to Bayesian networks and maximal ancestral graphs we refer the reader to (Pearl, 1988; Spirtes et al., 2000; Richardson and Spirtes, 2002).

A **directed acyclic graph** (DAG) is a graph that only contains directed edges ($\rightarrow$) and has no directed cycles. A **directed mixed graph** is a graph that, in addition to directed edges also contains bi-directed edges ($\leftrightarrow$). A triplet of vertices $\langle X, Y, Z \rangle$ is called a **collider** if there are directed or bi-directed edges from X and Z to Y. A path p is called a **collider path** if every non-endpoint vertex is a collider on p.

Bayesian networks (BNs) consist of a DAG $\mathcal{G}$ and a probability distribution $\mathcal{P}$ over a set of variables $\mathbf{V}$. A DAG $\mathcal{G}$ is Markov and faithful to $\mathcal{P}$ if (a) each variable is conditionally independent of its non-descendants given its parents, and (b) all and only those conditional

independencies in $\mathcal{P}$ are entailed by $\mathcal{G}$. BNs are not closed under marginalization, that is, they are not able to encode latent confounders. BNs have been extended to represent such marginal distributions, and are called **directed maximal ancestral graphs** (DMAGs) (Richardson and Spirtes, 2002). The graphical structure of a DMAG is a directed mixed graph with the following restrictions: (i) it contains no directed cycles, (ii) it contains no almost directed cycles, that is, if $X \leftrightarrow Y$ then neither X nor Y is an ancestor of the other, and (iii) there is no path p such that each non-endpoint on p is a collider and every collider is an ancestor of an endpoint vertex of p.

A **Markov blanket** of a variable T is a **minimal** set of variables $\mathbf{MB}(T)$ that renders T conditionally independent of all remaining variables $\mathbf{V} \setminus \mathbf{MB}(T)$. In case the distribution can be faithfully represented by a BN or DMAG, then the Markov blanket of T is **unique**, and is defined as follows (see Appendix A for a proof sketch).

Definition 1 (Markov blanket) *The Markov blanket of T in a BN or DMAG consists of all vertices adjacent to T, as well as all vertices that are reachable from T through a collider path.*

In case of Bayesian networks, this simplifies to the set of parents, children, and parents of children of T.

3. Speeding-up Forward-Backward Selection

The standard FBS algorithm has two main issues. The first is that it is slow: at each forward iteration, all remaining variables are reconsidered to find the best next candidate. If k is the total number of selected variables and p is the number of input variables, the number of model evaluations (or in our case, independence tests) FBS performs is of the order of $O(kp)$. Although relatively low-dimensional datasets are manageable, it can be very slow for modern datasets which often contain thousands of variables. The second problem is that it suffers from multiple testing issues, resulting in overfitting and a high false discovery rate. This happens because it reconsiders all remaining variables at each iteration; variables will often happen to seem important simply by chance, if they are given enough opportunities to be selected. As a result, it will often select a significant number of false positive variables (Flom and Cassell, 2007). This behavior is further magnified in high-dimensional settings and with larger significance levels α. Next, we describe a simple modification of FBS, improving its running time while reducing the problem of multiple testing.

3.1. The Early Dropping Heuristic

We propose the following modification: after each forward iteration, remove all variables that do not satisfy the criterion C for the current set of selected variables $\mathbf{S}$ from the remaining variables $\mathbf{R}$. In our case, those variables are the ones that are conditionally independent of T given $\mathbf{S}$. The idea is to quickly reduce the number of candidate variables $\mathbf{R}$, while keeping many (possibly) relevant variables in it. The forward phase terminates if no more variables can be selected, either because there is no informative variable or because $\mathbf{R}$ is empty; to distinguish between forward and backward phases, we will call a forward phase

Algorithm 2 Forward-Backward Selection with Early Dropping (FBED^K)

Input: Dataset $\mathcal{D}$, Target T, Maximum Number of Runs K
Output: Selected Variables $\mathbf{S}$

$\mathbf{S} \leftarrow \emptyset$ *//Set of selected variables*
$K_{cur} \leftarrow 0$ *//Initializing current number of runs to 0*

//Forward phase: iterate until (a) run limit reached, or (b) $\mathbf{S}$ *does not change*
while $K_{cur} \leq K \wedge$ $\mathbf{S}$ changes **do**
    $\mathbf{S} \leftarrow \textsc{OneRun}(\mathcal{D}, T, \mathbf{S})$
    $K_{cur} \leftarrow K_{cur} + 1$
end while

//Perform backward selection and return result
return $\textsc{BackwardSelection}(\mathcal{D}, \mathbf{T}, \mathbf{S})$

function $\textsc{OneRun}(\mathcal{D}, T, \mathbf{S})$
    $\mathbf{R} \leftarrow \mathbf{V} \setminus \mathbf{S}$ *//Set of remaining candidate variables*

    //Forward phase: iterate until $\mathbf{R}$ *is empty*
    while $|\mathbf{R}| > 0$ **do**
        //Identify best variable V_{best} *out of* $\mathbf{R}$*, according to* Perf
        $V_{best} \leftarrow \underset{V \in \mathbf{R}}{\text{argmax}}\ \textsc{Perf}(\mathbf{S} \cup V)$
        //Select V_{best} *if it increases performance according to criterion* C
        if $\textsc{Perf}(\mathbf{S} \cup V_{best}) \underset{\text{C}}{>} \textsc{Perf}(\mathbf{S})$ **then**
            $\mathbf{S} \leftarrow \mathbf{S} \cup V_{best}$
        end if
        //Drop all variables from $\mathbf{R}$ *not satisfying* C
        $\mathbf{R} \leftarrow \{V : V \in \mathbf{R} \wedge V \neq V_{best} \wedge \textsc{Perf}(\mathbf{S} \cup V) \underset{\text{C}}{>} \textsc{Perf}(\mathbf{S})\}$
    end while
    return S
end function

with early dropping a **run**. Extra runs can be performed to reconsider variables dropped previously. This is done by retaining the previously selected variables $\mathbf{S}$ and initializing the set of remaining variables to all variables which have not been selected yet, that is $\mathbf{R} = \mathbf{V} \setminus \mathbf{S}$. The backward phase employed afterwards is identical to the standard backward-selection algorithm (see Algorithm 1). Depending on the number of additional runs K, this defines a family of algorithms, which we call **Forward Backward Selection with Early Dropping** (FBED^K), shown in Algorithm 2. The function OneRun shown in the bottom of Algorithm 2, performs one run until no variables remain in $\mathbf{R}$. Three interesting members of this family are the FBED^0, FBED^1 and FBED$^\infty$ algorithms. FBED^0 performs the first run until termination, FBED^1 performs one additional run and FBED$^\infty$ performs runs until no more variables can be selected. We will focus on those three algorithms hereafter.

The heuristic is inspired by the theory of Bayesian networks and maximal ancestral graphs (Spirtes et al., 2000; Richardson and Spirtes, 2002). Similar heuristics have been applied by Markov blanket based algorithms such as the Max-Min Parents and Children (MMPC) algorithm (Tsamardinos et al., 2003a) and HITON (Aliferis et al., 2003) successfully in practice and in extensive comparative evaluations (Aliferis et al., 2010). These algorithms also remove variables from consideration, and specifically the ones that are conditionally independent given some **subset** of the selected variables. In contrast, FBED^K reconsiders variables dropped during previous runs, while existing methods do not. Thus, *FBED^K bridges two types of algorithms to combine their advantages*: those that condition on all currently selected variables (such as FBS, grow-shrink (Margaritis and Thrun, 2000) and incremental association Markov blanket (Tsamardinos et al., 2003b)), and those that condition on subsets of variables to drop some of them (like MMPC (Tsamardinos et al., 2003a) and HITON (Aliferis et al., 2003)). Doing so, FBED^K manages to have the theoretical properties of the former (as shown in the next subsection), while obtaining speed-ups similar to the latter.

3.2. Comparing the Theoretical Properties of FBED^K to FBS

Due to early dropping of variables, the distributions under which FBED^K and FBS perform optimally are not the same. For all versions of FBED^K, with the exception of FBED$^\infty$, it is relatively straightforward to construct examples where FBS is able to identify variables that can not be identified by FBED^K. We give an example for FBED^0. FBED^0 may remove variables that seem uninformative at first, but become relevant if considered in conjunction with other variables. For example, let $X = T + Y$, with T and Y being independent Gaussian random variables, and assume that T is the outcome for which variable selection is performed. When no variables have been selected (first iteration), X will be dependent with T, while Y will be independent of T as it does not give any information about T by itself, and thus will be dropped. However, after selecting X, Y becomes conditionally dependent again (as $T = X - Y$), but FBED^0 will not select it as it was dropped in the first iteration. Surprisingly, in practice this does not seem to significantly affect the quality of FBED^0. In contrast, FBED^0 often gives better results, while also selecting fewer variables than FBS (see Section 4.4).

As mentioned above, it is not clear how FBS and FBED$^\infty$ are related in the general case. What can be shown is that both identify a minimal set of variables, although the identified solutions may not necessarily be the same.

Definition 2 (Minimal Variable Set) *Let* $\mathbf{V}$ *be the set of all variables and* $\mathbf{S}$ *a set of selected variables. We call a set of variables* $\mathbf{S}$ *minimal with respect to some outcome* T*, if:*

1. *No variable can be removed from* $\mathbf{S}$ *given the rest of the selected variables, that is,* $\forall V_i \in \mathbf{S}, (T \not\perp V_i \mid \mathbf{S} \setminus V_i)$ *holds.*

2. *Let* $\mathbf{R} = \mathbf{V} \setminus \mathbf{S}$*. No variable from* $\mathbf{R}$ *can be included in* $\mathbf{S}$*, that is,* $\forall V_i \in \mathbf{R}, (T \perp V_i \mid \mathbf{S})$ *holds.*

Corollary 3 *Any set of variables* $\mathbf{S}$ *selected by FBS is minimal.*

Proof See Appendix B. ■

Corollary 4 *Any set of variables* $\mathbf{S}$ *selected by FBED*$^{\infty}$ *is minimal.*

Proof See Appendix B. ■

In words, a minimal set is a set such that no single variable can be included to or removed from using forward and backward iterations respectively, that is, it is a local optimum for stepwise algorithms. Note that, although no single variable is informative for T if looked at separately, there may be sets of variables that are informative if considered jointly. A simple example is if all variables are binary and $T = X \oplus Y$, where $\oplus$ is the logical XOR operator. In this case $\mathbf{S} = \emptyset$ is minimal, as neither X nor Y are dependent with T, even though the set $\{X, Y\}$ fully determines T. Thus, none of the algorithms gives a globally optimal solution in all distributions.

We next consider the special case in which distributions can be represented by Bayesian networks or maximal ancestral graphs. We show that FBED^1 and FBED$^{\infty}$ identify the Markov blanket of a BN and DMAG respectively, assuming (a) that the distribution can be faithfully represented by the respective graph, and (b) that the algorithms have access to an **independence oracle**[4], which correctly determines whether a given conditional (in)dependence holds. This also holds for FBS but will not be shown here; proofs for similar algorithms exist (Margaritis and Thrun, 2000; Tsamardinos et al., 2003b) and can be easily adapted to FBS. For FBED^0 it can be shown that it selects a superset of the variables that are adjacent to T in the graph; this can be shown using the fact that, under the Markov and faithfulness assumptions, adjacent variables are dependent with T given any subset of the remaining variables.

Theorem 5 *If the distribution can be faithfully represented by a Bayesian network, then FBED*^{1} *identifies the Markov blanket of the target* T*.*

Proof See Appendix B. ■

Theorem 6 *If the distribution can be faithfully represented by a directed maximal ancestral graph, then FBED*$^{\infty}$ *identifies the Markov blanket of the target* T*.*

4. Assuming access to an independence oracle allows one to analyze whether the strategy used by FBED^K for identifying a Markov blanket is correct; thus, in practice, any errors in the output are due to statistical errors of the tests and not due to the heuristics or strategy used by FBED^K. Furthermore, it allows one to analyze the asymptotic behavior of algorithms without parametric distributional assumptions (for example, multivariate normality), but structural assumptions (for example, faithfulness). Assuming a conditional independence oracle is a standard assumption for the theoretical analysis of Markov blanket and causal discovery algorithms (for example, see (Spirtes et al., 2000; Aliferis et al., 2010)). In practice, FBED^K will not have access to an oracle, but will perform conditional independence tests to decide (in)dependence. There exist tests that, in the sample limit, will correctly identify (in)dependence. Examples include the partial correlation test for multivariate Gaussian data, and the G-test (Agresti, 2002) for multinomial data.

Proof See Appendix B. ■

3.3. Limitations and Practical Considerations

We have shown that FBED^K is able to solve the feature selection problem (that is, identify the Markov blanket of T) for distributions that are faithful to causal graphs. In practice, FBED^K may fail to identify the Markov blanket for several reasons. Naturally, in case the distribution can't be faithfully modeled with causal graphs, there is no guarantee of how close the solution will be to the optimal solution. However, previous comparisons show that forward selection performs as well as best subset selection, and is competitive with lasso (Hastie et al., 2017), indicating that its solutions are reasonably good approximations to the best subset solution, which we also confirm in the experimental section. Another, more subtle issue is if the conditional independence tests used are not appropriate to capture the dependencies present in the distribution. For instance, if all relations are non-linear and linear tests are used, there is no guarantee that any of the important variables will be selected. However, this is an issue with all feature selection algorithms (and predictive algorithms in general) and is not specific to FBED^K. Finally, if sample size is too low, or if the significance level is not set appropriately, dependencies may be incorrectly labeled as independencies and vice versa. Again, this is a general problem with all algorithms and can be handled by increasing sample size (if possible) and by appropriately setting or tuning the significance level. For example, for the task of learning Bayesian networks from Gaussian data using the PC algorithm (Spirtes et al., 2000), Kalisch and Bühlmann (2007) have shown that (under mild conditions) the significance level can be set in a way to ensure consistency asymptotically (Kalisch and Bühlmann, 2007, Theorem 1). The problem of learning Bayesian networks and Markov blanket discovery are closely related, and such results can possibly be translated and used by algorithms such as FBED^K, but it is out of the scope of the current paper.

We proceed with additional considerations regarding the sample size required to use FBED^K. FBED^K identifies the next variable to select conditional on all currently selected variables. Because of this, it can in principle take complex multivariate dependencies into consideration when selecting a variable. The complexity depends on the conditional independence test used (for example, non-linear tests can model more complex relations than linear tests). However, there is a clear trade-off between the complexity of dependencies that can be identified, and the sample size required to do so. For instance, if all variables are binary, the G-test of conditional independence (Agresti, 2002) can be used, which can identify any type of interaction between variables. In this case, the number of parameters increases exponentially with the number of selected variables: for k selected variables, the number of parameters is in the order of $O(2^k)$, and consequently, the number of samples required to have sufficient power also increases exponentially. Using linear models (for example, linear, logistic or Cox regression for continuous, categorical or time-to-event outcomes respectively), simpler, linear dependencies can be identified, and the number of samples required increases only linearly with the number of parameters. Rules of thumb for setting the minimum sample size for linear models are given in (Peduzzi et al., 1996; Harrell, 2001; Vittinghoff and McCulloch, 2007). For binary logistic regression, one recommendation is to

use at least $s = c/\min(p_0, p_1) \cdot k$ samples (Peduzzi et al., 1996), where p_0 and p_1 are the proportion of negative and positive classes of T respectively, k is the number of parameters in the model and c is a user-set parameter, which is usually recommended to be between 5 and 20, with larger values leading to more accurate results. Thus, multivariable methods like FBED^K should only be used when sufficient sample size is available; alternatively, one can use rules of thumb as stopping criteria (that is, to determine when to stop selecting variables).

Next, we make a few recommendations based on the above considerations; exact rules are hard to devise, as they depend on the specific problem at hand. In case sample size is very low (a few tens or hundreds of samples), sample-efficient methods like the max-min parents and children algorithm (Tsamardinos et al., 2003a) (which condition only on small subsets of variables), information-theoretic feature selection methods (Brown et al., 2012) (which only condition on up to 1 variable), or univariate feature selection methods are more preferable than methods like FBED^K. Otherwise, we recommend using linear multivariable methods like OMP (Pati et al., 1993; Davis et al., 1994), LASSO (Tibshirani, 1996) or FBED^K with linear tests, and if sample size allows to also consider FBED^K using non-linear tests. Finally, we believe it is also worth considering robust tests (Lagani et al., 2017) for FBED^K, as outliers often exist in practice and may negatively impact tests which do not take them into account.

4. Experimental Evaluation

In this section we evaluate FBED^K, and compare it to the standard FBS algorithm[5], feature selection with LASSO (called LASSO-FS hereafter) (Tibshirani, 1996), the Max-Min Parents and Children algorithm (MMPC) (Tsamardinos et al., 2003a), and no feature selection (NO-FS), which was used as the baseline method. We note that MMPC is designed specifically for low-sample size and high-dimensional settings, and thus may not perform optimally in the datasets considered here. The reason we compare against it is because, it belongs in the same category of algorithms as FBED^K (that is, is also inspired by causal graphs).

We implemented all algorithms in Matlab except for LASSO, for which we used the glmnet implementation (Qian et al., 2013). We used 12 binary classification datasets, with sample sizes ranging from 200 to 16772 and number of variables between 166 and 100000. The datasets were selected from various competitions (Guyon et al., 2004, 2006a) and the UCI repository (Dietterich et al., 1994), and were selected to cover a wide range of variable and sample sizes. A summary of the datasets is shown in Table 1. All experiments were performed in Matlab, running on a desktop computer with an Intel i7-7700K processor and 32GB of RAM.

5. We want to point out that, although the early dropping heuristic only affects the forward phase of the algorithm, we chose evaluate FBED^K and FBS with the backward phase. This is done mainly as FBED^K and FBS require the backward phase to have provable theoretical guarantees, and because that is how the algorithms are presented throughout the paper. To ensure that this will not significantly affect the results, we performed a few preliminary anecdotal experiments, and observed that (a) the number of features removed by backward selection is typically small, and (b) the predictive performance is very similar when backward selection is used.

Table 1: Binary classification datasets used in the experimental evaluation. n is the number of samples, p is the number of predictors and $P(T = 1)$ is the proportion of instances where $T = 1$.

Dataset	n	p	P(T = 1)	Type	Domain	Source
musk (v2)	6598	166	0.15	Real	Musk Activity Prediction	UCI ML Repository (Dietterich et al., 1994)
sylva	14394	216	0.94	Mixed	Forest Cover Types	WCCI 2006 Challenge (Guyon et al., 2006a)
madelon	2600	500	0.5	Integer	Artificial	NIPS 2003 Challenge (Guyon et al., 2004)
secom	1567	590	0.93	Real	Semi-Conductor Manufacturing	UCI ML Repository M. McCann, A. Johnston
gina	3568	970	0.51	Real	Handwritten Digit Recognition	WCCI 2006 Challenge (Guyon et al., 2006a)
hiva	4229	1617	0.96	Binary	Drug discovery	WCCI 2006 Challenge (Guyon et al., 2006a)
gisette	7000	5000	0.5	Integer	Handwritten Digit Recognition	NIPS 2003 Challenge (Guyon et al., 2004)
p53 Mutants	16772	5408	0.01	Real	Protein Transcriptional Activity	UCI ML Repository (Danziger et al., 2006)
arcene	200	10000	0.56	Binary	Mass Spectrometry	NIPS 2003 Challenge (Guyon et al., 2004)
nova	1929	16969	0.72	Binary	Text classification	WCCI 2006 Challenge (Guyon et al., 2006a)
dexter	600	20000	0.5	Integer	Text classification	NIPS 2003 Challenge (Guyon et al., 2004)
dorothea	1150	100000	0.9	Binary	Drug discovery	NIPS 2003 Challenge (Guyon et al., 2004)

The remainder of this section is organized as follows. First, we describe in detail the experimental setup, that is, all algorithms used, their hyper-parameters, and how we performed model selection and performance estimation. We proceed by evaluating how the running time, number of selected variables and predictive performance of FBED^K is affected by the number of runs K. Afterwards, we compare FBED^K to FBS, to show the effects of the early dropping heuristic. Next, we evaluate FBED^K in a realistic scenario with other feature selection methods, where hyper-parameters of the feature selection and classification algorithms are optimized. Then, we compare FBED^K and LASSO in terms of predictive ability when the number of variables to select is fixed so that all algorithms produce solutions of equal size. This is done for two reasons: (a) because LASSO tends to select many variables otherwise, giving it an advantage over FBED^K in terms of predictive performance, and (b) because this allows us to evaluate how well FBED^K orders the variables in comparison to LASSO. Finally, we compare FBED^0, FBED^1, FBED$^\infty$ and FBS on simulated data containing only irrelevant variables, investigating how is each algorithm is affected by multiple testing in terms of the falsely selected variables.

4.1. Experimental Setup

We present an overview of the experimental setup next. Additional details for each specific experiment are described in the respective section.

4.1.1. Feature selection algorithms

First of all we note that, although the early dropping heuristic only affects the forward phase of the algorithm, we will use FBED^K and FBS with their backward phase in the experiments. This is done mainly as FBED^K and FBS require the backward phase to have provable theoretical guarantees, and because that is how they are presented throughout the paper.

As selection criteria for FBED^K, FBS and MMPC we used a nested likelihood-ratio independence test based on logistic regression. For FBED^K and MMPC, the significance level α of the conditional independence test was set to $\{0.001, 0.005, 0.01, 0.05, 0.1\}$, covering a range of commonly used values, while for FBS we explored a total of 100 values, uniformly spaced in $[0.001, 0.01]$. For the K value of FBED^K we used $\{0, 1, \ldots, \infty\}$, while the maximum conditioning size $maxK$ of MMPC was set to $\{1, 2, 3, 4\}$. For LASSO-FS we set all parameters to their default values and set the maximum number of λ values, λ_{max}, to 100. Thus, we used 5 hyper-parameter combinations for each value K of FBED^K, 100 for FBS and LASSO-FS, and 20 for MMPC.

Unfortunately, for MMPC there were 2 datasets where not all hyper-parameter combinations were executed, as they were taking too long to terminate (see results about running time in Appendix D); we stopped execution if a time limit of 2 days was exceeded. Specifically, for the gisette and nova datasets MMPC was only executed with $maxK \leq 2$

We would like to point out that for a given value K for FBED^K, all solutions with fewer runs (smaller K) can be computed with minimal computational overhead, as the forward phases have already been computed and only the backward phases need to be performed separately. As the number of variables selected is relatively small, the computational cost of the backward phases is usually negligible. Thus, FBED^K required a single execution with $K = \infty$ for a given α. Unfortunately, something similar can not be done with MMPC, and thus it has to be executed for each hyper-parameter value separately [6].

4.1.2. Predictive models

We used both, linear and non-linear predictive models. As linear models we used elastic net regularized logistic regression (Zou and Hastie, 2005), using $\lambda_{max} = 100$ and the mixture parameter α set to $\{0, 0.25, 0.5, 0.75, 1\}$ ($\alpha = 0$ corresponds to L2 regularization and $\alpha = 1$ to L1 regularization), leading to a total of 500 hyper-parameter combinations. We remind the reader that regularization is important, especially after feature selection has been performed, in order to improve predictive performance due to inflated coefficients (Flom and Cassell, 2007) (see also Section 2.2). As non-linear models we used Gaussian support vector machines (SVM) (Cortes and Vapnik, 1995) and random forests (RF) (Breiman, 2001). For SVMs we used the LIBSVM (Chang and Lin, 2011) implementation, while for RFs we used the

6. This is only partially true, as for FBED^K, FBS and MMPC we implemented a caching mechanism to avoid fitting the same logistic regression model more than once. This mechanism was not used however for experiments measuring the running time of the algorithms.

TreeBagger implementation in Matlab. The cost hyper-parameter C of SVMs was set to $\{2^{-10}, 2^{-9}, \ldots, 2^{9}\}$ (a total of 20 values), while the remaining hyper-parameters were set to their default values. For RFs the number of trees was set to 500, the minimum leaf node size was set to $\{1, 5, 9\}$ and the number of variables to split at each node was set to $\{0.5, 1, 1.5\}$ $\cdot\sqrt{p}$ (9 combinations in total).

4.1.3. Linear vs non-linear models

Throughout the section, we will report results obtained by using only linear models or a combination of linear and non-linear models. The former is done to evaluate the ability of the feature selection methods of identifying features that are linearly (or possibly monotonically) related to the outcome. The reason for that is that all evaluated methods can only identify such types of dependencies; we note that all algorithms (except for LASSO) can be trivially adapted to also handle non-linear dependencies by using an appropriate conditional independence test. Non-linear models were also considered to better simulate a realistic scenario, as such methods would be used in a typical analysis. Furthermore, it is interesting to see whether there are any significant differences between linear and non-linear modeling for any of the considered feature selection algorithms.

4.1.4. Model selection and performance estimation protocols

Ideally, we would like to evaluate each feature selection algorithm using an optimal predictive model, in order to measure how informative the selected features are. As an optimal model is not available in practice, we approximate this by using a variety of predictive algorithms as well as multiple hyper-parameter value combinations for each (see above), and perform hyper-parameter optimization (also called tuning or model selection) to find a good approximate model; interested readers may refer to (Feurer and Hutter, 2018; Tsamardinos et al., 2018b) for more details. We proceed with a description of the model selection and performance estimation protocols we used.

As the performance metric we optimize and report the area under the ROC curve (AUC). For model selection and performance estimation we used a 60/20/20 stratified split of the data, using 60% as a training set, 20% as a validation set and the remaining 20% as a test set. A hyper-parameter configuration is defined as a combination of a feature selection algorithm and its hyper-parameters, as well as a modeling algorithm and its hyper-parameters. Given a set of configurations, the best one is chosen by training models for all of them on the training set and selecting the configuration of the model with the highest performance on the validation set. Finally, the predictive performance of that configuration is obtained by training a final model on the pooled training and validation sets, and evaluating it on the test set. To account for the variation due to the data splitting, we repeated this procedure multiple times for different splits and report averages over repetitions. For datasets with more than 1000 samples the number of repetitions was set to 10, and to 50 for the rest.

4.2. Effect of the Number of Runs K

We performed an experiment to measure the effect of K on the running time, number of selected variables and predictive performance of FBED^K. For the running time and number of selected variables we executed FBED^K once for each hyper-parameter value on

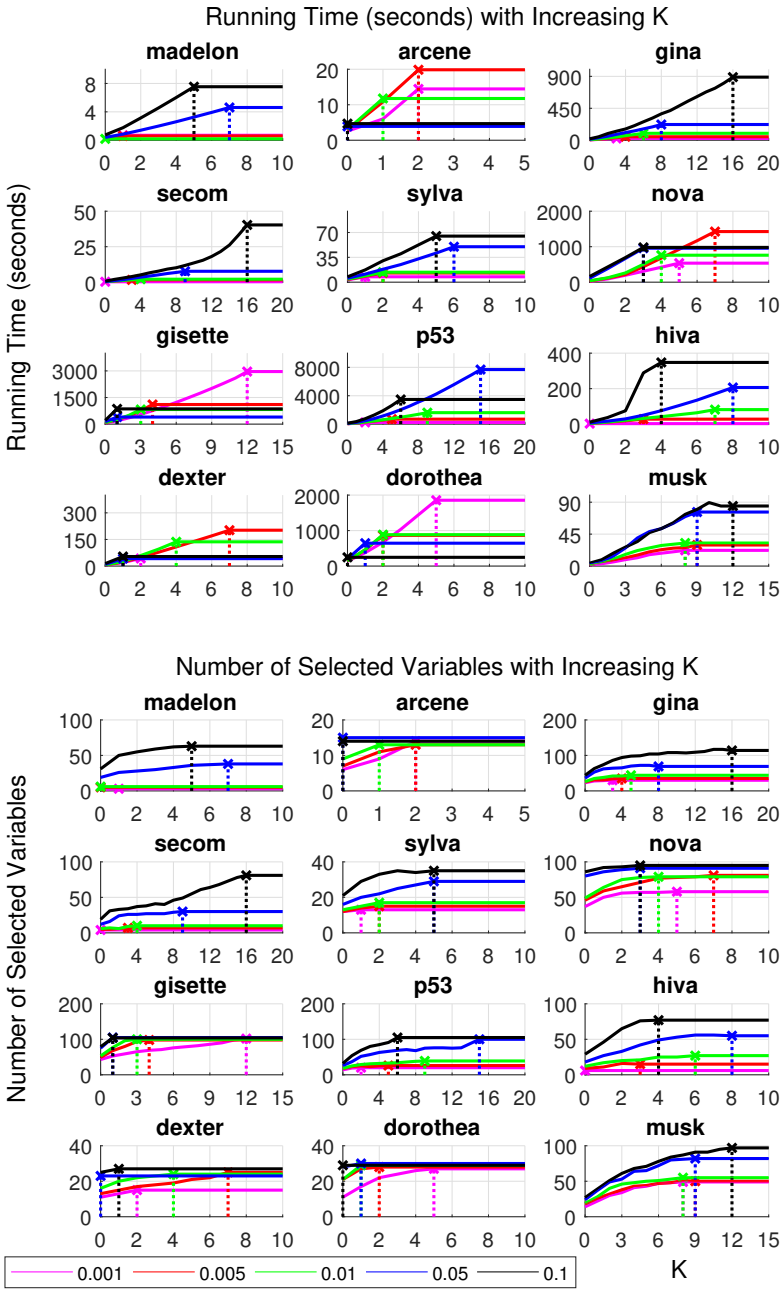


Figure 1: The figure shows how the running time (top) and the number of selected variables (bottom) vary with an increasing number of runs K for different values of the threshold parameter α. The vertical lines indicate the value of K for which FBED^K has converged. We can see that running time increases almost linearly with increasing K. Also, most progress is made in the first few runs, and any additional runs increase running time while only selecting a few more variables.

the complete datasets, while for the predictive performance we used the model selection and performance estimation protocols described previously and report averages over multiple repetitions of the experiment.

Figure 1 shows how the number of runs K affects the running time and the number of selected variables. Vertical lines show the value of K for which the algorithm has converged (that is, after that point more runs do not select any more variables). We can see that running time increases almost linearly with an increasing number of runs K, meaning that any additional run has a roughly linear computational cost with respect to the size of the dataset. Furthermore, convergence is typically achieved in less than 10 runs, although for a few cases up to 16 runs are required. As expected, the number of selected variables increases with K, as well as with the threshold α. In the majority of cases however, most progress is made in the first few runs, and further runs increase the number of selected variables only marginally. *Based on those results, we recommend considering relatively small values of K, up to $K < 10$.*

Figure 2 shows how the area under the ROC curve (AUC) varies with an increasing number of runs K, for 5 different values of the threshold parameter α of FBED^K. We observe that, although AUC often tends to increase with K, this is not always the case. For instance, for the nova and dexter datasets, AUC actually decreases with K for some values of α. The maximum AUC is typically achieved with relatively small values of K, further suggesting that considering higher values for K is not necessary. Also, there are no clear relationships between AUC, the value of α and the type of predictive models used. Depending on the dataset different values of α or predictive models may be optimal. *Thus, in practice we recommend considering several combinations of α and K.* Optimizing over both α and K will be considered in Section 4.4.

4.3. FBED^K vs FBS

In this section we compare FBED^K to the standard FBS algorithm in terms of predictive performance, number of selected variables and running time. The algorithms were compared on the same hyper-parameters (for example, FBS vs FBED^0 with $\alpha = 0.01$); results when also optimizing over hyper-parameters are shown in Section 4.4. Model selection and performance estimation was performed for each feature selection algorithm and each hyper-parameter value separately, following the procedure described in the experimental setup. *The main goal of this comparison is to show that FBED^K and FBS perform similarly for the same hyper-parameters, with the former being faster.* A summary of the results is presented next.

Figure 3 shows how the algorithms compare in terms of predictive performance, number of selected variables and running time. Each column shows the distribution of the respective metric across all thresholds and datasets, as well as the mean and median values. The difference in AUC is computed is AUC(FBED^K) - AUC(FBS), the relative number of selected variables is computed as the ratio of variables selected by FBED^K compared to FBS, and the speed-up is computed as Time(FBS) / Time(FBED^K). The y-axis corresponds to different values of K used by FBED^K. Only the first few values ($K \leq 9$), as well as the last one ($K = \infty$) are shown, as the left-out ones were almost identical to $K = \infty$.

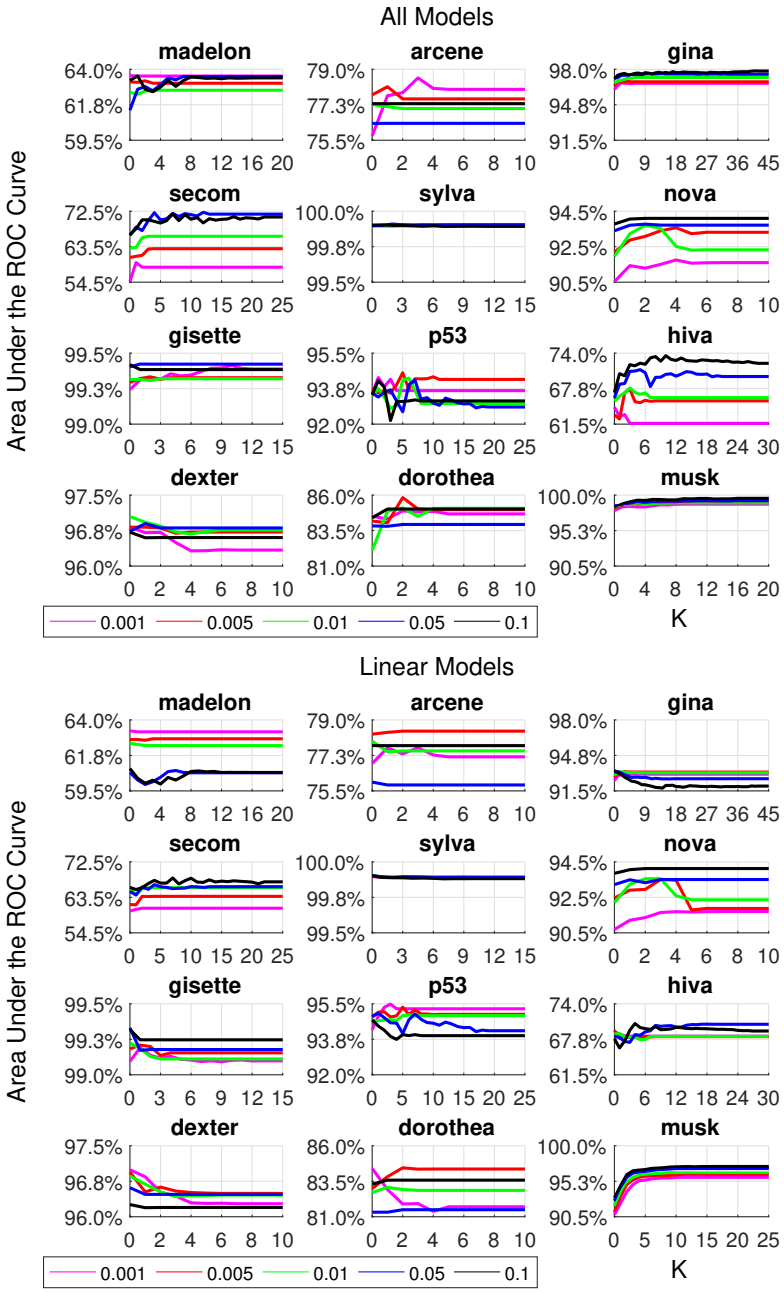


Figure 2: The figure shows how the AUC varies with an increasing number of runs K for different values of the threshold parameter α, using non-linear and linear models (top) or linear models only (bottom). There is no clear pattern for which thresholds or values of K to prefer, but the optimal values depend on the specific dataset, as well as on the predictive models used. However, in most cases only a few runs are required to achieve maximal AUC.

Regarding predictive performance, FBED^K performs as good as FBS on average, irrespective of the type of predictive models used. For FBED^K with $K < 3$, the average

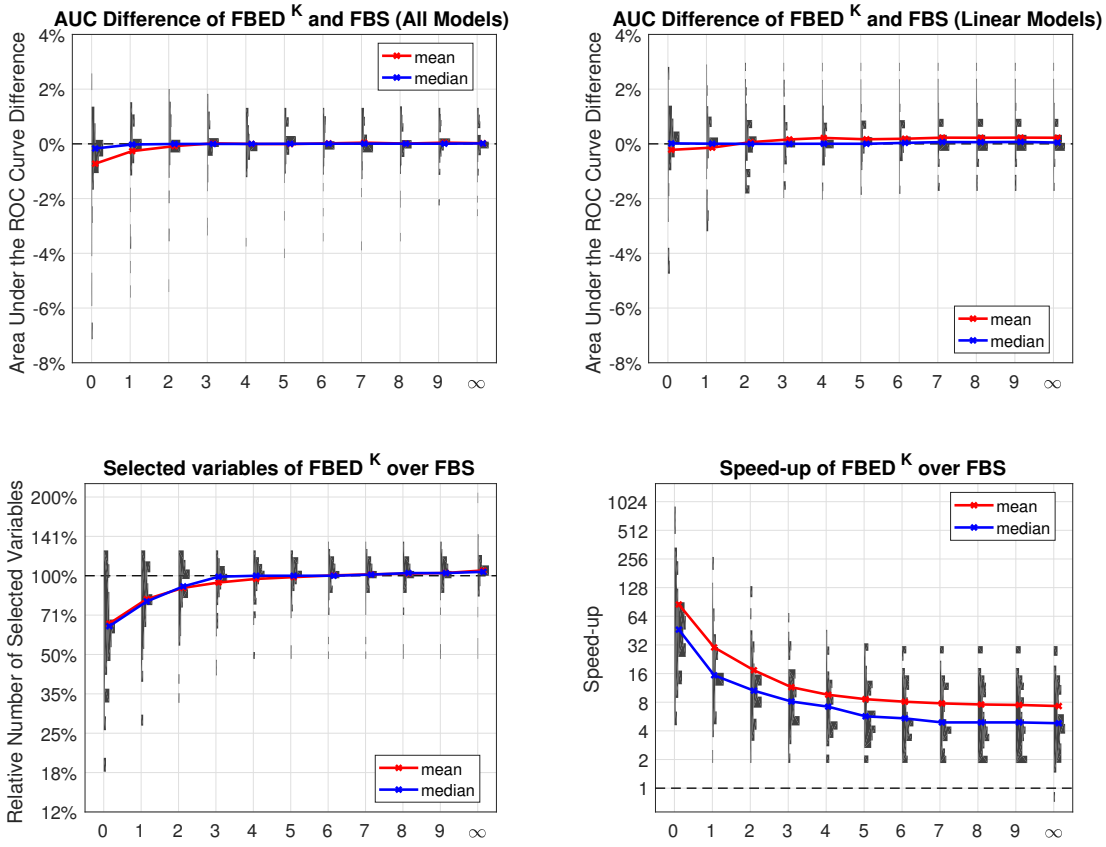


Figure 3: The x-axis of the figures on the top row shows the difference in AUC between FBED^K and FBS, with positive values indicating that FBED^K performs better than FBS. The AUC of the top left figure is computed by optimizing over all models, while for the one of the top right figure only linear models were considered. The relative number of selected variables (bottom left) shows the number of variables selected by FBED^K compared to the ones selected by FBS. The speed-up (bottom right) is computed as the one obtained by FBED^K over FBS. For all cases, the distribution over all thresholds and datasets is shown, as well as the mean and median values. The y-axis on all figures is the value of K used by FBED^K. Overall, FBED^K has a virtually identical performance with FBS, while being on average between 1 and 2 orders of magnitude faster.

difference in AUC is less than 1% while the median difference is close to 0, and for all other K the performance is almost identical to FBS. We note that all those lower-performing cases are also the ones where FBED^K selected much fewer variables than FBS. In terms of the number of selected variables, FBED^K produces smaller solutions for $K < 3$, and tends to select the same number as FBS with increasing K. Finally, in terms of running time, FBED^K is significantly faster than FBS, being about 1-2 orders of magnitude faster on average in all cases.

An interesting case is for FBED^3, where the number of selected variables and AUC between both algorithms is almost identical, while being around 10 times faster than FBS. If

speed and small solutions are important, FBED^0 and FBED^1 are good choices, as they are ∼30-100 times faster than FBS, selecting only ∼70%-80% of the variables with a minimal drop in AUC. Therefore, if the number of variables is high, FBED^0 and FBED^1 are preferable due to their low computational lost. Furthermore, in low sample settings the smaller solutions of FBED^0 and FBED^1 are important, as selecting many variables leads to loss of power and overfitting. If on the other hand the sample size is large and the number of variables is relatively small, both FBS and FBED^K with higher values of K are reasonable choices, with the latter being more attractive, as it is around 1 order of magnitude faster and thus can scale to higher variable sizes.

4.4. Comparison of FBED^K with other Feature Selection Methods

We performed an experiment where we also optimize over the hyper-parameter values of feature selection algorithms. *The main objective of this comparison is to compare FBED^K to other feature selection algorithms in a realistic scenario, where hyper-parameter values are optimized.* For this comparison we focus on the predictive performance and number of selected variables; additional results showing the running time of each algorithm can be found in Appendix D.

4.4.1. Setup

For FBED^K optimization is performed over the threshold α and the number of runs K. We examine four versions of FBED^K: FBED^0, FBED$^{\leq 1}$, FBED$^{\leq 3}$ and FBED$^{\leq \infty}$. FBED$^{\leq K}$ means that optimization was performed for all results up to K runs. Thus, the number of hyper-parameter configurations used were 5, 10, 20 and around 50 (for most cases) for FBED^0, FBED$^{\leq 1}$, FBED$^{\leq 3}$ and FBED$^{\leq \infty}$ respectively. The hyper-parameter values for FBS, MMPC and LASSO-FS are the ones described in Section 4.1.1 (a total of 100, 20 and 100 respectively). We also included results when no feature selection was performed (NO-FS). Finally, we remind the reader that we used two sets of classification algorithms and hyper-parameters, one containing only linear algorithms (elastic net regularized logistic regression) and one also containing non-linear ones in addition to the linear ones (Gaussian support vector machines and random forests). For brevity, we will refer to linear models as LM and to the combination of linear and non-linear models as NLM hereafter.

4.4.2. Results

A summary of the results averaged over repetitions, measuring the AUC and number of selected variables is shown in Tables 2 and 3. For each algorithm, we computed a score which is the average rank of that algorithm over all datasets. The final rank of an algorithm is then computed based on that score. We used a bootstrap-based procedure to compute the probability of an algorithm being significantly better or worse than all competitors, and used a threshold of 95%. The procedure is described in more detail in Appendix C. In the tables, algorithms that are statistically significantly better than all others are shown in bold, whereas algorithms that are worse than the rest are shown in italic.

Overall, performing no feature selection has the highest AUC. Out of all feature selection methods, LASSO-FS offers the best predictive performance, statistically significantly outperforming the rest in 4 datasets. MMPC outperforms the rest in 2 and 3 datasets using

Table 2: Area under the ROC curve and number of selected variables for all feature selection algorithms using linear and non-linear models. The results are obtained after optimizing the hyper-parameters of the feature selection and modeling algorithms. Bold and italic entries denote that the method is significantly better or worse than all other feature selection methods (excluding NO-FS) respectively. The score is the average rank of each method over all datasets and the final rank is computed using those scores. Methods that select more variables tend to also perform better (Spearman correlation between AUC and variable rankings is -0.976).

	Algorithm	musk	sylva	madelon	secom	gina	hiva	gisette	p53	arcene	nova	dexter	dorothea	Score	Rank
AUC (all models)	FBED^0	*98.5*	99.9	63.4	63.7	*97.0*	67.4	99.4	93.3	77.5	93.6	96.7	83.8	6.33	8
	FBED$^{\leq 1}$	98.7	99.9	63.3	63.0	97.3	70.9	99.4	93.5	76.9	94.0	96.8	83.5	6.17	7
	FBED$^{\leq 3}$	99.2	99.9	63.0	68.0	97.3	68.8	99.4	93.2	77.3	93.6	96.8	84.3	5.46	6
	FBED$^{\leq \infty}$	99.5	99.9	63.6	67.6	97.6	72.3	99.4	93.5	77.3	93.6	96.8	84.9	4.71	4
	FBS	99.5	99.9	64.8	69.5	97.5	69.1	99.4	94.1	77.2	92.4	*95.9*	84.3	5.00	5
	MMPC	98.9	99.9	65.1	63.4	98.1	68.3	99.6	93.1	79.6	**96.7**	97.3	**91.7**	4.00	3
	LASSO-FS	**99.9**	99.9	**82.3**	69.8	98.2	74.2	**99.7**	94.2	**82.4**	96.1	97.2	89.0	2.58	2
	NO FS	99.9	99.9	82.8	71.9	98.3	74.3	99.6	94.7	90.0	96.6	98.2	94.6	1.75	1
Selected Variables	FBED^0	**22.1**	13.4	8.4	15.0	**32.9**	**18.8**	72.9	24.2	8.7	66.5	17.4	21.6	1.33	1
	FBED$^{\leq 1}$	35.6	15.2	8.2	18.6	47.7	34.1	80.2	23.5	9.6	71.9	18.0	23.2	2.79	2
	FBED$^{\leq 3}$	47.1	21.1	14.0	24.4	56.5	43.5	80.2	26.4	9.3	72.1	18.5	22.3	3.63	4
	FBED$^{\leq \infty}$	77.6	25.5	14.9	41.1	105.8	75.1	79.4	33.7	9.3	72.3	18.7	22.4	4.79	5
	FBS	76.1	21.1	19.6	28.4	78.7	33.8	**65.0**	32.6	11.0	71.6	16.9	22.0	3.46	3
	MMPC	42.5	20.7	17.0	16.2	155.4	51.7	*388.2*	68.8	*43.9*	*879.7*	*187.2*	*445.1*	5.42	6
	LASSO-FS	*138.2*	*43.0*	*495.4*	*133.3*	*406.4*	*306.6*	187.5	*161.9*	29.6	349.8	97.9	70.1	6.58	7
	NO FS	166.0	213.0	500.0	468.0	970.0	1617.0	4948.0	5408.0	9955.0	11853.0	9988.0	88215.0	8.00	8

Table 3: Area under the ROC curve and number of selected variables for all feature selection algorithms using linear models. The results are obtained after optimizing the hyper-parameters of the feature selection and modeling algorithms. Bold and italic entries denote that the method is significantly better or worse than all other feature selection methods (excluding NO-FS) respectively. The score is the average rank of each method over all datasets and the final rank is computed using those scores. Methods that select more variables tend to also perform better (Spearman correlation between AUC and variable rankings is -0.595), but the effect is not as strong as the one of the previous results (Table 2).

	Algorithm	musk	sylva	madelon	secom	gina	hiva	gisette	p53	arcene	nova	dexter	dorothea	Score	Rank
AUC (linear models)	FBED^0	*93.0*	99.9	62.3	65.1	93.3	69.4	99.3	94.0	78.0	93.3	96.6	81.7	4.58	4
	FBED$^{\leq 1}$	94.6	99.9	62.2	62.6	93.3	68.3	99.3	94.4	78.0	93.7	96.6	83.5	4.83	5
	FBED$^{\leq 3}$	96.5	99.9	62.0	64.0	93.2	68.3	99.2	95.0	77.7	93.4	96.5	83.5	5.38	6
	FBED$^{\leq \infty}$	97.1	99.9	62.0	63.8	92.1	69.0	99.2	95.1	77.7	93.4	96.5	83.3	5.71	7
	FBS	97.1	99.9	62.2	67.0	92.5	67.3	99.2	94.6	75.5	92.5	*95.6*	83.3	5.92	8
	MMPC	93.8	**99.9**	62.3	64.3	93.4	68.3	99.3	95.3	75.6	**96.7**	96.6	**89.3**	3.50	3
	LASSO-FS	**97.5**	99.9	62.1	64.8	92.1	70.7	**99.6**	95.4	**80.6**	95.8	**97.3**	86.9	3.17	2
	NO FS	97.6	99.9	59.5	66.6	91.2	71.8	99.6	95.8	87.3	96.5	98.3	91.7	2.92	1
Selected Variables	FBED^0	**22.7**	**14.1**	8.0	15.6	**34.2**	**20.4**	71.5	23.2	**8.6**	**66.5**	17.5	21.0	1.21	1
	FBED$^{\leq 1}$	35.2	16.8	10.6	18.0	46.9	31.2	79.3	25.5	9.4	71.4	17.9	22.0	2.71	2
	FBED$^{\leq 3}$	51.4	23.3	11.6	26.4	51.9	40.3	83.9	32.8	9.7	71.6	18.3	21.2	3.88	4
	FBED$^{\leq \infty}$	86.7	25.3	11.6	47.0	114.1	70.2	84.0	33.8	9.7	71.6	18.3	21.3	4.71	5
	FBS	79.5	22.8	9.0	34.7	80.1	29.3	**65.0**	34.6	11.0	72.4	16.7	22.0	3.63	3
	MMPC	43.3	34.2	8.0	17.7	137.8	36.1	*389.0*	92.2	*43.0*	*879.7*	*193.4*	*363.2*	5.29	6
	LASSO-FS	*144.5*	*57.8*	47.2	93.5	*369.3*	*285.5*	206.5	*208.0*	25.7	307.8	87.0	74.6	6.58	7
	NO FS	166.0	213.0	500.0	468.0	970.0	1617.0	4948.0	5408.0	9955.0	11853.0	9988.0	88215.0	8.00	8

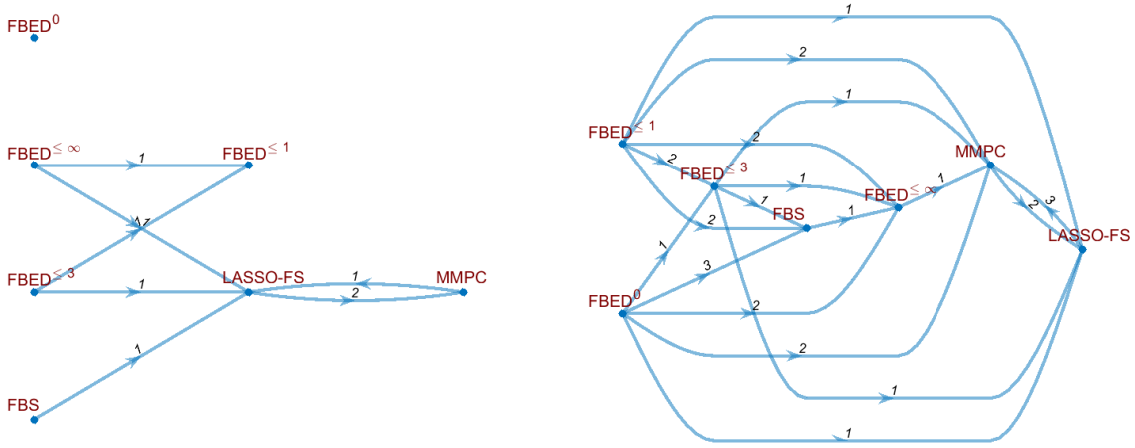
NLM and LM respectively. In two cases FBED^0 is significantly worse than the rest (musk and gina), while FBS is the worst in 1 dataset. However, in terms of the number of selected

variables, LASSO-FS selects statistically significantly more in 7 and 5 cases for NLM and LM, while MMPC selects the most variables in 5 datasets for both NLM and LM. FBED^K and FBS on the other hand tend to select fewer variables. Thus, *there is a clear trade-off between model interpretability (number of selected variables) and predictive performance (AUC)*. Specifically, there is a -0.976 and -0.595 Spearman correlation between the AUC and selected variables ranks for NLM and LM respectively. For that reason, we performed an additional experiment, comparing the AUC between FBED^K, FBS and LASSO-FS by constraining the total number of variables to select, presented in Section 4.5.

A strong outlier in the NLM case is the difference in performance of LASSO-FS compared to the other methods on the madelon dataset, where LASSO-FS reaches an AUC that is $17.3 - 19.5\%$ higher, which is also close to the AUC of NO-FS. The main reason why all methods fail is because the madelon dataset has been constructed in a way that makes it hard for linear methods (FBED^K, FBS and MMPC using the logistic regression test). Specifically, the outcome variable has been artificially constructed to be a XOR-type problem of 5 variables (Guyon et al., 2006b). Further evidence for the hardness of this problem is the fact that using LM and NO-FS achieves an AUC of only 59.5 (even lower than all feature selection methods). LASSO-FS, although also linear, is able to pick up the signal as it basically performs no feature selection, selecting 495.4 out of 500 variables on average. This happens because LASSO-FS explores up to 100 values for λ, some of which correspond to very dense solutions. In contrast, due to the experimental setup, none of the remaining methods selects that many variables in this case.

An interesting observation is the fact that FBED^0 and FBED$^{\leq 1}$, the forward selection methods selecting the fewest variables, are ranked higher in terms of AUC than FBED$^{\leq 3}$, FBED$^{\leq \infty}$ and FBS when using LM. Thus, they are better suited to pick out linear trends in the data, producing solutions that are smaller and more predictive compared to algorithms of the same type. Using NLM gives an even bigger advantage to methods selecting more variables, as this increases the chance to also capture some non-linear signals in the data.

Finally, we performed a statistical test between all pairs of methods to identify cases where a method outperforms others, both in terms of AUC and number of selected variables. Again, we used a bootstrap-based test, computing the joint probability that method A has a higher AUC and selecting fewer variables than method B, using the same procedure as described in Appendix C. A method is considered to dominate another, if that probability is higher than 95%. The results are summarized in Figure 4. Each node corresponds to a feature selection method, and a directed edge from method A to B with weight w denotes that A dominates B in w datasets. We observe that, except for a single case where FBED$^{\leq \infty}$ gets dominated by FBS in 1 dataset for LM, FBED^K is never dominated by any other method, neither for the NLM nor for the LM case. In both NLM and LM cases, FBED^K dominates the competitors in 1-3 cases, while LASSO-FS and MMPC only dominate each other in 1-3 cases. Especially interesting is the fact that for NLM, FBED$^{\leq 3}$, FBED$^{\leq \infty}$ and FBS (that is, the forward-selection algorithms typically selecting the most variables) are the only algorithms that dominate others, while also not getting dominated. On the other hand, using LM only FBED^0 and FBED$^{\leq 1}$ both dominate others and are not getting dominated. This agrees with the observation made before, that FBED^0 and FBED$^{\leq 1}$ are particularly well suited to identify compact and linearly predictive solutions.



(a) Dominating Relationships (All Models) (b) Dominating Relationships (Linear Models)

Figure 4: The figures show how often a feature selection method dominates another (that is, has a higher AUC while selecting fewer variables), using non-linear models (left) and linear models (right). An edge from method A to B with weight w indicates that A dominates B in w datasets. Except for FBED$^{\leq\infty}$ for linear models, which gets dominated by FBS in 1 dataset, methods in the FBED^K family are never dominated by FBS, MMPC or LASSO-FS, while typically dominating them in 1-3 datasets.

Overall, *there is no clear winner, and the choice depends solely on the goal.* If the goal is predictive performance, LASSO-FS or MMPC are clearly preferable. If on the other hand one is interested in interpretability, then methods from the FBED^K family with small values of K are preferable. Regarding FBS, there is no scenario where it is preferable over one of the other algorithms. We must note that those results are somewhat artificial, as the performance of FBED^K and FBS highly depends on the hyper-parameter values chosen for the experiment, while LASSO-FS is not as sensitive to those choices. Furthermore, the fact that hyper-parameters are optimized based on performance naturally tends to favor methods that select more variables, putting LASSO-FS at a disadvantage in terms of interpretability.

4.5. Fixing the Number of Selected Variables

As confirmed by the previous experiment, there are two main trade-offs for feature selection algorithms: (a) the number of selected variables, with fewer variables leading to more interpretable results, and (b) the predictive ability of the selected variables, with more variables typically leading to better results. In general, algorithms that select more variables also tend to perform better in terms of predictive performance. Because of that, *we performed a comparison where algorithms are forced the select the same number of variables. That way, the predictive performance of algorithms can be compared on equal footing.*

We will compare FBED^K and LASSO-FS, when both are limited to select the same number of features. FBS is not included in the comparison, as we have already shown in

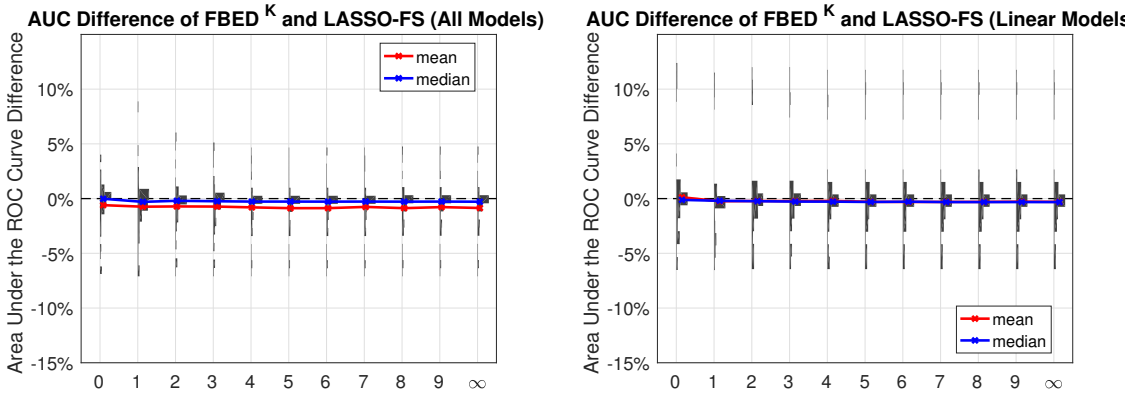


Figure 5: **LASSO-FS with limit on selected variables:** The x-axis shows the distribution of the difference in AUC of FBED^K and LASSO-FS, with positive values indicating that FBED^K performs better. The y-axis corresponds to value of K used by FBED^K. The mean and median values are shown in red and blue respectively. The average difference using non-linear models is 0.78%, and 0.23% when using only linear models. The difference can be explained by the arcene dataset, where LASSO-FS outperforms FBED^K even when selecting the same number of variables. A more detailed explanation is given in the main text.

Section 4.3 that FBS and FBED^K exhibit similar predictive performance with a similar number of selected features, with FBED^K being orders of magnitude faster. MMPC was not included, because neither MMPC nor FBED^K allow to set the number of features to select.

In order to perform the comparison, we executed FBED^K for multiple hyper-parameter values (the ones given in the experimental setup) and then executed LASSO-FS with the constraint to select the same number of variables as FBED^K did. As it was not always possible to select the exact same number of variables, we identified the solution of LASSO-FS with at least as many variables as FBED^K. Except for a few cases where LASSO-FS selected 1 more variable than FBED^K, both methods selected the same number of variables. As a final comment, we note that the above experiment does not favor FBED^K over LASSO-FS, as no optimization over its hyper-parameter values is performed. The reason we did not use a fixed number of variables M to select is because there is no easy way to select exactly M variables using FBED^K.

The comparison was performed similarly to the one between FBED^K and FBS in Section 4.3. Specifically, for a given K and threshold α, we computed the difference in AUC obtained by FBED^K and LASSO-FS. Thus, the only hyper-parameter optimization performed was over predictive models and not over the hyper-parameters of FBED^K. The results are shown in Figure 5. The x-axis corresponds to the difference in AUC between FBED^K and LASSO-FS, while the y-axis indicates the value K of FBED^K. We can see that, overall, both methods perform very similarly, regardless of whether linear models or non-linear models were used. For non-linear models, LASSO-FS outperforms FBED^K on average (over all K and α) by 0.78%, while using linear models the difference drops to

0.23%. In terms of median difference in AUC, a metric which is more stable with the respect to the datasets and hyper-parameter values used, both algorithms perform almost identically. Those results also agree with previous comparisons between forward selection and LASSO-FS (Hastie et al., 2017).

We investigated the difference in performance and found that it can be attributed to the arcene dataset. By removing this dataset, LASSO-FS outperforms FBED^K by 0.32% on average using non-linear models, while for linear models FBED^K performs better by 0.22%. Note that, arcene is the dataset which contains the fewest number of samples, while also containing a large number of variables. It contains 200 samples and 10000 variables, and only 120/160 are used for training and validation respectively. Theoretical results by Ng (2004) show that LASSO performs well in settings with low sample size and many irrelevant variables, as is the case for the arcene dataset. One possible explanation for the lower performance of FBED^K on arcene is that forward selection based procedures use more effective degrees of freedom (Hastie et al., 2017, Figure 1), thus requiring more sample size to have sufficient statistical power to pick up weak signals. It would be interesting to study this effect in more depth, but it is out of the scope of the current paper. In summary, *LASSO-FS and FBED^K perform similarly when the number of variables to select is the same.*

4.6. Simulation Study on the Multiple Testing Problem

The idea of early dropping of variables used by FBED^K does not only reduce the running time, but also reduces the problem of multiple testing, in some sense. Specifically, it reduces the number of variables falsely selected due to type I errors. In general, the number of false selections is related to the total number of variables considered in all forward iterations. Thus, the effect highly depends on the value of K used by FBED^K, with higher values of K leading to more false selections. We show this for FBED^0 by considering a simple scenario, where none of the candidate variables are predictive for the outcome. Then, in the worst case, FBED^0 will select about $\alpha \cdot p$ of the variables on average (where α is the significance level), since all other variables will be dropped in the first iteration. This stems from the fact that, under the null hypothesis of conditional independence, the p-values are uniformly distributed. In practice, the number of selected variables will be even lower, as FBED^0 will keep dropping variables after each variable inclusion. On the other hand, FBS may select a much larger number of variables, since each variable is given the chance to be included in the output at each iteration and will often do so, simply by chance.

We performed a small simulation to investigate the behavior of FBED^0, FBED^1, FBED$^\infty$ and how they compare to FBS. We generated 500 normally distributed datasets with 1000 samples each, a uniformly distributed random binary outcome, and considered different variable sizes $p \in \{100, 200, 300, 400, 500\}$ and 5 significance levels α uniformly spaced in $[0.01, 0.1]$. All variables are generated randomly, and there is no dependency between any of them. Thus, a false positive rate of about α is expected, if no adjustment is done to control the false discovery rate. For each setting, we computed the ratio of false positives with respect to the expected number of false positives.

Figure 6 (top) shows how the ratio varies for sample sizes 100 and 500 with increasing α, and Figure 6 (bottom) shows how the ratio varies for α 0.01 and 0.1 with increasing number

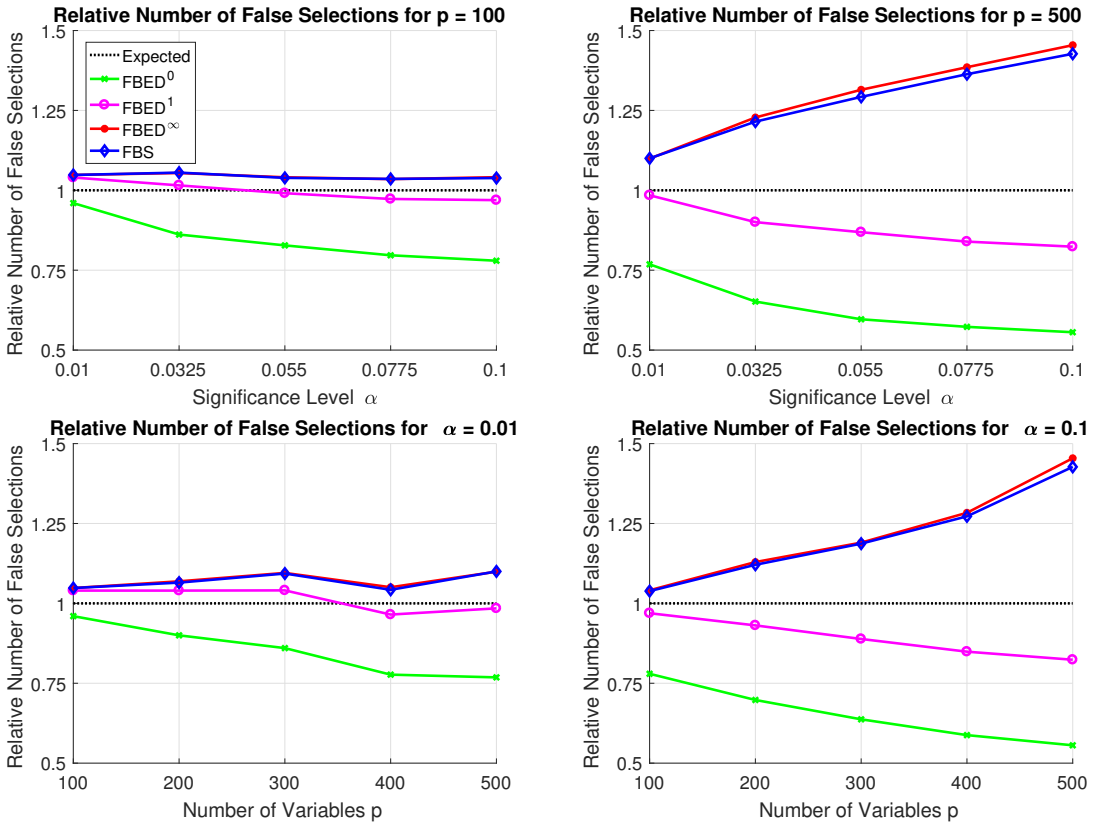


Figure 6: The figures show the relative number of false selections by each algorithm on randomly generated data. The expected number of false selections is $\alpha \cdot p$, where α is the significance level and p the number of variables. The numbers are computed as the ratio between the average number of selected variables to the expected false positives. FBED^0 and FBED^1 typically select fewer variables than expected, and their behavior improves with increasing α and p. FBED$^\infty$ and FBS on the other hand select more false positive variables, getting worse with larger values of α or on datasets with more variables.

of variables. In all cases, FBED^0 and FBED^1 select fewer false positives than expected, and their behavior improves both with increasing α and number of variables. FBED$^\infty$ and FBS perform almost identically, and tend to select more variables. We also observe that the number of false positives increases both with α and with the number of variables. Thus, *in case one is interested to limit the number of false selection, we recommend running FBED^K with a small value of K.*

5. Conclusion

We presented the early dropping heuristic to speed-up the forward-backward feature selection algorithm, which gives rise to a family of algorithms, called forward-backward selection with early dropping (FBED^K). Early dropping is a simple heuristic that leads to orders of magnitude speed-up, especially in high-dimensional datasets, while still maintaining the

theoretical guarantees of forward-backward selection. We prove that FBED^1 and FBED$^\infty$ identify the optimal solution (Markov blanket) if the distribution of the data can be faithfully represented by a Bayesian network or maximal ancestral graph respectively, similar to the standard forward-backward selection (FBS).

A useful property of FBED^K is that it is a general algorithm that can be adapted to handle different variable types (for example, continuous, categorical, ordinal), cross-sectional and time-course data, linear and non-linear dependencies, as well as different analysis tasks (for example, regression, classification, survival analysis) by using an appropriate conditional independence test. In contrast, algorithms like LASSO (Tibshirani, 1996), although being computationally fast and performing well in terms of predictive performance for common problems like regression and classification, are not as general (Meier et al., 2008; Schelldorfer et al., 2011; Ivanoff et al., 2016) and are computationally demanding for some problems (Fan et al., 2010; Groll and Tutz, 2014; Tsagris et al., 2018).

In experiments we demonstrate that FBED^K behaves similarly to FBS in terms of predictive performance and number of selected variables, while being 1-2 orders of magnitude faster. Compared to other feature selection algorithms like LASSO (Tibshirani, 1996) and MMPC (Tsamardinos et al., 2003a), FBED^K has competitive predictive performance, while selecting the fewest variables, which is especially important if feature selection is performed for knowledge discovery. An interesting result is that FBED^K and LASSO perform about equally well, when limited to select the same number of variables. This, combined with the fact that FBED^K is more general, makes it an attractive alternative to LASSO, especially for problems where no efficient solution to the LASSO problem exists.

Acknowledgments

We would like to thank Vincenzo Lagani and Michalis Tsagris for their helpful comments. This work was funded by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 617393.

Appendix A. Bayesian Networks and Maximal Ancestral Graphs

We will briefly introduce Bayesian networks and maximal ancestral graphs, which we will use to show theoretical properties of FBED^K.

A **directed acyclic graph** (DAG) is a graph that only contains directed edges ($\rightarrow$) and has no directed cycles. A **directed mixed graph** is a graph that, in addition to directed edges also contains bi-directed edges ($\leftrightarrow$). The graphs contain no self-loops, and vertices can be connected only by a single edge. Two vertices are called **adjacent** if they are connected by an edge. An edge between X and Y is called **into** Y if $X \rightarrow Y$ or $X \leftrightarrow Y$. A **path** in a graph is a sequence of unique vertices $\langle V_1, \ldots, V_k \rangle$ such that each consecutive pair of vertices is adjacent. The first and last vertices in a path are called **endpoints**. A path is called directed if $\forall 1 \leq i \leq k$, $V_i \rightarrow V_{i+1}$. If $X \rightarrow Y$ is in a graph, then X is a **parent** of Y and Y a **child** of X. A vertex W is a **spouse** of X, if both share a common child. A vertex X is an **ancestor** of Y, and Y is a **descendant** of X, if $X = Y$ or there is a directed path from X to Y. A triplet $\langle X, Y, Z \rangle$ is called a **collider** if Y is adjacent to X and Z, and both, X and Z are into Y. A triplet $\langle X, Y, Z \rangle$ is called **unshielded** if Y is

adjacent to X and Z, but X and Z are not adjacent. A path p is called a **collider path** if every non-endpoint vertex is a collider on p.

Bayesian networks (BNs) consist of a DAG $\mathcal{G}$ and a probability distribution $\mathcal{P}$ over a set of random variables $\mathbf{V}$. Each such variable is represent by a vertex in $\mathcal{G}$, and thus, the terms variable and vertex will be used interchangeably. The DAG represents dependency relations between variables in $\mathbf{V}$ and is linked with $\mathcal{P}$ through the **Markov condition**, which states that each variable is conditionally independent of its non-descendants given its parents. Those are not the only independencies encoded in the DAG; the Markov condition entails additional independencies, which can be read from the DAG using a graphical criterion called **d-separation** (Verma and Pearl, 1988; Pearl, 1988). In order to present the d-separation criterion we first introduce the notion of blocked paths. A (not necessarily directed) path p between two nodes X and Y is called **blocked** by a set of nodes $\mathbf{Z}$ if there is a node V on p that is a collider and, neither V nor any of its descendants are in $\mathbf{Z}$, or if V is not a collider and it is in $\mathbf{Z}$. If all paths between X and Y are blocked by $\mathbf{Z}$, then X and Y are **d-separated** given $\mathbf{Z}$; otherwise X and Y are **d-connected** given $\mathbf{Z}$. The **faithfulness condition** states that all and only those conditional independencies in $\mathcal{P}$ are entailed by the Markov condition applied to $\mathcal{G}$. In other words, the faithfulness condition requires that two variables X and Y are d-separated given a set of variables $\mathbf{Z}$ if and only if they are conditionally independent given $\mathbf{Z}$.

Bayesian networks are not closed under marginalization: a marginalized DAG, containing only a subset of the variables of the original DAG, may not be able to exactly represent the conditional independencies of the marginal distribution (Richardson and Spirtes, 2002). **Directed maximal ancestral graphs** (DMAGs) (Richardson and Spirtes, 2002) are an extension of BNs, which are able to represent such marginal distributions, that is, they admit the presence of latent confounders. The graphical structure of a DMAG is a directed mixed graph with the following restrictions: (i) it contains no directed cycles, (ii) it contains no almost directed cycles, that is, if $X \leftrightarrow Y$ then neither X nor Y is an ancestor of the other, and (iii) there is no primitive inducing path between any two non-adjacent vertices, that is, there is no path p such that each non-endpoint on p is a collider and every collider is an ancestor of an endpoint vertex of p. The d-separation criterion analogue for DMAGs is called the **m-separation criterion**, and follows the same definition.

A **Markov blanket** of a variable T is a **minimal** set of variables $\mathbf{MB}(T)$ that renders T conditionally independent of all remaining variables $\mathbf{V} \setminus \mathbf{MB}(T)$. In case faithfulness holds, and the distribution can be represented by a BN or DMAG, then the Markov blanket is **unique**. For a BN, the Markov blanket of T consists of its parents, children and spouses. For DMAGs it is slightly more complicated: the Markov blanket of T consists of its parents, children and spouses, as well as its district (all vertices that are reachable by bi-directed edges), the districts of its children and the parents of all districts (Richardson, 2003). An alternative definition is given next.

Definition 1 The Markov blanket of T in a BN or DMAG consists of all vertices adjacent to T, as well as all vertices that are reachable from T through a collider path.

A proof sketch follows. Recall that a collider path of length $k-1$ is of the form $X_1 * \rightarrow X_2 \dots X_{k-1} \leftarrow * X_k$, where the path between X_2 and X_{k-1} contains only bi-directed edges. Given this, it is easy to see that Definition 1 includes vertices directly adjacent to T, its spouses (collider path of length 2), and in the case of DMAGs, vertices D in the district of

T ($T \leftrightarrow \cdots \leftrightarrow D$), vertices D in the district of any children C of T ($T \rightarrow C \leftrightarrow \cdots \leftrightarrow D$), and all parents P of any vertex D in some district ($T* \rightarrow \cdots \leftrightarrow D \leftarrow P$). As the previous cases capture exactly all possibilities of nodes reachable from T through a collider path, Definition 1 does not include any additional variables that are not in the Markov blanket of T.

Appendix B. Proofs

We proceed by listing some axioms about conditional independence (Pearl, 2000), called **semi-graphoid** axioms, which will be useful later on. Those axioms are general, as they hold for any probability distribution. For all of the proofs we assume that the algorithms have access to an **independence oracle** that can perfectly determine whether a given conditional dependence or independence holds. Furthermore, in all proofs we will use the terms d-connected/m-connected (d-separated/m-separated) and dependent (independent) interchangeably; this is possible due to the faithfulness assumption.

Symmetry	$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) \Rightarrow (\mathbf{Y} \perp \mathbf{X} \mid \mathbf{Z})$
Decomposition	$(\mathbf{X} \perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \Rightarrow (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp \mathbf{W} \mid \mathbf{Z})$
Weak Union	$(\mathbf{X} \perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) \Rightarrow (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W})$
Contraction	$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp \mathbf{W} \mid \mathbf{Y} \cup \mathbf{Z}) \Rightarrow (\mathbf{X} \perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z})$

Using those axioms we prove the following lemma.

Lemma 7 *Let A, T be variables and $\mathbf{B}$, $\mathbf{C}$ sets of variables. Then* $(T \perp A \mid \boldsymbol{B} \cup \mathbf{C}) \wedge (T \perp \boldsymbol{B} \mid \mathbf{C}) \Rightarrow (T \perp A \mid \mathbf{C})$ *holds for any such variables.*

Proof

$(T \perp A \mid \mathbf{B} \cup \mathbf{C}) \wedge (T \perp \mathbf{B} \mid \mathbf{C}) \Rightarrow$ (**Contraction**)
$(T \perp A \cup \mathbf{B} \mid \mathbf{C}) \Rightarrow$ (**Decomposition**)
$(T \perp A \mid \mathbf{C}) \wedge (T \perp \mathbf{B} \mid \mathbf{C})$

■

The following lemma will be useful for proving some of the theorems.

Lemma 8 *Let $\mathbf{S}$ be a set of variables selected for some target T and $\mathbf{R} = \mathbf{V} \setminus \mathbf{S}$. Assume that $\forall V_r \in \mathbf{R}$ $(T \perp V_r \mid \mathbf{S})$ holds. Then, if $\exists V_s \in \mathbf{S}$ such that $(T \perp V_s \mid \mathbf{S} \setminus V_s)$ holds, $\forall V_r \in \mathbf{R}$ $(T \perp V_r \mid \mathbf{S} \setminus V_s)$ also holds.*

Proof We are given that $\forall V_r \in \mathbf{R}$ $(T \perp V_r \mid \mathbf{S})$ holds. By applying Lemma 7 to each variable in $V_r \in \mathbf{R}$ with $A = V_r$, $\mathbf{B} = \{V_s\}$ and $\mathbf{C} = \mathbf{S} \setminus V_s$, we get that $(T \perp V_r \mid V_s \cup (\mathbf{S} \setminus V_s)) \wedge (T \perp V_s \mid \mathbf{S} \setminus V_s) \Rightarrow (T \perp V_r \mid \mathbf{S} \setminus V_s)$ holds for any such V_r, which concludes the proof. ■

To put it simple, Lemma 8 states that if we remove any variable V_s from a set of selected variables $\mathbf{S}$ by conditioning on $\mathbf{S} \setminus V_s$, no variable that is not in $\mathbf{S}$ becomes conditionally dependent with T given $\mathbf{S} \setminus V_s$. In practice this means that removing variables using backward selection from a set of variables selected by forward selection will not create any additional conditional dependencies, meaning that we do not have to reconsider them again.

Proof of Corollary 3
Proof To show that $\mathbf{S}$ is minimal, we have to show the following

i $\forall V_s \in \mathbf{S}\ (T \not\perp V_s \mid \mathbf{S} \setminus V_s)$ (*No variable can be removed*)

ii $\forall V_r \in \mathbf{V}_{\mathcal{D}} \setminus \mathbf{S}, (T \perp V_r \mid \mathbf{S})$ (*No variable can be added*)

Proof of (i): This holds trivially, as backward selection removes any variable $V_s \in \mathbf{S}$ if $(T \perp V_s \mid \mathbf{S} \setminus V_s)$ holds.
Proof of (ii): We know that after the termination of forward selection, no variable can be added, that is, $\forall V_r \in \mathbf{R}\ (T \perp V_r \mid \mathbf{S})$ holds. Given that, Lemma 8 can be repeatedly applied after each variable removal by backward selection, and thus no variable in $\mathbf{R}$ can be added to $\mathbf{S}$. ■

Proof of Corollary 4
Proof As is the case with FBS, the forward selection phase of FBED$^{\infty}$ stops if no more variables can be included. Using this fact, the proof is identical to the one of Theorem 3. ■

Proof of Theorem 5
Proof In the first run of FBED^{1}, all variables that are adjacent to T (that is, its parents and children) will be selected, as none of them can be d-separated from T by any set of variables. In the next run, all variables connected through a collider path of length 2 (that is, the spouses of T) will become d-connected with T, since the algorithm conditions on all selected variables (including its children), and thus will be selected. The resulting set of variables includes the Markov blanket of T, but may also include additional variables. Next we show that all additional variables will be removed by the backward selection phase. Let MB(T) be the Markov blanket of T and $\mathbf{S_{ind}} = \mathbf{S} \setminus \text{MB}(T)$ be all selected variables not in the Markov blanket of T. By definition, $(T \perp \mathbf{X} \mid MB(T))$ holds for any set of variables $\mathbf{X}$ not in MB(T), and thus also for variables $\mathbf{S_{ind}}$. By applying the weak union graphoid axiom, one can infer that $\forall S_i \in \mathbf{S_{ind}}, (T \perp S_i \mid MB(T) \cup \mathbf{S_{ind}} \setminus S_i)$ holds, and thus some variable S_j will be removed in the first iteration. Using the same reasoning and the definition of a Markov blanket, it can be shown that all variables in $\mathbf{S_{ind}}$ will be removed from MB(T) at some iteration. To conclude, it suffices to use the fact that variables in MB(T) will not be removed by the backward selection, as they are not conditionally independent of T given the remaining variables in MB(T). ■

Proof of Theorem 6
Proof In the first run of FBED$^{\infty}$, all variables that are adjacent to T (that is, its parents, children and variables connected with T by a bi-directed edge) will be selected, as none

of them can be m-separated from T by any set of variables. After each run additional variables may become admissible for selection. Specifically, after k runs all variables that are connected with T by a collider path of length k will become m-connected with T, and thus will be selected; we prove this next. Assume that after k runs all variables connected with T by a collider path of length at most $k-1$ have been selected. By conditioning on all selected variables, all variables that are into some selected variable connected with T by a collider path will become m-connected with T. This is true because conditioning on a variable Y in a collider $\langle X, Y, Z\rangle$ m-connects X and Z. By applying this on each variable on some collider path, it is easy to see that its end-points become m-connected. Finally, after applying the backward selection phase, all variables that are not in the Markov blanket of T will be removed; the proof is identical to the one used in the proof of Theorem 5 and thus will be omitted. ■

Appendix C. Bootstrap Test For Comparing Algorithms

We used bootstrapping to compute the probability that algorithm A_i is better/worse or equal than all others in terms of some measure of interest f (for example, AUC), that is $P(\wedge_{j\neq i} f(A_i) \geq f(A_j))$. The procedure is described next. Let $f_{i,k}$ denote the measure of interest of algorithm i on test set k. We resample with replacement $B = 100000$ times the test sets and compute f, denoted as $f_{i,k,b}$ for the b-th sample of algorithm i and test set k. Then, $f_{i,k,b}$ are averaged over test sets, obtaining $\hat{f}_{i,b}$. The probability $P(\wedge_{j\neq i} f(A_i) \geq f(A_j))$ is then computed as $1/B \sum_b I(\hat{f}_{i,b} \geq \max_j \hat{f}_{j,b})$, where I is the indicator function.

Appendix D. Running Times of FBED^K, FBS, MMPC and LASSO

Table 4 shows the running time of each feature selection algorithm and configuration, on all datasets. The values correspond to a single run on the complete dataset. All runs were performed on a single machine, and no runs were performed simultaneously. For FBED^K we only show running times for $K \in \{0, 1, 3, \infty\}$, and for LASSO-FS we show results for $\lambda_{max} \in \{25, 100, 500\}$. MMPC for a given value of $maxK$ is denoted as MMPC^{maxK}.

It can clearly be seen that LASSO-FS is the fastest in large datasets, irrespective of the number of λ values used. For smaller datasets (musk, sylva, madelon, secom, gina and hiva), FBED^0 and FBED^1 are often at least as fast as LASSO-FS. FBS and MMPC with $maxK \geq 3$ are the slowest among all algorithms. For the gisette and nova datasets, MMPC with $maxK \geq 3$ fails to terminate after a timeout limit of 2 days. For $maxK \leq 2$ MMPC often has competitive performance with the other algorithms. We note that MMPC was designed specifically for low sample sizes and high dimensional data, such as data from biological domains which contain a few tens or hundreds of samples and tens of thousands of variables, explaining the high running times on the datasets considered in our experiments, most of which contain thousands of samples.

The large difference between the running time of LASSO-FS and the other algorithms can largely be attributed to their implementations. For LASSO-FS the glmnet implementation was used, which is highly optimized and written in FORTRAN. In contrast, for FBED^K, FBS and MMPC we used a custom logistic regression implementation written in Matlab.

A difference of 1-2 orders of magnitude can be expected between the same implementation in a low-level language such as FORTRAN, C or C++ and higher-level languages such as Matlab. Therefore, we would expect that an implementation in a lower-level language would perform similarly to LASSO-FS. Of course, LASSO-FS has the advantage that it returns the whole solution path, and thus would still be faster in practice if hyper-parameter optimization is also performed.

Table 4: Running times in seconds on the full datasets.

	Algorithm	musk	sylva	madelon	secom	gina	hiva	gisette	p53	arcene	nova	dexter	dorothea
$\alpha = 0.001$	FBED^0	1.5	3.3	0.3	0.3	5.0	2.8	56.7	48.6	2.6	21.8	6.2	133.9
	FBED^1	3.2	7.6	0.7	0.3	11.3	2.8	186.5	137.8	6.0	90.6	20.1	377.9
	FBED^3	9.1	7.6	0.7	0.3	24.8	2.8	546.3	242.3	14.5	308.0	42.8	1033.0
	FBED$^\infty$	22.1	7.6	0.7	0.3	24.8	2.8	2962.7	242.3	14.5	536.1	42.8	1855.3
	FBS	46.0	36.3	1.3	1.4	140.4	22.4	5773.0	1186.4	66.8	3486.3	309.2	7233.3
	MMPC^1	3.9	9.5	0.3	0.5	21.2	5.0	309.2	115.0	4.1	112.9	14.9	91.8
	MMPC^2	7.9	17.9	0.3	0.5	84.0	5.4	1885.4	163.4	4.1	1524.9	17.9	92.0
	MMPC^3	7.9	20.9	0.3	0.6	171.2	5.6	N/A	175.9	4.1	N/A	19.9	92.0
	MMPC^4	9.8	21.8	0.3	0.5	292.4	5.6	N/A	210.7	4.1	N/A	21.3	93.1
$\alpha = 0.005$	FBED^0	1.8	3.9	0.2	0.3	6.7	3.3	79.2	60.4	2.8	31.3	6.5	139.9
	FBED^1	5.4	8.3	0.6	0.8	14.2	10.7	244.2	158.5	10.8	114.1	22.3	457.0
	FBED^3	11.7	12.9	0.6	1.6	36.2	28.1	756.1	418.0	19.9	435.7	77.9	865.6
	FBED$^\infty$	29.7	12.9	0.6	1.6	49.2	28.1	1105.3	700.5	19.9	1426.2	202.2	865.6
	FBS	82.2	50.4	2.1	3.1	192.7	82.7	10932.7	3864.6	66.8	7015.8	503.0	7233.3
	MMPC^1	4.6	10.4	0.3	0.6	25.8	6.0	384.6	142.2	4.7	238.3	17.4	106.3
	MMPC^2	10.5	24.5	0.3	0.6	126.0	6.4	2869.2	230.4	4.7	3993.7	30.2	107.1
	MMPC^3	10.2	43.6	0.3	0.6	307.1	6.9	N/A	330.3	4.7	N/A	36.6	115.0
	MMPC^4	11.5	45.0	0.3	0.6	470.4	6.9	N/A	469.6	4.7	N/A	54.2	126.6
$\alpha = 0.01$	FBED^0	2.0	4.3	0.2	0.3	8.3	3.6	88.0	64.2	3.0	39.0	6.8	144.7
	FBED^1	5.3	8.9	0.2	0.8	16.8	11.9	261.1	164.3	11.8	123.3	25.2	463.0
	FBED^3	16.5	14.1	0.2	1.7	46.0	32.3	824.3	450.7	11.8	505.8	95.4	884.6
	FBED$^\infty$	32.5	14.1	0.2	2.1	97.6	81.8	824.3	1615.0	11.8	760.0	136.7	884.6
	FBS	94.3	61.0	2.5	4.4	234.8	150.2	10932.7	5160.2	66.8	8811.8	503.0	7233.3
	MMPC^1	5.0	12.8	0.3	0.6	28.0	7.0	426.9	161.3	5.1	347.0	19.6	119.1
	MMPC^2	12.4	28.7	0.3	0.6	141.0	7.8	3570.4	269.8	5.3	8262.0	41.6	128.8
	MMPC^3	12.7	45.1	0.4	0.6	426.0	9.1	N/A	460.1	5.3	N/A	63.5	146.0
	MMPC^4	13.3	62.0	0.4	0.7	573.4	9.8	N/A	728.7	5.3	N/A	104.6	200.6
$\alpha = 0.05$	FBED^0	3.3	5.6	0.4	0.5	14.2	6.4	140.2	95.2	3.9	111.9	9.5	201.0
	FBED^1	6.4	11.4	0.9	1.1	34.9	17.0	398.6	242.8	3.9	380.7	42.0	644.6
	FBED^3	25.6	24.8	2.0	2.7	82.8	50.5	398.6	779.5	3.9	957.2	42.0	644.6
	FBED$^\infty$	76.1	49.9	4.6	7.7	221.6	206.5	398.6	7688.4	3.9	957.2	42.0	644.6
	FBS	191.9	108.6	18.6	15.4	1243.4	3490.9	10932.7	31681.6	66.8	10119.3	621.0	7233.3
	MMPC^1	6.4	15.0	0.6	0.8	38.3	16.6	604.2	266.6	7.9	1050.6	48.9	406.0
	MMPC^2	15.8	48.1	1.5	1.1	222.2	22.4	6279.9	566.7	10.2	45093.2	204.1	750.2
	MMPC^3	22.0	152.2	3.3	1.1	921.7	46.3	N/A	977.6	12.7	N/A	525.3	1499.5
	MMPC^4	23.3	209.4	5.6	1.3	1499.4	68.8	N/A	2103.9	12.2	N/A	624.8	3818.6
$\alpha = 0.1$	FBED^0	4.1	7.6	0.8	0.7	20.1	11.0	213.0	128.4	4.7	162.3	14.1	246.7
	FBED^1	8.9	17.3	1.8	1.9	49.0	37.2	858.6	359.8	4.7	428.5	53.7	246.7
	FBED^3	26.8	40.0	4.5	4.0	127.5	289.0	858.6	1280.7	4.7	980.2	53.7	246.7
	FBED$^\infty$	84.6	64.9	7.5	40.3	889.1	347.7	858.6	3476.2	4.7	980.2	53.7	246.7
	FBS	253.7	246.7	48.7	29.0	3688.9	9997.9	10932.7	31681.6	66.8	10784.8	621.0	7669.2
	MMPC^1	7.0	16.9	1.2	1.3	44.2	28.1	733.7	388.2	13.3	2063.8	117.0	1033.3
	MMPC^2	17.7	96.9	7.3	2.6	278.0	68.4	8313.7	955.9	24.5	122716.3	715.0	3873.2
	MMPC^3	25.2	187.9	52.4	4.2	1315.8	160.0	N/A	1675.6	29.5	N/A	2065.3	10196.9
	MMPC^4	33.6	371.5	236.8	6.3	2959.9	405.1	N/A	4359.6	39.5	N/A	3761.7	24259.0
	LASSO-FS^{25}	12.2	7.0	2.7	6.0	12.3	13.7	6.9	118.6	0.2	1.6	0.4	4.0
	LASSO-FS^{100}	12.2	11.3	4.1	7.3	16.2	16.3	8.9	65.0	0.3	3.3	0.8	11.0
	LASSO-FS^{500}	10.5	19.7	4.9	11.2	22.5	34.1	24.5	133.4	1.0	14.7	3.5	51.1

References

Alan Agresti. *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2nd edition, 2002.

Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281, 1973.

Constantin F. Aliferis, Ioannis Tsamardinos, and Alexander Statnikov. HITON: a novel Markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings*, volume 2003, page 21. American Medical Informatics Association, 2003.

Constantin F. Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(Jan):171–234, 2010.

Thomas Blumensath and Mike E. Davies. On the difference between Orthogonal Matching Pursuit and Orthogonal Least Squares. Technical report, University of Edinburgh, 2007.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13:27–66, January 2012.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology*, 2(3):27, 2011.

Jiahua Chen and Zehua Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

Sheng Chen, Stephen A Billings, and Wan Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50 (5):1873–1896, 1989.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Samuel A. Danziger, S. Joshua Swamidass, Jue Zeng, Lawrence R. Dearth, Qiang Lu, Jonathan H. Chen, Jianlin Cheng, Vinh P. Hoang, Hiroto Saigo, Ray Luo, et al. Functional census of mutation sequence spaces: the example of p53 cancer rescue mutants. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):114–125, 2006.

Geoffrey M. Davis, Stephane G. Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions. *Optical engineering*, 33(7):2183–2192, 1994.

Thomas G. Dietterich, Ajay N. Jain, Richard H. Lathrop, and Tomas Lozano-Perez. A comparison of dynamic reposing and tangent distance for drug activity prediction. *Advances in Neural Information Processing Systems*, pages 216–216, 1994.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Jianqing Fan, Yang Feng, Yichao Wu, et al. High-dimensional variable selection for Cox's proportional hazards model. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown*, pages 70–86. Institute of Mathematical Statistics, 2010.

Yingying Fan and Cheng Yong Tang. Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):531–552, 2013.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *AutoML: Methods, Sytems, Challenges*, chapter 1, pages 3–37. Springer, December 2018. To appear.

Livio Finos, Chiara Brombin, and Luigi Salmaso. Adjusting stepwise p-values in generalized linear models. *Communications in Statistics-Theory and Methods*, 39(10):1832–1846, 2010.

Peter L. Flom and David L. Cassell. Stopping stepwise: Why stepwise and similar selection methods are bad, and what you should use. In *NorthEast SAS Users Group Inc 20th Annual Conference*, 2007.

Robert V. Foutz and R. C. Srivastava. The performance of the likelihood ratio test when the model is incorrect. *The Annals of Statistics*, 5(6):1183–1194, 1977.

Andreas Groll and Gerhard Tutz. Variable selection for generalized linear mixed models by L1-penalized estimation. *Statistics and Computing*, 24(2):137–154, 2014.

Max Grazier G'Sell, Stefan Wager, Alexandra Chouldechova, and Robert Tibshirani. Sequential selection procedures and false discovery rate control. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(2):423–444, 2016.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2004.

Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann. Performance prediction challenge. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1649–1656. IEEE, 2006a.

Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Feature selection with the CLOP package. Technical report, 2006b.

Mazin Abdulrasool Hameed. Comparative analysis of orthogonal matching pursuit and least angle regression. Master's thesis, Michigan State University, Electrical Engineering, 2012.

Frank Harrell. *Regression Modeling Strategies*. Springer, corrected edition, January 2001.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2nd edition, 2009.

Trevor Hastie, Robert Tibshirani, and Ryan J Tibshirani. Extended comparisons of best subset selection, forward stepwise selection, and the lasso. *arXiv preprint arXiv:1707.08692*, 2017.

Jing-Shiang Hwang and Tsuey-Hwa Hu. A stepwise regression algorithm for high-dimensional variable selection. *Journal of Statistical Computation and Simulation*, 85 (9):1793–1806, 2015.

Stéphane Ivanoff, Franck Picard, and Vincent Rivoirard. Adaptive lasso and group-lasso for functional Poisson regression. *Journal of Machine Learning Research*, 17(Jan):1903–1948, 2016.

George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann, 1994.

Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8(Mar):613–636, 2007.

Yongdai Kim, Sunghoon Kwon, and Hosik Choi. Consistent model selection criteria on high dimensions. *Journal of Machine Learning Research*, 13(Apr):1037–1057, 2012.

Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw-Hill/Irwin, 5 edition, August 2004.

Vincenzo Lagani, Giorgos Athineou, Alessio Farcomeni, Michail Tsagris, and Ioannis Tsamardinos. Feature selection with the R package MXM: Discovering statistically equivalent feature subsets. *Journal of Statistical Software*, 80(7), 2017.

Jinchi Lv and Jun S. Liu. Model selection principles in misspecified models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):141–167, 2014.

Dimitris Margaritis. Toward provably correct feature selection in arbitrary domains. In *Advances in Neural Information Processing Systems*, pages 1240–1248, 2009.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 2000.

Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group Lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2008.

Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44. IEEE, 1993.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

Judea Pearl. *Causality, Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, U.K., 2000.

Peter Peduzzi, John Concato, Elizabeth Kemper, Theodore R. Holford, and Alvan R. Feinstein. A simulation study of the number of events per variable in logistic regression analysis. *Journal of clinical epidemiology*, 49(12):1373–1379, 1996.

Junyang Qian, Ttrevor Hastie, Jerome Friedman, Rob Tibshirani, and Noah Simon. Glmnet for Matlab, 2013.

Thomas Richardson. Markov properties for acyclic directed mixed graphs. *Scandinavian Journal of Statistics*, 30(1):145–157, 2003.

Thomas Richardson and Peter Spirtes. Ancestral graph Markov models. *Annals of Statistics*, pages 962–1030, 2002.

Jürg Schelldorfer, Peter Bühlmann, and Sara Van De Geer. Estimation for high-dimensional linear mixed-effects models using L1-penalization. *Scandinavian Journal of Statistics*, 38 (2):197–214, 2011.

Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2): 461–464, 1978.

Peter Spirtes, Clark N. Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2 edition, 2000.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 267–288, 1996.

Ryan J. Tibshirani, Jonathan Taylor, Richard Lockhart, and Robert Tibshirani. Exact post-selection inference for sequential regression procedures. *Journal of the American Statistical Association*, 111(514):600–620, 2016.

Michail Tsagris, Vincenzo Lagani, and Ioannis Tsamardinos. Feature selection for high-dimensional temporal data. *BMC bioinformatics*, 19(1):17, 2018.

Ioannis Tsamardinos and Constantin F. Aliferis. Towards principled feature selection: relevancy, filters and wrappers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the Ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678. ACM, 2003a.

Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander R. Statnikov. Algorithms for large scale Markov blanket discovery. In *FLAIRS conference*, volume 2, 2003b.

Ioannis Tsamardinos, Giorgos Borboudakis, Pavlos Katsogridakis, Polyvios Pratikakis, and Vassilis Christophides. A greedy feature selection algorithm for Big Data of high dimensionality. *Machine Learning*, Aug 2018a.

Ioannis Tsamardinos, Elissavet Greasidou, and Giorgos Borboudakis. Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. *Machine Learning*, May 2018b.

Thomas S. Verma and Judea Pearl. Causal Networks: Semantics and Expressiveness. In *Proceedings, 4^{th} Workshop on Uncertainty in Artificial Intelligence*, pages 352–359, August 1988.

Eric Vittinghoff and Charles E. McCulloch. Relaxing the rule of ten events per variable in logistic and Cox regression. *American journal of epidemiology*, 165(6):710–718, 2007.

Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: Journal of the Econometric Society*, pages 307–333, 1989.

Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

Halbert White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50 (1):1–25, 1982.

Samuel S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, March 1938.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 804–813, 2011.

Yongli Zhang and Xiaotong Shen. Model selection procedure for high-dimensional data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(5):350–358, 2010.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67:301–320, 2005.