NOVEL METHODS FOR VARIABLE SELECTION IN NON-FAITHFUL DOMAINS, UNDERSTANDING SUPPORT VECTOR MACHINES, LEARNING REGIONS OF BAYESIAN NETWORKS, AND PREDICTION UNDER MANIPULATION

By

Laura E. Brown

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Biomedical Informatics

December 2009

Nashville, Tennessee

Approved:

Dr. Ioannis Tsamardinos

Dr. Douglas Hardin

Dr. Constantin F. Aliferis

Dr. Daniel Masys

ACKNOWLEDGEMENTS

I am appreciative and would like to recognize the National Library of Medicine for my initial support through the Vanderbilt Training Program. In addition, the department and Dr. Hardin who generously supported the continuation of my work through TA and research funding.

This work has depended on the support, counsel, and aid of many individuals. First, I would like to thank the members of my committee who have contributed to the design on this research. I greatly appreciate their input and suggestions to improve and advance this research. Additionally, I would like to acknowledge the work of Alexander Statnikov, a fellow student and programmer for the Discovery Systems Laboratory. Alex is responsible for a large code base from which many of my systems and algorithms are based. Most emphatically, I wish to thank and commend Dr. Ioannis Tsamardinos and Dr. Douglas Hardin who simultaneously worked as my advisor for the completion of my research. Without their continual mentorship, patience, support, and insights this work might not have been completed.

Ultimately, this work was not possible without the support of all my family and friends. To the DBMI students, faculty, and staff thanks for everyone's help throughout the years in classes, research, and life. Mom, Dad, and Ian, you have nurtured and guided me in continuing my education. If I can prove to have one-tenth of your wisdom, discipline, skills, and knowledge I would consider myself fortunate and blessed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The fields of biomedicine have seen an extensive growth in scientific methods, measurements, procedures, collection, and storage resulting in the abundance of new, high-dimensional data sets. Such data comes from varied domains including the "omics" fields (e.g., genomic, proteomics, etc.), new high-resolution images, pharmaceutical studies, or clinical enterprises. With the growth of the volume and variety of data being collected and aggregated, new methods must be created and evaluated to aid analyses for the discovery of new information in each domain. In this thesis several novel computational techniques for discovering informative patterns and complex relationships in biomedical data are developed and investigated. Specifically, this thesis focuses on the following areas: (I.1) an algorithm for the identification of the top-weighted features in a polynomial Support Vector Machine model, (I.2) a variable selection method to identify the Markov Blanket in feature space, (I.3) a comparison of techniques (global and local) for learning a region of a Bayesian network, (I.4) and a strategy for making predictions under manipulation using the three techniques and methods developed in this thesis research.

The first research area (I.1) concentrates on determining what variables and combinations of variables (features) are important to a classification task when using a polynomial Support Vector Machine (SVM) model (Vapnik, 1995, 1998; Boser *et al.*, 1992). SVMs are a type of a kernel method that have been shown to be very successful on classification tasks for high-dimensional data (Joachims, 2002; Ling *et al.*, 2005; Li *et al.*, 2005; Ling *et al.*, 2005; Larranaga *et al.*, 2006). While SVMs have been used frequently for classification models, they are often treated as black boxes. This method attempts to make these models more understandable to a human user.

Next, the research focused on methods for variable selection (I.2) which have been studied and applied in a number of biomedical domains (Friedman *et al.*, 2000; Furey *et al.*, 2000; Guyon *et al.*, 2002a; Inza *et al.*, 2004; Yoo & Cooper, 2004; Wang *et al.*, 2005; Woolf *et al.*, 2005). Many methods

and approaches have been developed for this problem with each having advantages and limitations to their approach and application. This research combines the advantages of two approaches to create a new method that removes some limitations. First, the new method developed is designed to scale to domains with thousands up to hundreds of thousands of variables. Such high-dimensional data often occur as the result of the application of biological mass-throughput measurement methods. Second, the new method aims to improve the quality and comprehensibility of current kernel-based methods (an important family of methods that have demonstrated their ability to handle high-dimensional data). Finally, the new method is designed to discover important variables in specific data distributions where other state-of-the-art methods fail (Glymour & Cooper, 1999; Spirtes *et al.*, 2000).

The third research area focuses (I.3) on the discovery of Bayesian network structures (Pearl, 1988). Even the most advanced algorithms for learning the structure of Bayesian network have limitations in the number of variables (on the order of thousands of variables) to the structures learned (Tsamardinos *et al.*, 2006b). The limitations are both in terms of the quality of the learned structure and also the efficiency of the methods to produce the results. For example, a data set consisting of five thousands variables, could take several days of computation to learn the network structure. The new method works by focusing the learning of the structure to a region about a variable of interest. This new approach is expected to be more time-efficient, but the quality of the regions learned was unknown before the evaluation. The evaluation showed that the new method learned regions of equal or better quality to a traditional (global) approach.

Finally, using the above new methods and contemporary research, a principled submission (I.4) to the Causality Challenge tasks was developed (WCCI 2008 Causality Challenge, 2008). The focus of this challenge was on predicting the results of actions performed by an external agent. This publicly available challenge, a part of the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008) had over 1400 submissions by more than 30 teams. The submission used the formalism of Causal Bayesian Networks to model and induce causal relations and to make predictions about the effects of the manipulation of the variables. The overall strategy made use of the three

other techniques described in this thesis as well as developing theory to perform predictions under manipulation. The submission performed best on one of the four tasks presented (ranked first out of 30 teams and over 350 entries). In addition, the results and methods were carefully analyzed to discuss where the methods performed well and where they did not, on these real world data tasks. Further details on the four areas of focus are presented below.

I.1 Identification of Top-Weighted Features of Polynomial Support Vector Machine Models

Support Vector Machine (SVM) and kernel methods in general have been proven very successful methods for standard binary classification problems (Vapnik, 1998). Polynomial Support Vector Machine models of degree d are linear functions in a feature space of monomials of, at most, degree d (Boser *et al.*, 1992). However, the actual representation is typically in the form of support vectors and Lagrange multipliers. This representation of the model and the resulting classification function are efficient to compute but unsuitable for the human modeler to gain an understanding of the model's "logic". Specifically, it is difficult to determine from this representation which variables (input components of data space) or combinations of these variables (features) strongly affect the output of the classification function. Arguably, this is one reason why rule-learning classification, decision trees and other easier-to-interpret classifiers are sometimes preferred over the SVM classifier.

The classification function may also be expressed explicitly as a linear function involving all variables and combination of variables (determined by the Kernel function). This explicit representation allows easy interpretation and identification of the most important combinations of variables (called interaction terms in standard-statistical linear regression): these are the monomials with the largest absolute weight. This information can be used for human validation of the semantics of the model, visualization and for further feature selection. Unfortunately, while potentially helpful, it is time-prohibitive to compute the explicit representation for large models due to the explosion in the number of features, f, to consider, for m variables and degree d of the kernel there is $f = \binom{m+d}{d}$ number of features.

This thesis includes the design of a new algorithm that heuristically selects the most heavily

weighted (and thus most important for classification) constructed features of a polynomial SVM. The selected features may provide a new intuition into the behavior of the SVM and convey what features are important to a domain. For example, if this proposed new method returns the feature $X_1X_2X_3$, where X_i is the expression level of a gene i, as the most important feature for the prediction of the cancer type of a tissue, then a biologist can immediately infer that it is the interaction between these three genes that is the most important factor in the classification task (the effect of each variable alone may be masked by epistatic effects).

Sufficient conditions are provided for the heuristic algorithm to correctly return the top r weights. Even when the sufficient conditions fail, the research empirically shows that the returned weights closely approximate the true set of r top-weighted features when only examining a very small portion of the feature space. In addition, the new algorithm is shown to return predictive features by comparing the classification performance of the top-weighted features to models with all variables or subsets identified by variable selection methods on several real-world data sets.

Research Summary: *Support Vector Machine models are often used for classification tasks, but do not immediately allow for interpretation of important variables to the task. This research presents an efficient, heuristic method to identify the features of a polynomial Support Vector Machine model with the largest absolute weights. These features are expected to be important to the classification task.*

I.2 Variable Selection in Feature Space

Variable selection is often employed as a dimensionality reduction and discovery technique in the biomedical domain due to the high dimensional data sets. The goal of variable selection techniques is to reduce a high-dimensional data set, containing M variables, to a low-dimensional representation that is still highly predictive of the target variable but containing instead only m variables, where $m < M$. The new low-dimensional representation is expected to make evident patterns and hidden information that were previously obfuscated in the abundance of data and noise at the high-dimensional level.

Several variable selection techniques use kernel methods, often Support Vector Machines (SVMs) (Vapnik, 1998). The Recursive Feature Elimination (RFE) method will be used as a prototypical example (Guyon *et al.*, 2002b). Use of the kernel methods allows for the input data to be mapped to a different space where the data patterns and relations may be simpler; this new space is called the feature space and consists of constructed features. A constructed feature may be the product of the original variables, for example the feature X_1X_2 is a combination of the two input variables X_1 and X_2. The feature space is typically of even higher dimensionality than the original variable space, reaching numbers of billions of dimensions. Nevertheless, kernel methods remain computationally efficient because the mapping to feature space is performed implicitly.

Predictive variables however, do not necessarily have a direct causal relation with the target variable (e.g., a clinical outcome). Causal discovery methods exist that formalize the induction of causal relations (Aliferis *et al.*, 2003a; Tsamardinos *et al.*, 2003c,a; Peña *et al.*, 2007; Margaritis & Thrun, 1999; Koller & Sahami, 1996; Aliferis *et al.*, 2009a). A principled approach in variable selection is based on identifying the Markov Blanket of the target variable. A Markov Blanket of the target is defined as a minimal set, conditioned on which all other variables become independent of the target (Pearl, 1988). For the Markov Blanket to be a solution to the variable selection problem, several conditions are required (Tsamardinos & Aliferis, 2003). Most local causal discovery algorithms suffer from a significant drawback: they fail to identify causal relations in "difficult" distributions, e.g., distributions where a variable has low or even non-existence pairwise association with the target, but exhibits a high multivariate association with it in the context of other variables.

The new method attempts to combine the advantages from kernel methods for variable selection and those designed for causal discovery. Currently, kernel-based methods and Bayesian network-based causal discovery methods are considered as two totally different classes of methods with no obvious way of combining them and leveraging their strengths. The new algorithm developed in this thesis research is evaluated on several simulated, "difficult" distributions where the new method is able to correctly identify the Markov Blanket with high sensitivity and specificity. Additionally, the new method is run on several real world data sets returning a small number of features from

which simple linear classification models can be constructed and assessed in terms of classification performance versus other classifiers with all or other subsets of variables selected. The resulting classification models are small compared to other variable selection methods (for the two data sets, the model consists of 2-3 features which can be visualized). On another data set where a Markov Blanket-based variable selection method performs poorly, the new method has improved performance suggesting the existence of a complex, multivariate relationship in the underlying domain.

Research Summary: *Combining the ideas of kernel-based and Markov Blanket-based variable selection methods, a new algorithm was designed that improves the efficacy of inducing causal relations in "difficult" distributions. With this method, it is possible to identify variables causally related to a target variable, even when they exhibit no univariate association but only a multivariate association.*

I.3 Learning Local Regions

Discovery of causal relationships is a widely researched and hotly debated subject in many fields including philosophy, sociology, psychology, economics, statistics and recently computer science. One common formalism to represent and learn causal models from observational data in biomedicine is the causal Bayesian Network (Andreassen *et al.*, 1989; Beinlich *et al.*, 1989; Cowell *et al.*, 1999; Heckerman *et al.*, 1992; Lucas *et al.*, 1998; Andreassen *et al.*, 1999; Tong & Koller, 2001; Yoo & Cooper, 2004; Friedman *et al.*, 2000; Hartemink *et al.*, 2002; Yoo *et al.*, 2002; Bay *et al.*, 2002; Sachs *et al.*, 2005; Woolf *et al.*, 2005). As presented in my Masters research, one of the best (in terms of quality of learning and time efficiency) algorithms for learning such models, the Max-Min Hill Climbing algorithm (*MMHC*) has been developed and published (Tsamardinos *et al.*, 2006b). However, this method (and all other similar methods) is still unable to scale up to the dimensionality of data sets frequently appearing in biomedicine.

One option has been proposed for handling large data sets in these domains. This approach is to learn only a part of the network, focusing the learning on a subgraph (region) of the network around a variable of interest, e.g., a gene or protein of interest (Tsamardinos *et al.*, 2003c; Peña *et al.*, 2005).

This is in contrast to the current approach of globally learning the complete set of causal relations among all observed quantities. *Learning a local region without inducing the complete network is not trivial using previous methods and approaches.* However, the work on *MMHC* has paved the way to allow for the design of local causal discovery techniques.

A new local causal discovery method is designed and studied in this thesis research, empirically and theoretically, for the focused learning of network regions from observational data. The new method is expected to require only a fraction of the time required to learn the full network (global approach) and so to be able to scale up to even larger data sets than what is currently possible. It is unknown however, whether learning a region in a myopic way severely affects the quality of learning, i.e., without simultaneously considering all parts of the network and how these interact. Thus, an additional question is raised whether local learning is possible without sacrificing the quality of learning, a question that is of general interest to all future local algorithms. In short, the evaluation shows that in general the method for learning the region locally is more time-efficient and also produces structures of equal or higher-quality.

Research Summary: *A global and local approach are designed to learn a Bayesian network region. The local method is expected to be computationally more efficient than the global method approach; interestingly, the comparison in quality of the approaches reveals the local approach learning structure of equal or higher-quality.*

I.4 Making Predictions Under Manipulation

The Causality Challenge required researchers to use and develop an arsenal of methods using real-world data to make predictions in the presence of manipulations (WCCI 2008 Causality Challenge, 2008). For the tasks presented, the training data comes from an unmanipulated or "natural" distribution. However, several test sets were evaluated; the first is an unmanipulated test set and comes from the same distribution, the second and third test sets had some of all of the variables manipulated. When a variable is manipulated it is in a sense disconnected from its causes and

consequently its predictive power may be affected. Methods should take into account the effect of the manipulations in developing predictive models.

The new method developed in this thesis research uses Causal Bayesian networks (CBN) as the formalism to model and induce causal relations and to make predictions about the effects of the manipulation of the variables. Rather than use all variables to form the predictive model, the focus was on identifying the Markov Blanket of the target node. In the case of the unmanipulated test data, the approach for this task was to identify a Markov Blanket of the target then learn a predictive model using only these variables. In the case of test data with known manipulations, a manipulated Markov Blanket is identified (the manipulated Markov Blanket is a subset of the unmanipulated Markov Blanket with the manipulated children and their corresponding spouses removed). For the third case of test data with unknown manipulations, in order to avoid including irrelevant or misleading variables, the prediction model is built using only the parents of the target.

Many algorithms exist to learn the Markov Blanket. These algorithms typically do not infer the orientation of the causal relations of the members of the set. For the manipulated data sets, the directionality of the causal relations is also needed to select the correct variables to build a predictive model. Consequently, several methods were combined to determine this directionality. The approach developed, combines methods for: (a) finding the Markov Blanket of the target even under some non-faithfulness conditions *using the methods developed in Chapters II and III*, (b) reducing the problems to a size manageable by subsequent algorithms *using the method developed in Chapter IV*, (c) identifying and orienting the network edges, (d) identifying causal edges (i.e., not confounded), and (e) selecting the causal Markov Blanket of the target in the manipulated distribution.

The results of this new method on this challenge are reported where the method performed best on one of the four tasks. Additionally, an extensive discussion is included addressing issues raised when adapting and combining these varied methods to address a real world problem.

Research Summary *The method developed was used to create a submission to the Causality Challenge. Overall, the research provides theoretical results on the approach to the problem and also*

employs a suite of contemporary algorithms (including the other methods described in this thesis) to the real-world problems.

I.5 Summary

The thesis is constructed with chapters from the papers that have been submitted (or are in preparation) on this research (note, the Chapters II-V are written in the voice (we) appropriate for publication with advisor co-authors). Chapter II describes the algorithm to identify the top-weighted features of a Support Vector Machine model and its evaluation. Chapter III introduces the variable selection method, Feature Space Markov Blanket, that makes use of the previous method to perform variable selection in several difficult distributions. In Chapter IV, the focus is on the local approach of Bayesian network learning of regions. Chapter V presents the approach used for the submission to the Causality Challenge. Within each of these chapters, the method will be described along with any necessary theoretical concepts, definitions, and related work. Please note that because the methods developed fit into several different research communities, the notation is consistent with the standard in that community and may vary between chapters. However, within each chapter the necessary notation and definitions are presented. Each chapter also presents self-contained results on the evaluation of each method and discussion of future work and analyses. A final Chapter VI summarizes overall conclusions and recommendations for future work based on this thesis research.

CHAPTER II

IDENTIFYING TOP-WEIGHTED FEATURES IN POLYNOMIAL SVM MODELS

Polynomial Support Vector Machine models of degree d are linear functions in a feature space of monomials of at most degree d. However, the actual representation is typically in the form of support vectors and Lagrange multipliers that is unsuitable for human understanding. An efficient, heuristic method for searching the feature space of a polynomial Support Vector Machine model for those features with the largest absolute weights is presented. The time complexity of this method is $\Theta(dms^2 + sdp)$, where m is the number of variables, d the degree of the kernel, s the number of support vectors, and p the number of features the algorithm is allowed to search. In contrast, the brute force approach of constructing all weights and then selecting the largest weights has complexity $\Theta(sd\binom{m+d}{d})$. The method is shown to be effective in identifying the top-weighted features on several simulated data sets, where the true weight vector is known. Additionally, the method is run on several high-dimensional, real world data sets where the features returned may be used to construct classifiers with classification performances similar to models built with all or subsets of variables returned by variable selection methods. This algorithm provides a new ability to understand, conceptualize, visualize, and communicate polynomial SVM models and has implications for feature constructions, dimensionality reduction, and variable selection.

II.1 Introduction

Support Vector Machine (SVM) and kernel methods in general have been proven very successful methods for standard binary classification problems. A polynomial SVM on a given learning task returns a model represented by the set $\{\langle a_i, y_i, \mathbf{x}_i\rangle\}$ and the scalar b (called the *offset*), where $\mathbf{x}_i$ denotes the ith training vector, $y_i \in \{-1, +1\}$ its class, and each $a_i \geq 0$ is a scalar *Lagrange multiplier*. The training vectors for which $a_i \neq 0$ are called support vectors. The model classifies an input vector $\mathbf{x} \in \mathbb{R}^m$ to a member of $\{-1, +1\}$ using a function of the form:

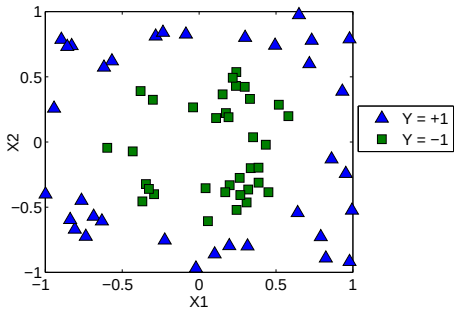
$$g(\mathbf{x}) = \text{sgn}\left(\sum_{k,a_k \neq 0} a_k y_k K(\mathbf{x}_k, \mathbf{x}) + b\right)$$

where $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$ is called the (full) polynomial Kernel of degree d. This representation of the model and the resulting $g(\mathbf{x})$ are efficient to compute but unsuitable for the human modeler to gain an understanding of the model's "logic". Specifically, it is difficult to determine from this representation which variables (components of $\mathbf{x}$) or combinations of these variables strongly affect the output of $g(\mathbf{x})$. Arguably, this is one reason why rule-learning classification, decision trees and other easier-to-interpret classifiers are sometimes preferred over the SVM classifier. The SVM model however, can also be represented as an affine function composed with a function Φ mapping $\mathbf{x} \in \mathbb{R}^m$ to a new space $\mathbb{R}^f$ called the feature space:

$$g(\mathbf{x}) = \text{sgn}\,(\mathbf{w} \cdot \Phi(\mathbf{x}) + b) = \text{sgn}\,(\sum_q w_q \Phi_q(\mathbf{x}) + b).$$

Here $\mathbf{w}$ is a weight vector in $\mathbb{R}^f$ and a multinomial notation is used to index the feature space. Feature space is then spanned by monomials of the form $\Phi_q(\mathbf{x}) = x_{i_1} \cdot \ldots \cdot x_{i_j} = x_1^{q_1} \cdots x_m^{q_m}$, $j \leq d$ and $\sum q_i = j$. Thus, the feature space contains all monomials of degree up to d. The former is called the *Kernel* representation and the latter the *explicit* representation of $g(\mathbf{x})$. The latter one allows easy interpretation and identification of the most important combinations of variables (called interaction terms in standard-statistical linear regression): these are the monomials with the largest absolute weight. This information can be used for human validation of the semantics of the model, visualization and for further feature selection.

The Kernel and the explicit formulations of the SVM model are juxtaposed in the following example problem. The ith component of the $\mathbf{x}$ vector is denoted with x_i and the ith component (variable) of the random vector X with capital X_i. The training data consist of 70 independent samples chosen from the uniform distribution on the square $\{-1, 1\} \times \{-1, 1\}$, where the class labels are given by the classification function, $Y = sgn(X_1^2 + X_2^2 - 0.5)$; see Figure II-1(b) for a portion of this data. Positive training examples are shown as (green) squares, with negative training examples



(a)

Training Data Set			
Sample	X_1	X_2	Target
1	0.319	-0.364	-1
2	0.241	0.536	-1
3	-0.380	0.390	-1
4	-0.368	-0.455	-1
⋮	⋮	⋮	⋮
70	0.791	0.596	1

(b)

Figure II-1: Example Problem Data: (a) Simulated training data over two variables X_1 and X_2 classified by $Y = sgn(X_1^2 + X_2^2 - 0.5)$. Positive (negative) training examples are denoted with triangles (squares). (b) A small portion of the training data set is presented.

	Support Vectors		Lagrange
	X_1	X_2	Multipliers
$\mathbf{x}_1$	−0.368	−0.455	6.273
$\mathbf{x}_2$	0.057	−0.607	4.884
$\mathbf{x}_3$	−0.566	0.620	−2.120
$\mathbf{x}_4$	0.649	−0.543	−14.031
$\mathbf{x}_5$	−0.085	0.825	−1.570
$\mathbf{x}_6$	−0.223	−0.753	−33.436

(a)

Decision Function	
Features	Weights
X_2^2	1.000
X_1^2	0.993
b	−0.522
X_1X_2	0.080
X_2	−0.018
X_1	−0.018

(b)

Figure II-2: Example Problem SVM Models: (a) The SVM model produced from the training data in Figure II-1 in Kernel representation consisting of 6 support vectors and corresponding Lagrange multipliers. (b) The model in explicit form. The weights (scaled by the maximum weight) and features of the decision function of the SVM model.

given as (blue) triangles; the data is presented in Figure II-1(a). An SVM model was trained on this data set using a degree 2 polynomial kernel. The six support vectors and Lagrange multipliers of the model are given in Figure II-2(a). The corresponding explicit representation of the SVM model is given in Figure II-2(b); the corresponding linear classification function is

$$g(\mathbf{x}) = sgn(-0.018X_1 - 0.018X_2 + 0.080X_1X_2 + 0.993X_1^2 + 1.000X_2^2 - 0.522).$$

which is a good approximation of the true classifying function. From this representation it is easy to determine that the dominating features are X_1^2, X_2^2.

Table II-1: Explosive Growth of Number of Features: The number of features in feature space is presented as a function of the number of variables and degree of the polynomial kernel. For the values greater than 10^{15}, the results are not exact and only serve as estimates.

Number of Variables	Degree 2	3	4	5	10
10	6.6×10^1	2.86×10^2	1.00×10^3	3.00×10^3	1.84×10^5
100	5.15×10^3	1.77×10^5	4.60×10^6	9.66×10^7	4.68×10^{13}
1000	5.01×10^5	1.68×10^8	4.21×10^{10}	8.46×10^{12}	2.91×10^{23}
10000	5.00×10^7	1.68×10^{11}	4.17×10^{14}	8.35×10^{17}	2.77×10^{33}
100000	5.00×10^9	1.67×10^{14}	4.17×10^{18}	8.33×10^{22}	2.76×10^{43}
1000000	5.00×10^{11}	1.67×10^{17}	4.17×10^{22}	8.33×10^{27}	2.76×10^{53}

Unfortunately, while potentially helpful, it is time-prohibitive to compute the explicit representation for large models. The number of features for m variables (components of the training vectors) and degree d of the kernel is $f = \binom{m+d}{d}$. Table II-1 depicts the growth of this function.

In this Chapter, the problem of converting the Kernel representation to an approximation of the explicit representation calculated by the r top-weighted features and their corresponding weights is studied. In other words, the efficient identification of the set Q_r of indexes of the r top-weighted features, $|Q_r| = r$ and $|w_i| \geq |w_j|$ for $i \in Q_r$ and $j \notin Q_r$ is sought. Two heuristic algorithms are presented and empirically compared that take as input a polynomial SVM model in Kernel representation, the number r of weights and features to return, and a parameter p, the number of features that are allowed to search in their effort to identify the top-weighted features. Sufficient conditions are provided for the heuristic algorithms to correctly return the top r weights. Even when the sufficient conditions fail, empirically the returned weights closely approximate the true set of r top-weighted features when only examining a very small portion of the feature space $p << \binom{m+d}{d}$. The best heuristic method has a time complexity in terms of multiplications of $\Theta(dms^2 + sdp)$, where m is the number of variables, d is the degree of the kernel, s is the number of support vectors, and p is the number of features to construct. In contrast, constructing all features and selecting the r largest ones has complexity $\Theta(sdf) = \Theta(sd\binom{m+d}{d}) \leq O(sdm^d)$. An empirical evaluation on real high-dimensional data sets is presented, showing that the algorithm can convert SVM models to human readable form, help interpret them and provide domain knowledge.

The overall approach in constructing a model and gaining understanding into the domain is to first build a potential high-degree polynomial SVM, then to identify the top-weighted features. In contrast, other typical approaches are to either (i) use classifiers that produce easy-to-understand models or (ii) to reduce the number of *variables* (in contrast to features) *before* building a model, i.e., couple classification with variable selection. While certainly feasible, these approaches often have the following disadvantages:

- Classifiers that produce comprehensive models (e.g., decision tress, rule-learning, linear regression) may be suboptimal for the learning task.
- Finding a variable selection method that adequately reduces the number of used variables without decreasing the performance may not be possible or may require time-consuming experimentation.

Particularly problematic are learning tasks where a group of variables has a high-multivariate association with the class Y, but every strict subset of the group has no association with Y. For example let Y be the parity function of X_1, X_2 and X_3. If all variables take values in $\{-1, +1\}$, then $Y = X_1 \cdot X_2 \cdot X_3$. *If all joint patterns of* $\{X_1, X_2, X_3\}$ *are equiprobable*, then no strict subset has any association with Y. This means that most variable selection methods that are based on some kind of pairwise association of each variable with Y will fail to identify any of $\{X_1, X_2, X_3\}$ as relevant. Such methods are selecting the top k variables ranked by any pairwise association with Y (e.g., Pearson correlation, χ^2, mutual information, etc.) and current Markov-Based approaches (e.g., HITON (Aliferis *et al.*, 2003a, 2009a,b), Koller-Sahami (Koller & Sahami, 1996)). Greedy wrapper methods such as forward variable selection will also fail as trying adding any of $\{X_1, X_2, X_3\}$ does not affect the performance of the model.

There is no greedy way to identify the high-order interaction term as relevant (i.e., by checking its subsets). Thus, only a method that simultaneously examines the three variables together can safely hone in on the feature. Notice that a polynomial SVM of degree each to the arity of the parity will implicitly construct this feature and thus be able to learn the correct classification function. However, the problem of identifying the specific interaction term remains. SVM-based variable

selection methods, such as the Recursive Feature Elimination algorithm, could possibly identify such features but, as shown in the empirical results, it fails more often than our proposed methods. Given the preceding discussion, the new method should be mostly suitable for problems with the following properties:

- the SVM model is sparse or nearly sparse in feature space, that is, the weight vector $\mathbf{w}$ has relatively few components with large magnitude,
- there are high-order interactions without correlated low-order interactions that make it difficult for current variable selection methods to identify relevant variables,
- and the problem is too large to allow a brute force calculation of $\mathbf{w}$.

Parity functions, where all input patterns are equiprobable, may seem fine-tuned, contrived problems. However, cases of high-order interactions without correlated low-interactions may be more common in nature than currently believed. This is because there are physical systems that *produce* such mechanisms, biological Evolution being one of them. For instance, consider a motivating example taken from (Scheines, 2009). Assume that gene C is up-regulated (with some noise) when either A or (inclusive) B is up-regulated. However, when A is up-regulated, B is down-regulated and vice-versa. The situation is graphically and qualitatively depicted in Figure II-3. The biological semantics are that C is up-regulated when A is; if A is not being expressed (and only then) a redundant mechanism (gene B) is activated to continue up-regulating C. Evolution produces such mechanisms for redundancy and optimization of resources. Notice that, neither A or B is associated with C: C is up-regulated even when A and B are not. This characteristic would make it hard for most variable selection methods to identify the AB interaction as important among tens of thousands of measured gene expressions. When looked at together however, A and B are highly-associated with C. The proposed algorithm is a first step in identifying such interactions in real biological data.

In recent years, many researchers have worked on the problem of variable selection with SVMs[1]. There are methods that rank the variables by scaling factors, where scaling factors are added into the kernel and are optimized in the training of the model (Weston *et al.*, 2000). The Recursive Feature

[1] Often called feature selection in the literature, but here variable selection (selecting a subset of the input variables) should be distinguished from feature selection (selecting a subset of the features).

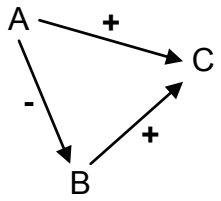


Figure II-3: Redundant Mechanism Example: A small 3 variable network example illustrating a redundant mechanism for activating variable C.

Elimination method (RFE) ranks each variable by removing each variable from consideration in turn to construct a score, removes the lowest ranked variables, and iterates through this process (Guyon *et al.*, 2002b). Recently, methods for constructing SVMs with sparse weight vectors have been developed (cf. l_0- and l_1 AROM (Weston *et al.*, 2003) and the methods of (Rakotomamonjy, 2003)). For the most part, these methods have been developed for linear SVMs. In (Weston *et al.*, 2003), the authors also describe minimizing the zero-norm with non-linear kernels. However, they note the difficulty in looking at the components of the resulting weight vector and only consider an exemplar problem of sufficiently small size that permits the exhaustive examination of all weights. The focus of the Chapter is not on a method to select variables or constructing sparse polynomial SVM models, but on determining the largest weights (and their features) in a model regardless of its origin. Note, the heuristic search algorithms performance is improved for SVMs with sparse weight vectors and so the experimental results for real data sets are expected to improve when using methods that produce such SVM models.

The Chapter is presented as follows, first in Section II.2 a short review of the necessary theory of Support Vector Machines is introduced. The properties of the polynomial kernel exploited for the new methods are presented in Section II.2.1. The next section contains a review of the explicit construction of the decision surface of a polynomial SVM through brute force calculations (Section II.3.1) followed by a description of the new heuristic methods for identifying the features with the largest magnitude weights. Section II.4 reveals the behavior of both the brute force calculations and the heuristic methods on several simulated and real data sets. How this new approach compares

to contemporary research on variable selection and feature extraction is described in a section on related work (Section II.4.2). Finally, a discussion reviewing the new approach including: limitations of the method, relationships to other approaches, and implications for future research.

II.2 Support Vector Machines

We briefly review the soft-margin Support Vector Machine (SVM) with a polynomial kernel for classification (Boser *et al.*, 1992; Vapnik, 1998). We assume a training data set, $\mathcal{D} = \{\mathbf{x}_k, y_k\},\ k = 1, \ldots, n$ consisting of sample vectors, $\mathbf{x}_k$ from the input data space $S \subset \mathbb{R}^m$, and associated class labels, $y_k \in \{-1, 1\}$. The SVM learning algorithm first maps each training vector $\mathbf{x}_k \in \mathbb{R}^m$ to a vector in feature space $\Phi(\mathbf{x}_k) \in \mathbb{R}^f$. The SVM algorithm learns from the training data a weight vector $\mathbf{w}$ in feature space and an offset b that define the hyperplane h in feature space:

$$h(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b. \tag{II.1}$$

A sample vector $\mathbf{x}$ is classified by the decision function $g(\mathbf{x}) = \operatorname{sgn}(h(\mathbf{x}))$. The parameters of the model (weights $\mathbf{w}$ and the bias (intercept) b) are learnt as the solution to the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{k=1}^{n} \xi_k^1 \tag{II.2}$$

subject to the constraints $y_k h(\mathbf{x}_k) \geq 1 - \xi_k$ and $\xi_k \geq 0$, for $k = 1, \ldots, n$. The solution hyperplane defines a linear decision surface that balances the margin of separation between the two classes (equal to $2/\|\mathbf{w}\|_2$) and the p-norm of the distances ξ_k of the data falling on the wrong side of the margin of separation. It is out of the scope of this work to discuss the intuition and the generalization properties of the SVM classifier.

For a class of mapping functions Φ (i.e., those that satisfy the Mercer theorem) there exists a kernel function K such that for $\mathbf{x}, \mathbf{x}' \in S$,

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') \tag{II.3}$$

and the minimization problem of Eq. II.2 is equivalent to the Wolfe dual formulation of the problem,

$$\min_{\mathbf{a}} \frac{1}{2} \sum_{k=1,l=1}^{n} a_k a_l y_k y_l K(\mathbf{x}_k, \mathbf{x}_l) - \sum_{k=1}^{n} a_k \tag{II.4}$$

subject to the constraints $\sum_{k=1}^{n} y_k a_k = 0$ and $C \geq a_k \geq 0$, for $k = 1, \ldots, n$. The components of the vector $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$ are called Lagrange multipliers, with each component corresponding to a constraint $y_k h(\mathbf{x}_k) \geq 1 - \xi_k$. The weight vector $\mathbf{w}$ is then given by

$$\mathbf{w} = \sum_{k=1}^{n} a_k y_k \Phi(\mathbf{x}_k) = \sum_{k, a_k \neq 0}^{s} a_k y_k \Phi(\mathbf{x}_k), \tag{II.5}$$

where the second summation is over the support vectors that is, the data samples with $a_k \neq 0$ (we let s denote the number of support vectors). The offset b is the solution to the equation $a_i \{ y_i \left(\sum_{k=1}^{n} a_k y_k K(\mathbf{x}_k, \mathbf{x}_i) \right) + b - 1 + \xi_i \} = 0$, for any i such that $0 < a_i < C$. From Eq. II.5, the function h can be rewritten to reflect this relationship,

$$h(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b = \sum_{k, a_k \neq 0}^{s} a_k y_k K(\mathbf{x}_k, \mathbf{x}) + b. \tag{II.6}$$

In many cases, and particularly for the polynomial kernel that we focus on in this work, the kernel function K is easily computable. Thus, by using the kernel function, K, and the dual formulation, the explicit mapping to feature space, Φ is never computed. As a result of the "kernel trick", a linear decision surface is implicitly constructed in a feature space that, for typical non-linear kernels, could be extremely high-dimensional. Consequently, typically for non-linear kernels the representation of an SVM model is in the Kernel form $\{\langle a_i, y_i, \mathbf{x}_i \rangle\}$ and the scalar b using the Lagrange multipliers and support vectors rather than the features and weights of the decision surface. While the classification function g becomes easy to compute even in high-dimensional feature spaces, the Kernel representation becomes a nonintuitive model that provides little information as regards to its internal workings.

II.2.1 Polynomial Kernel Properties

This work focuses solely on the full polynomial kernel, defined as $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$, where d is the degree of kernel. Kernel K is employed to implicitly map the training sample by a function Φ, to be defined, to a feature space. We also use the notation $H_d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$ to denote the homogenous polynomial kernel of degree d. The kernels are related via

$$K(\mathbf{x}_k, \mathbf{x}_j) = (\mathbf{x}_k \cdot \mathbf{x}_j + 1)^d = \sum_{l=0}^{d} \binom{d}{l} \cdot (\mathbf{x}_k \cdot \mathbf{x}_j)^l = \sum_{l=0}^{d} \binom{d}{l} H_l(\mathbf{x}_k, \mathbf{x}_j) \tag{II.7}$$

Letting $x_0 = 1$, the polynomial kernel can be written in the form,

$$K(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{m} x_i x_i' + 1 \right)^d = \left(\sum_{i=0}^{m} x_i x_i' \right)^d \tag{II.8}$$

Now, we observe that

$$K(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=0}^{m} x_i x_i' \right)^d = \sum_{i_1 = \ldots = i_d = 0}^{m} x_{i_1} \ldots x_{i_d} x'_{i_1} \ldots x'_{i_d} \tag{II.9}$$

Each monomial $x_{i_1} \ldots x_{i_d}$ chooses exactly d variables with replacement from vector $\mathbf{x}$ (including the dummy variable $x_0 = 1$). Instead of using the vector $\langle i_1, \ldots, i_d \rangle$ to index the monomial, we can use a vector $\mathbf{q} = \langle q_1, \ldots, q_m \rangle$ with the exponents to raise each variable $x_{i_1} \ldots x_{i_d} = x_0^{q_0} \ldots x_m^{q_m}$, with $q_i \geq 0$, $||\mathbf{q}||_1 \leq d$ and $q_0 = d - ||\mathbf{q}||_1$. For example, for $m = 2$ and $d = 3$, the factor $x_0 x_0 x_2$ corresponds to $\mathbf{q} = \langle 0, 1 \rangle$ ($q_0 = 2$). Notice that, a monomial, e.g., $x_0 x_0 x_2$ can occur multiple times in the summation of Eq. II.9 and in fact, it can be shown it occurs exactly $c_{\mathbf{q}}^2 = \binom{d}{\mathbf{q}} = \frac{d!}{q_0! \cdots q_m!}$ times for the corresponding exponent vector $\mathbf{q}$. Let us also use the multinomial notation $\mathbf{x}^{\mathbf{q}} = x_1^{q_1} \cdots x_m^{q_m}$. The set of all possible exponent vectors $\mathbf{q}$ corresponding to the monomials that appear in the summation of Eq. II.9 becomes $Q_{m,d} = \{\mathbf{q} = \langle q_1, ..., q_m \rangle \mid q_i \geq 0 \text{ for } i = 1, ..., m, ||\mathbf{q}||_1 \leq d\}$. Given the above,

we can rewrite Eq. II.9 as

$$K(\mathbf{x},\mathbf{x}') = \sum_{\mathbf{q}\in Q_{m,d}} \binom{d}{\mathbf{q}} x^{\mathbf{q}} x'^{\mathbf{q}} \tag{II.10}$$

$$= \sum_{\mathbf{q}\in Q_{m,d}} c_{\mathbf{q}}\mathbf{x}\cdot c_{\mathbf{q}}\mathbf{x}' \tag{II.11}$$

$$= \sum_{\mathbf{q}\in Q_{m,d}} \Phi_{\mathbf{q}}(\mathbf{x})\Phi_{\mathbf{q}}(\mathbf{x}') \tag{II.12}$$

by defining

$$\Phi_{\mathbf{q}}(\mathbf{x}) = c_q\mathbf{x}^{\mathbf{q}} = \sqrt{\binom{d}{\mathbf{q}}}\prod_{v=1}^{m} x_v^{q_v} \tag{II.13}$$

we obtain that

$$K(\mathbf{x},\mathbf{x}') = \Phi(\mathbf{x})\Phi(\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}'+1)^d$$

for the feature space defined by Φ that includes all monomials of degree exactly equal to d when the dummy variable $x_0 = 1$ is included, or equivalently, to the space of all monomials of degree less or equal to d of the original variables. Function Φ is not the unique function for which $K(\mathbf{x},\mathbf{x}') = \Phi(\mathbf{x})\Phi(\mathbf{x}')$; however, for all learning and classification purposes any function for which the previous equation holds is equivalent. For the purposes of this research, we will concentrate on identifying the features in the feature space defined Eq. II.13. The exact order of the variables in the vector $\Phi(\mathbf{x})$ is not important. Consider an example: a data set consisting of 2 variables and a polynomial kernel with a degree of 2 results in 6 features. A vector in the data space, $\mathbf{x} = \langle x_1, x_2\rangle$ maps to the following features

$$\Phi(\mathbf{x}) = \begin{pmatrix} c_{0,0}\ x_1^0x_2^0 \\ c_{1,0}\ x_1^1x_2^0 \\ c_{0,1}\ x_1^0x_2^1 \\ c_{1,1}\ x_1^1x_2^1 \\ c_{2,0}\ x_1^2x_2^0 \\ c_{0,2}\ x_1^0x_2^2 \end{pmatrix} = \begin{pmatrix} 1 & x_1^0x_2^0 \\ \sqrt{2} & x_1^1x_2^0 \\ \sqrt{2} & x_1^0x_2^1 \\ \sqrt{2} & x_1^1x_2^1 \\ 1 & x_1^2x_2^0 \\ 1 & x_1^0x_2^2 \end{pmatrix} \tag{II.14}$$

We now develop scores to assess the importance of a specific variable to classifying using Eq. II.1. A variable participates in more than one features (in the above example x_1 participates, i.e., $q_1 > 0$, in three features: $\sqrt{2}x_1, \sqrt{2}x_1x_2, x_1^2$) and thus is associated with a set of weights corresponding to these features. Specifically, we show how to efficiently calculate $||\mathbf{w}_l^v||$ defined as the norm of the weights corresponding to monomials of exactly degree l containing variables v.

Consider the following quantities to be calculated from the weight vector. For $v = 1, ..., m$ and $\mathbf{x} \in S$ and $\mathbf{z} \in \mathbb{R}^f$, let $\mathbf{x}^{\setminus v}$ denote the vector $\mathbf{x}$ with its vth component set to zero and $\mathbf{z}^{\setminus v}$ be the vector obtained from $\mathbf{z}$ by setting all components of $\mathbf{z}$ involving the variable v to zero, i.e., the qth component of $\mathbf{z}^{\setminus v}$ is:

$$z_{\mathbf{q}}^{\setminus v} = \begin{cases} 0 & \text{if } q_v \neq 0 \\ z_{\mathbf{q}} & \text{if } q_v = 0 \end{cases} \tag{II.15}$$

It then follows easily that $\Phi(\mathbf{x})^{\setminus v} = \Phi(\mathbf{x}^{\setminus v})$. Recall from Eq. II.5, the weight vector $\mathbf{w}$ can be written in the form $\mathbf{w} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k)$, where $\alpha_k = a_k y_k$. Then, $\mathbf{w}^{\setminus v}$ can be written as

$$\mathbf{w}^{\setminus v} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k)^{\setminus v} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k^{\setminus v}) \tag{II.16}$$

The sum of the squares of the weights of the features that do not contain variable v, that is the norm of $w^{\setminus v}$ is then,

$$\|\mathbf{w}^{\setminus v}\|^2 = \mathbf{w}^{\setminus v} \cdot \mathbf{w}^{\setminus v} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k)^{\setminus v} \cdot \sum_{j=1}^{n} \alpha_j \Phi(\mathbf{x}_j)^{\setminus v} \tag{II.17}$$

$$= \sum_{j,k=1}^{n} \alpha_k \alpha_j \Phi(\mathbf{x}_k^{\setminus v}) \cdot \Phi(\mathbf{x}_j^{\setminus v}) \tag{II.18}$$

$$= \sum_{j,k=1}^{n} \alpha_k \alpha_j (\mathbf{x}_k^{\setminus v} \cdot \mathbf{x}_j^{\setminus v} + 1)^d \tag{II.19}$$

Hence, the quantity $\|\mathbf{w}^{\setminus v}\|^2$ can be computed efficiently. We next consider the weight vector $\mathbf{w}^v$ of features that *do contain* variable v and is defined as $\mathbf{w}^v = \mathbf{w} - \mathbf{w}^{\setminus v}$. Since $\mathbf{w}^v$ and $\mathbf{w}^{\setminus v}$ vanish on complimentary subsets of features space, we have $\mathbf{w}^v \cdot \mathbf{w}^{\setminus v} = 0$ and hence by the Pythagorean

theorem, we have

$$\|\mathbf{w}^v\|^2 = \|\mathbf{w}\|^2 - \|\mathbf{w}^{\setminus v}\|^2 \tag{II.20}$$

The weight vector may be further decomposed by the degree of the features considered. Recall from Eq. II.7, the polynomial kernel is $K(\mathbf{x}_k, \mathbf{x}_j) = \sum_{l=0}^{d} \binom{d}{l} H_l(\mathbf{x}_k, \mathbf{x}_j)$. Consider a vector $\mathbf{z}$ in $\mathbb{R}^f$, let $\mathbf{z}_l$ be the vector obtained from $\mathbf{z}$ by setting all components of $\mathbf{z}$ whose product of variables do not have degree exactly l to zero, i.e., the qth component of $\mathbf{z}_l$ is,

$$z_{l,\mathbf{q}} = \begin{cases} 0 & \text{if } \sum_v q_v \neq l \\ z_{\mathbf{q}} & \text{if } \sum_v q_v = l \end{cases} \tag{II.21}$$

Similarly, $\mathbf{w}_l$ denotes the weights corresponding to all the features (products of original variables) with degree exactly l and the norm of this vector is

$$\|\mathbf{w}_l\|^2 = \sum_{k,j=1}^{n} \alpha_k \alpha_j \cdot \binom{d}{l} H_l(\mathbf{x}_k, \mathbf{x}_j) \tag{II.22}$$

$$= \binom{d}{l} \sum_{k,j=1}^{n} \alpha_k \alpha_j \cdot H_l(\mathbf{x}_k, \mathbf{x}_j) \tag{II.23}$$

Arranging features by degree forms a partition of feature space, we have

$$\|\mathbf{w}\|^2 = \|\mathbf{w}_0\|^2 + \cdots + \|\mathbf{w}_d\|^2 \tag{II.24}$$

Equations II.19 and II.23 can be combined in the following formula; the norm of the weights of the features of a specific degree l that do not contain variable v, may be written as

$$\|\mathbf{w}_l^{\setminus v}\|^2 = \binom{d}{l} \sum_{k,j=1}^{n} \alpha_k \alpha_j H_l(\mathbf{x}_j^{\setminus v}, \mathbf{x}_k^{\setminus v}) \tag{II.25}$$

Finally, the norm of the weights of the features of a specific degree l that *do contain* variable v is given by

$$\|\mathbf{w}_l^v\|^2 = \|\mathbf{w}_l\|^2 - \|\mathbf{w}_l^{\setminus v}\|^2 \tag{II.26}$$

The number of weights being summed into the norm quantities grows exponentially as the number

Table II-2: Number of weights involved in sums use the following base quantities: number of variables m, degree of the kernel d, degree level of interest l, and the variable of interest v.s

	Number of terms in Sum
$\|\mathbf{w}^v\|$	$\binom{m+(d-1)}{d-1}$
$\|\mathbf{w}_l\|$	$\binom{m+(l-1)}{l}$
$\|\mathbf{w}_l^v\|$	$\binom{m+(l-2)}{l-1}$

of variables grow (similar to how the number of features grow). Specifically, the norm of the weights of the features that contain variable v, $\|\mathbf{w}^v\|$, is a sum of $\binom{m+(d-1)}{d-1}$ features. The number of features in the sum of norm of the weights of the features of degree l is $\binom{m+(l-1)}{l}$ where $l \leq d$. The number of features in the sum of the weights of features of a degree l that contain variable v is $\binom{m+(l-2)}{l-1}$ features. This information is summarized in Table II-2.

Methods to Partition the Features and Weights

The previous definitions in this section have focused on examining subsets of the weight vector partitioning on whether a single variable is present or absent. It will be useful to also think of sets of features (and their corresponding weights) defined over subsets of the variables. For a given subset of variables $V \subset \{1, ..., m\}$, we define the following subsets of features:

$$F^{\setminus V} = \{ \mathbf{q} \mid \mathbf{q}_v = 0 \text{ if } v \in V\} \tag{II.27}$$

$$F^{V} = \{ \mathbf{q} \mid \mathbf{q}_v \neq 0 \text{ for some } v \in V\} \tag{II.28}$$

$$F^{\overline{V}} = F^{\setminus V^c} = \{\mathbf{q} \mid \mathbf{q}_v \neq 0 \text{ for } v \in V, \mathbf{q}_v = 0 \text{ for } v \notin V\}. \tag{II.29}$$

That is, $F^{\setminus V}$ is the set of features which do not contain any variable $v \in V$. The set of features F^V includes all features where some variable $v \in V$ is in each feature. Finally, $F^{\overline{V}}$ is the set of features consisting solely of variables $v \in V$.

To construct the calculations for the norm of the weight vector over these sets we will revisit the notation of the previous section updating it to consider sets of variables rather than a single

variable. For $v = 1, \ldots, m$, $\mathbf{x} \in S$, and $\mathbf{z} \in \mathbb{R}^f$, let $\mathbf{x}^{\setminus V}$ denote the vector,

$$\mathbf{x}^{\setminus v} = \begin{cases} 0 & \text{if } v \in V, \\ x^v & \text{otherwise.} \end{cases} \tag{II.30}$$

and let $\mathbf{z}^{\setminus V}$ denote the vector,

$$z_{\mathbf{q}}^{\setminus V} = \begin{cases} 0 & \text{if } \; q_v \neq 0 \text{ for some } v \in V \\ z_{\mathbf{q}} & \text{if } \; q_v = 0 \end{cases} \tag{II.31}$$

That is, $\mathbf{x}^{\setminus V}$ is the vector obtained from $\mathbf{x}$ by setting the each of the components in V to zero and $\mathbf{z}^{\setminus V}$ is the vector obtained from $\mathbf{z}$ by setting all each of the components involving variables $v \in V$ to zero.

Alternatively, $z_{\mathbf{q}}^{\setminus V}$ is zero when $q \in F^V$ and equal to $z_{\mathbf{q}}$ otherwise. It follows that $\Phi(\mathbf{x})^{\setminus V} = \Phi(\mathbf{x}^{\setminus V})$. Then, $\mathbf{w}^{\setminus V}$ can be written as

$$\mathbf{w}^{\setminus V} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k)^{\setminus V} = \sum_{k=1}^{n} \alpha_k \Phi(\mathbf{x}_k^{\setminus V}). \tag{II.32}$$

The sum of the squares of the weights of the features that do not contain variables $v \in V$, $F^{\setminus V}$ is then,

$$\|\mathbf{w}^{\setminus V}\|^2 = \sum_{j,k=1}^{n} \alpha_k \alpha_j (\mathbf{x}_k^{\setminus V} \cdot \mathbf{x}_j^{\setminus V} + 1)^d. \tag{II.33}$$

From this the other sum of the squares of the weights over the different feature sets may be calculated, namely,

$$\|\mathbf{w}^V\|^2 = \|\mathbf{w}\|^2 - \|\mathbf{w}^{\setminus V}\|^2. \tag{II.34}$$

For instance if $V = \{v_i, v_j, v_k\}$ then the sum of the squares of the weights of the feature that contain a variable $v \in V$ is $\|\mathbf{w}^V\|^2 = \|\mathbf{w}^{\{v_i, v_j, v_k\}}\|^2$. The norm of the weight vector over a set of features consisting solely of variables $v \in V$, $F^{\overline{V}}$ can also be defined. For example, consider the set of features $F^{\overline{V}}$ that *contain both variable u and v*, $F^{\overline{\{u,v\}}}$. The norm of the weight vector for these features is

denoted as $\mathbf{w}^{\overline{\{u,v\}}}$ through the following equation

$$\|\mathbf{w}^{\{u,v\}}\|^2 = \|\mathbf{w}^u\|^2 + \|\mathbf{w}^v\|^2 - \|\mathbf{w}^{\overline{\{u,v\}}}\|^2. \tag{II.35}$$

This equation can be generalized to sets of variables V of all sizes. Or alternatively, from Eq. II.29, the norm of the weights over the features $F^{\overline{V}}$ is equivalent to the norm over the features $F^{\setminus V^C}$.

II.2.2 Efficient Computation of the Weight Norms

In the algorithms to follow, we compute $\|\mathbf{w}_l^v\|$ for all variables v and all degrees l. We now show how to perform this computation efficiently. Let us denote with $X - [s \times s]$ the matrix with support vectors as rows, and $\mathbf{a}$ the row vector of α_i's. As defined in Eq. II.5, $\|\mathbf{w}\| = \sum_k^s \alpha_k \Phi(\mathbf{x}_k) \sum_j^s \alpha_j \Phi(\mathbf{x}_j) = \sum_{k,j} \alpha_k K(\mathbf{x}_k, \mathbf{x}_j) \alpha_j = \sum_{k,j} \alpha_k (\mathbf{x}_k \cdot \mathbf{x}_j + 1)^d \alpha_j$ from which we obtain

$$\|\mathbf{w}\|^2 = \mathbf{a}(XX^T + \mathbf{1})^{\cdot d}\mathbf{a}^T \tag{II.36}$$

where we use $X^{\cdot d}$ to denote element-wise exponentiation and $\mathbf{1}$ denotes a matrix of the same size as X with elements all 1. The computation requires $\Theta(s^2(m+d))$ multiplications and additions, and for $d < m$ this becomes $\Theta(s^2 m)$. Similarly, by using Eq. II.23 we obtain:

$$\|\mathbf{w}_l\|^2 = \mathbf{a}(XX^T)^{\cdot l}\mathbf{a}^T \tag{II.37}$$

Now, regarding $\|\mathbf{w}_l^{\setminus v}\|^2$ we can write this computation using Eq. II.25 as $\binom{d}{l} \sum_{k,j=1}^n \alpha_k \alpha_j (\mathbf{x}_j^{\setminus v} \cdot \mathbf{x}_k^{\setminus v})^l$ $= \binom{d}{l} \sum_{k,j=1}^n \alpha_k \alpha_j (\mathbf{x}_j \mathbf{x}_k - x_{k,v} x_{j,v})^l$. Thus, before exponentiating, we remove from each vector product $\mathbf{x}_j \mathbf{x}_k$ the factor $x_{k,v} x_{j,v}$. By defining the row vector X_v as the values of the support vectors of the v variable, we can write the above in matrix format:

$$\|\mathbf{w}_l^{\setminus v}\|^2 = \binom{d}{l} \mathbf{a}(XX^T - X_v^T X_v)^{\cdot l}\mathbf{a}^T \tag{II.38}$$

we finally obtain

$$\|\mathbf{w}_l^v\|^2 = \binom{d}{l}\mathbf{a}(XX^T)^{\cdot l}\mathbf{a}^T - \binom{d}{l}\mathbf{a}(XX^T - X_v^T X_v)^{\cdot l}\mathbf{a}^T \tag{II.39}$$

Once again the time-complexity of this computation is $\Theta(s^2 m)$. Calculating $\|\mathbf{w}_l^v\|^2$ for all variables at all levels without any caching of intermediate results would incur a cost of $\Theta(s^2 m^2 d)$, which is prohibitive for high-dimensional problems. Let us assume however, that we compute XX^T and $X_v^T X_v$ for each variable and cache the results. Both of these operations require a cost of $s^2 m$ respectively. Now, once we have computed $\|\mathbf{w}_1^v\|$ and cached the intermediate results, we can compute the first term of $\|\mathbf{w}_2^v\|$ with s^2 multiplications: $(XX^T) * (XX^T)$, where $*$ is the element-wise multiplication. Similarly, the second term of the equation requires another s^2 multiplications. Thus, in total it requires $\Theta(s^2 md)$ to calculate all $\|\mathbf{w}_l^v\|^2$.

II.3 Identifying The Top-Weighted Features

In this section, we present the brute force and two heuristic methods for identifying the weights, i.e. components of a weight vector $\mathbf{w}$, that are largest in magnitude. The heuristic approaches introduced here avoid the expensive explicit construction of the entire weight vector, by using condensed information of the weights. Specifically, the heuristic methods conduct a search for the top weights of the SVM model, guided by the norm of the weights summed over various subsets of features described in Section II.2.1. All methods are currently implemented in Matlab. Because complex data structures are not commonly used or efficiently implemented in Matlab, the methods' complexity is analyzed in its current implementation and although other better-suited data structures are discussed. Briefly, the three methods presented are:

- *Brute Force*: Explicitly calculate the weight vector $\mathbf{w}$ and identify the largest in magnitude components (feature weights).
- *Heur1*: Rank variables according to the sum of weights in all features they participate in, i.e., rank variable v according to $\|\mathbf{w}^v\|^2$. Select the top k variables and explicitly calculate all weights of the features that can be constructed with these variables only.

- *Heur2*: Calculate $s(v,l) = \|\mathbf{w}_l^v\|^2$ for each variable and degree $l \leq d$. These are the sums of weights of features involving a particular variable of specific degree l. Using this information, select a level l' and a set of variables H and construct all possible features and their weights involving only variables in H at level l'. The identified weights are removed from the corresponding sums of weights $s(v,l')$ for each $v \in H$. Thus, at this point $s(v,l)$ contain the sum of *remaining-to-identify* weights involving v of degree l. The process is repeated until enough weights have been explicitly constructed.

II.3.1 Exhaustive Search

To identify each feature Φ_q and its corresponding weight w_q we use Eq. II.20 and II.13:

$$\mathbf{w}_q = \sum_k^s \alpha_k \Phi_q(\mathbf{x}_k) = \sum_k^s \alpha_k \sqrt{\binom{d}{\mathbf{q}}} \mathbf{x}_k^{\mathbf{q}} \tag{II.40}$$

Calculating $\mathbf{x}_k^{\mathbf{q}}$ requires at most d multiplications (there are at most d non-zero exponents in $\mathbf{q}$); calculating $\mathbf{w}_q$ requires another s additions, where s is the number of support vectors. We consider the calculation of the constants $c_{\mathbf{q}} = \sqrt{\binom{d}{\mathbf{q}}}$ of constant complexity: for low d there is a small number of different values of these constants (the number of different $c_{\mathbf{q}}$'s is the number of possible ways to obtain the sum of d by summing integers from 0 to d). Thus, to identify all features the time-complexity is $\Theta(sdf)$, where f the total number of features, equal to:

$$f = \binom{m+d}{d}. \tag{II.41}$$

This growth in the number of features is presented in Table II-1. *The explosive growth does not allow all features (and weights) to be calculated for large data sets and* $d > 1$. Sorting and obtaining the largest r weights (and corresponding features) requires another $\Theta(f \log f)$ time. Notice that $\Theta(f) = \Theta(\binom{m+d}{d}) \subset O(m^d)$ and so, the complexity order of the method becomes $\Theta(sdf + f \log f) \subset O(sdm^d)$.

This brute force method is currently implemented in Matlab to maximize speed in the mapping

to feature space. Namely a quick but memory inefficient method is used to determine $Q_{m,d} = \{\mathbf{q} = \langle q_1, ..., q_m \rangle\}$ the exponents for all features. It is a memory limitation in calculating this matrix that limits the brute force calculation of the weight vector (using an ordinary PC the scope of the problems is limited to $\sim>$ 700 variables with a degree 2 kernel, and $\sim>$ 50 variables with a degree 4 kernel). However, alternative methods for constructing all features are also possible using iterations of for-loops to consider each feature's weight to be calculated. This method has also been coded into Matlab, however the for-loop construction of this method does not take advantage of Matlab's matrix operations and is therefore limited in its use by the speed of the method (for example, on a degree 2 problem with 200 variables to construct the 20,301 elements of the weight vector takes over 45 seconds). An implementation in C could be faster in execution time yet is still limited by the overall complexity requirements of the method.

Even these straight-forward approaches will eventually also run into memory limitations of keeping the data, weight vector, and Q matrix of corresponding feature in memory (the size of the weight vector alone for a 10,000 variable, degree 3 problem is over 150GB). Because we are looking at identifying the top weights, the brute force method may be implemented such that only the top r weights are kept and returned (similar to the heuristic methods). Therefore, rather than just place each weight in turn in a vector as it is created, a sorted list of the top weights will be constructed and maintained. This memory-saving implementation will add complexity to the algorithm to keep the sorted list no matter what data structures used.

II.3.2 Selection of Top Ranked Variables Then Exhaustive Search

The first heuristic method is referred to as `Heur1`. The idea of this method is to perform variable selection using the "naive" variable ranking score of non-linear RFE, that is using only the first iteration of RFE (Guyon *et al.*, 2002b), then the brute force calculation of features and weights for the selected variables. The variables are ranked according to the sum of the weights of all features they participate in, that is rank each variable v according to $\|\mathbf{w}^v\|$. The top k variables are selected and all features and weights among those variables are explicitly constructed.

This method is presented in Algorithm 1 where the inputs are the support vectors **SV**, alpha values $\boldsymbol{\alpha}$ and two parameters p the number of features to construct and r the number of features to return. The parameter p determines how many features constructed and weights calculated and consequently determines the number of variables considered. The number of variables for which all features are constructed, k, is maximized so that,

$$k = \max_{i=1,\ldots,m} \binom{i+d}{d} \text{ such that } \binom{i+d}{d} \leq p. \tag{II.42}$$

For each variables, a score is calculated as the norm of the weight vector for all features involving that variable, $\|\mathbf{w}^v\|^2$ (line 3-5). The top k ranked variables are calculated from this score and stored in the subset V (line 6). All features involving solely variables in V are constructed ($F^{\overline{V}}$) and weights calculated (line 7-10). From these the top r weights ($\mathbf{w}^{\overline{V},r}$) are corresponding features ($F^{\overline{V},r}$) are identified and returned as output of the `Heur1` method.

Algorithm 1 IdentifyTopWeights-`Heur1` Method

1: **procedure** IDENTIFYTOPWEIGHTS-HEUR1 (**SV**, $\boldsymbol{\alpha}$, p, r)
Inputs: **SV** $-[s \times m]$, support vectors; $\alpha - [s \times 1]$, alpha values
p - number of features to construct; r - number of features to return
Output: $\mathbf{O} = \{\langle \mathbf{w}_{F_{V^r}}, F_{V^r} \rangle\}$, list of top r largest weights $\mathbf{w}_{F_{V^r}}$ and feature indices F_{V^r}

% Determine k, number of variables in feature construction
2: $k = max_{i=1,\ldots,m} \binom{i+d}{d}$ s.t. $\binom{i+d}{d} \leq p$

% Score each variable, Select V the top k ranked variables
3: **for** $i = 1, ..., m$ **do**
4: $\mathbf{s}[i] = \|\mathbf{w}^i\|^2$
5: **end for**
6: $V = \{i \mid s[i] \text{ in top } k\}$

% Construct all features among variables in V
7: $F^{\overline{V}} = \{\mathbf{q} \mid \mathbf{q}_v \geq 0 \text{ for any } v \in V \text{ and } \mathbf{q}_v = 0 \text{ for } v \notin V\}$
8: **for** $\mathbf{q} \in F^{\overline{V}}$ **do**
9: $\mathbf{w}_q = \sum_j \alpha_j \Phi_q(\mathbf{SV_j})$
10: **end for**
% Sort to identify top r weighted features
11: $\mathbf{w}_{F_{V^r}} = \{\mathbf{w}_q \mid \mathbf{q} \in F^{\overline{V}} \text{ and } \mathbf{w}_q \text{ in top } r \text{ weights}\}$
12: $F_{V^r} = \{\mathbf{q} \mid \mathbf{q} \in F^{\overline{V}} \text{ and } \mathbf{w}_q \text{ in top } r \text{ weights}\}$
13: **return** $\mathbf{O} = \{\langle \mathbf{w}_{F_{V^r}}, F_{V^r} \rangle\}$
14: **end procedure**

The overall time complexity of this method is $\Theta(dms^2 + sdp)$ multiplications. The complexity

is broken down as follows: the cost to calculate the score for each variable is $\Theta(dms^2)$ (where the complexity of this operation is discussed in Section II.2.2) and the cost to calculate the weights first mapping the support vectors to the p features identified $\Theta(sdp)$ then computing the weights $\Theta(sp)$. In addition to the cost of the multiplications, the vector of variable scores and weights are sorted at cost $O(m \log m)$ and $O(p \log p)$ for each item respectively; however, the main computational time of the method is dominated by the weight and score calculations.

II.3.3 Guided Search to Construct Top Features

The second heuristic method, referred to as `Heur2`, uses the norm of the weight vector decomposed by both variable and degree of the features to guide the search for the top-weighted features ($s(v, l) = \|\mathbf{w}_l^v\|^2$, Eq. II.26). This information is used to select a level l' and set of variables V for which all possible features (among variables V at that level) and their weights are calculated. The weights found are subtracted from corresponding sums $s(v, l')$ and the next level and variables are selected; repeating this process until a specified number of features are constructed.

If a variable v participates in only one feature at degree level l with a non-zero weight $w_{\mathbf{q}}$ then $\|\mathbf{w}_l^v\| = |w_{\mathbf{q}}|$. When a variable v participates in more than one feature at level l then $\|\mathbf{w}_l^v\| > |w_{\mathbf{q}}|$. In either case, the quantity $\|\mathbf{w}_l^v\|$ is an upper bound on the largest (in magnitude) weight for any feature involving variable v at level l. The search for the top weights uses these quantities as a guide to selectively calculate the weights of suspected top features.

This method takes the same inputs as `Heur1`: the support vectors of the SVM model $\mathbf{SV}$, the alpha values $\boldsymbol{\alpha}$, p - the number of features to construct, and r - the number of features to return. A pseudo-code description of the implementation is presented in Algorithm 2. The method begins with the construction of the $d \times m$ *contributions matrix* $\mathbf{s}$ where $\mathbf{s}[l][v] = \|\mathbf{w}_l^v\|^2$ for $l = 1, \ldots, d$ and $v = 1, \ldots, m$. This quantity is calculated via Equations II.26 and II.25, therefore,

$$\|\mathbf{w}_l^v\|_2^2 = \binom{d}{l} \sum_{k,j=1}^{n} \alpha_k \alpha_j \cdot H_l(\mathbf{x}_j, \mathbf{x}_k) - H_l(\mathbf{x}_j^{\setminus v}, \mathbf{x}_k^{\setminus v}). \tag{II.43}$$

After the calculation of the initial contributions matrix, $\mathbf{s}$, the heuristic search loops through the

following 3 sub-procedures: (1) select the next level and variable(s) to focus construction, (2) explicitly construct the features and calculate their weights for the selected variables and level, and (3) update the bounds of the contribution matrix. Once the search procedure constructs p features, the features are sorted by their absolute weight and the top r features are returned.

Algorithm 2 IdentifyTopWeights-`Heur2` Method

1: **procedure** IDENTIFYTOPWEIGHTS-`HEUR2` ($\mathbf{SV}$, α, p, r)
Inputs: $\mathbf{SV} - [s \times m]$, support vectors; $\alpha - [s \times 1]$, alpha values, p - number of features to construct; r - number of features to return
Output: $\mathbf{O} = \{\langle \mathbf{W^r}, \mathbf{F^r} \rangle\}$, list of r largest weights $\mathbf{W^r}$ and their feature indices $\mathbf{F^r}$

% Create contributions matrix, $\mathbf{s}$
2: **for** $l = 1, \ldots, d$ and $v = 1, \ldots, m$ **do**
3: $\mathbf{s}[l][v] = \|\mathbf{w}_l^v\|^2$
4: **end for**
5: $\mathbf{W} \leftarrow \emptyset$, $\mathbf{F} \leftarrow \emptyset$
6: **while** $|\mathbf{W}| \leq p$ **do**
% Select Expansion Level and Variables of Interest
7: $[l', V_{l'}, v'] =$ select-level-variables($\mathbf{s}, \mathbf{F}$)

% Construct all features among variables in $V_{l'}$ and v'
8: $F_{V_{l'}} = \{\mathbf{q} \mid \mathbf{q}_{v'} > 0,\ \mathbf{q}_v \geq 0$ for any $v \in V,\ \mathbf{q}_v = 0$ for $v \notin V$, and $\sum \mathbf{q} = l'\}$
9: **for** $\mathbf{q} \in F_{V_{l'}}$ **do**
10: $\mathbf{w}_q = \sum_j \alpha_j \Phi_q(\mathbf{SV_j})$
11: **end for**

% Update Bounds
12: **for each** $\mathbf{q} \in F_{V_{l'}}$ **do**
13: $\mathbf{s}[l'][v] = \mathbf{s}[l'][v] - w_q^2$, where $q_v > 0$
14: **end for**
15: $\mathbf{W} = \mathbf{W} \cup \mathbf{w}_{F_{V_{l'}}}$; $\mathbf{F} = \mathbf{F} \cup F_{V_{l'}}$
16: **end while**

% Sort to identify top r weighted features
17: $\mathbf{W^r} = \{\mathbf{w}_q \mid \mathbf{q} \in \mathbf{F}$ and $\mathbf{w}_q$ in top r weights$\}$
18: $\mathbf{F^r} = \{\mathbf{q} \mid \mathbf{q} \in \mathbf{F}$ and $\mathbf{w}_q$ in top r weights$\}$
19: **return** $\mathbf{O} = \{\langle \mathbf{W^r}, \mathbf{F^r} \rangle\}$
20: **end procedure**

The first sub-procedure (function select-level-variables, line 7, Algorithm 2) determines what are the next features to be constructed and weights calculated is described in Algorithm 3. First, the level on which to focus, l', is selected by one of two simple methods. The first simply selects the level of the contributions matrix with the maximum value. The second method normalizes the contributions matrix by the number of items in each cell ($\mathbf{N}[l][v]$), then selects the level with the maximum value.

Algorithm 3 IdentifyTopWeights Select Level and Variables Function

1: **function** SELECT-LEVEL-VARIABLES($\mathbf{s}, \mathbf{F}$)
% Select level on which to focus, l'
2: **for** $i = 1, \ldots, d$ **do**
3: $maxval[i] = \max(\mathbf{s}[i][:])$ or $maxval[i] = \max(\mathbf{s}[i][:]/\mathbf{N}[i][:])$
4: **end for**
5: $l' = \arg\max maxval[:]$

% Determine variables previously considered at level l', $V_{l'}$,
% and new variable to consider v'
6: $V_{l'} = \{i \mid \text{ any } \mathbf{q} \in \mathbf{F}, \mathbf{q}_i > 0\}$
7: $v' = \arg\max \mathbf{s}[l'][\{1, \ldots, m\} \setminus V_{l'}]$
8: **return** $l', V_{l'}, v'$
9: **end function**

The selection of a set of variables from which all features will be constructed uses only the contributions information at the selected level and also considers what features of this level have already been constructed. First, the variables involved in features at this level is determined, $V_{l'}$. Then, the variable with the top contributions matrix value at level l' that is not in $V_{l'}$ is selected v'.

The second sub-procedure explicitly constructs the features and calculates the weights for the selected level and variable sets found (line 8-11, Algorithm 2). The construction of new features includes all combinations with any variables already constructed at the selected level, $V_{l'}$ with the new selected variable v'. Initially, $V_{l'}$ is empty so only the feature consisting of variable v' at level l' is constructed, but as the algorithm continues combinations of variables are considered.

The final step involves updating the bounds on the top weight (lines 12-14, Algorithm 2), consequently the contributions matrix is updated by any weights explicitly calculated. For example, if the feature $X1X3^2$ is constructed and its weight, w_{X1X3^2}, is calculated, then the contribution matrix is updated: $\mathbf{s}[l][v] = \mathbf{s}[l][v] - (w_{X1X3^2})^2$ where $l = 3$, the degree of the feature constructed and $v = 1$ and 3 the variables in this feature.

The three sub-procedures reside in a loop that continues until the list of features (and their weights) has at least p items. Overall, the calculation of the top p weights via the heuristic method is $\Theta(dms^2 + sdp)$, with s support vectors, m variables, d the degree of the kernel, and p the number of features to construct.

The cost of creating the contributions matrix could be estimated as $\Theta(dm^2s^2)$, where for each cell

of the matrix of which there are $[d \times m]$, the calculations cost $\Theta(ms^2)$. However, by efficient storing of partial information this cost is reduced to $O(dms^2)$. For instance, the Gram matrix $SV * SV^T$ costing $O(ms^2)$ is calculated only once; all subsequent operations use the resulting matrix. Also, for a given variable the partial results from the previous level ($\mathbf{H}_{l-1}^{\setminus v}$) are stored and used in the calculation of the current level ($\mathbf{H}_{l}^{\setminus v}$). The cost of this select level and variables sub-procedure requires either no numeric calculations for determining the level or the $\Theta(dm)$ divisions for the version which does a normalization. In addition, the procedure requires finding the maximum value for each level or $\Theta(dm)$ look-ups.

The cost of constructing/calculating each new feature/weight is $\Theta(sd)$ multiplications. This procedure does not require the storage of all previously created features, rather just the variables at each level. A look-up just identifies the next variable that may be used for a given level. When developing this heuristic several implementations were considered that slightly altered the order of constructing new features and what information was stored between iterations. However, the final method described here was simple and efficient without sacrificing quality of the results.

The heuristic methods require initial calculations above and beyond the brute-force approach (the heuristic methods the contributions calculations are $\Theta(dms^2)$). Recall, the brute force calculation costs in total $\Theta(sdf)$, where f is the number of features. Therefore, for small problems, with $f < p + ms$, the brute force approach is expected to more efficient; see Section II.4 for confirmation of this proposition. In practice, the brute force calculation is first limited by the memory requirements for the calculation rather than the efficiency of the method (although more memory efficient implementations are possible, as discussed in Section II.3.1). In contrast, the heuristic method is quite scalable. The method has been run on data sets consisting of over 100,000 variables with a degree 2 kernel completing the calculations in less than 2 hours.

II.3.4 Sufficient Conditions for Heuristic Methods to Return Top r Weights

We next present sufficient conditions on $\mathbf{w}$ and p for the heuristic methods to return the top r weights. These sufficient conditions may also be used as stopping criteria for choosing p (note the maximally choice of p depends on the computational resources available to a user).

Lemma 1. Let $V \subset \{1, \ldots, m\}$ denote the set of k variables selected in `Heur1` and let $Q \subset F$ denote the set of r features returned by `Heur1`. Let $w_r = \min_{\mathbf{q} \in Q} |w_{\mathbf{q}}|$ If

$$w_r \geq \max_{i \in \{1,\ldots,m\} \setminus V} \|\mathbf{w}^i\| \tag{II.44}$$

then `Heur1` returns the top r weighted features, that is, $|w_{\mathbf{q}}| \geq |w_{\mathbf{q}'}|$ for all $\mathbf{q} \in Q$ and all $\mathbf{q}' \in F \setminus Q$.

Proof. Suppose II.44 holds. Let $w_r = \min_{\mathbf{q} in Q} |w_{\mathbf{q}}|$ and let $\tilde{Q}$ denote the features that only involve the variables in V. If $\mathbf{q}' \in \tilde{Q} \setminus Q$, then $|w_{\mathbf{q}'}| \leq w_r$ since Q consists of the r features from $\tilde{Q}$ with the largest magnitude weights. If $\mathbf{q}' \notin \tilde{Q}$ then $\mathbf{q}'_i \neq 0$ for some $i \notin V$ and so $w^2_{\mathbf{q}'} \leq \|\mathbf{w}^i\|^2 \leq w_r^2$ which completes the proof. □

Lemma 2. Let $Q \subset F$ denote the set of r features returned by `Heur2` and let $\mathbf{s}$ denote the final 'contributions' matrix. Let $w_r = \min_{\mathbf{q} \in Q} |w_{\mathbf{q}}|$. If

$$w_r \geq \max_{i,j} \mathbf{s}[i][j] \tag{II.45}$$

then `Heur2` returns the top r weighted features, that is, $|w_{\mathbf{q}}| \geq |w_{\mathbf{q}'}|$ for all $\mathbf{q} \in Q$ and all $\mathbf{q}' \in F \setminus Q$.

These sufficient conditions for stopping for both `Heur1` and `Heur2` methods may be loose bounds in practice. The parameter p will be used in the experimental evaluation to facilitate comparisons between the methods.

II.4 Experimental Results

The experimental evaluation compares the ability of the brute force and heuristic methods of identifying the largest weights of the SVM model. First, the evaluation is performed on several simulated data sets. These data sets have the advantage that the classification function is known

thus, providing insight into the ideal top weighted features. The results emphasize: (1) the ability of the heuristic methods to identify the features with the largest magnitude weights for a SVM model, (2) the heuristic method identifies the top features efficiently, and (3) the identified features provide insight into the functionality of the SVM model. Additionally, there will be focused attention on two simulated problems which serve as motivating examples to show where the use of this technique may be focused for difficult problems. Finally, the heuristic method that performs best is also then tested on several large real data sets. Here the results demonstrate the ability of the heuristic method to efficiently identify features with high predictive classification performance that may provide new or corroborate existing domain knowledge.

II.4.1 Simulated Data Results

We consider three simulated problems referred to as the Circle, Double-XOR, and Checkerboard problems. For the Circle problem, the classification is determined by the function $Y = sgn(x_1^2 + x_2^2 < 0.5)$. All components of $\mathbf{x}$ are independent samples from the uniform distribution on $[-1, 1]$. For the Double-XOR problem, the data was sampled from the Bayesian Network shown in Figure II-4. The variable T determines the classification of a sample. For the Checkerboard problem, classification is determined using the 3 relevant variables by which octant a data point resides (the class labels are determined by a parity function on the sign of the 3 relevant variable's coordinates). The data was sampled from a uniform distribution on $[-1, +1]$. For each problem, the number of input variable sizes was varied and is specified for each simulation. Data sets of 50, 100, 500, and 1000 training instances were sampled for each problem and input variable size. It is know in the large sample limit that these classifiers are sparse, that is all features involving variables not used to classify the data will have a weight that goes to zero (this statement is also observed in practice with the data sets) (Hardin *et al.*, 2004).

SVM Parameters

In general, a user does not know the optimal parameters to create an SVM model; for a polynomial SVM the degree of the kernel and C soft-margin parameter must be selected. In practice,

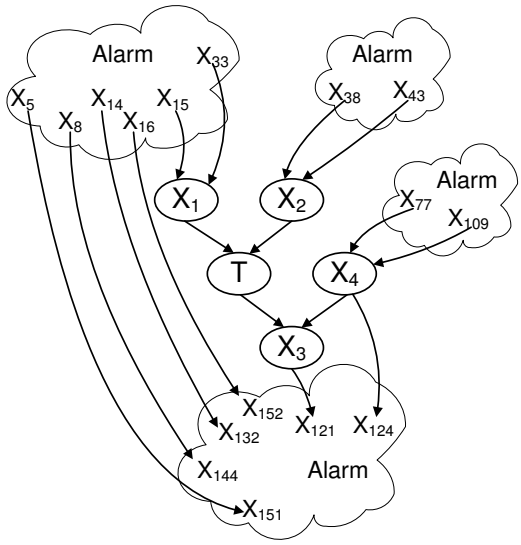


Figure II-4: Double-XOR Example Problem: The network from which data is sampled. The variables X_1 and X_2 are in an XOR relationship with T. Similarly, T and X_4 are in an XOR relationship with X_3. Tiled copies of the Alarm network form the other variables in the network.

classification performance is often used to choose these parameters, e.g., a nested cross-validation design is performed to optimize the selection of the SVM model parameters, with the expectation that the degree of the kernel will need to reflect the degree of the underlying classification function.

The focus of this work is not how best to optimize parameters for a SVM model, but on selecting the top features once an SVM model has been trained. For the rest of the simulated experimental section, the degree of the SVM kernel is selected to match the classification function and the C parameter is set to 10^3, i.e, the Circle and Double-XOR problem use a degree 2 kernel and the Checkerboard problem uses a degree 3 kernel.

Encoding of Binary Data

For problems consisting of binary data, the manner in which the binary data is encoded may have an effect on the learned SVM model. Consider the case of learning an XOR relationship of two variables with a degree 2 polynomial kernel. The features X_1, X_2, X_1^2, X_2^2, X_1X_2 and intercept b are available to the SVM model to assign weights in constructing the separating hyperplane. When a binary 0/1 encoding is used, one arithmetic expression that is equivalent to XOR to separate the data is $X_1 + X_2 - 2X_1X_2$. However, if the data was encoded as -1/+1, then the arithmetic expression

X_1	X_2	X_1^2	X_1^3	X_1X_2	$X_1^2X_2$
0	0	0	0	0	0
0	1	0	0	0	0
1	0	1	1	0	0
1	1	1	1	1	1

(a)

X_1	X_2	X_1^2	X_1^3	X_1X_2	$X_1^2X_2$	$X_1^3X_2$
-1	-1	+1	-1	+1	-1	+1
-1	+1	+1	-1	-1	+1	-1
+1	-1	+1	+1	-1	-1	-1
+1	+1	+1	+1	+1	+1	+1

(b)

Figure II-5: Redundancies of Binary Encodings: (a) Using the binary 0/1 encoding, all variables raised to a power have the same functionality as the single variable, e.g., $X_1^2 \equiv X_1$. (b) When the +1/-1 encoding is used, the all variables raised to an even power become constant and variables raised to an odd power are equivalent to the variable itself, e.g., $X_1^2 \equiv 1$ and $X_1^3 \equiv X_1$.

$-X_1X_2$ could be used to express the XOR relationship. The equation of the hyperplane is therefore affected by the encoding of the data, where one encoding may be preferred over another due to the simplicity of the learned function. However, when faced with a problem with a unknown underlying distribution it is impossible to select the best encoding a priori.

Regardless of the which of these two encodings are selected, when the input data to the SVM is binary the features of the SVM model are redundant in their representation. For example, when the input data is using a 0/1 encoding, any variable raised to a higher power is equivalent to the original variable. Figure II-5(a) shows several such equivalencies, e.g., $X_1 \equiv X_1^2$ and $X_1 \equiv X_1^3$. When the input data uses -1/+1 encoding, a different set of redundancies emerge. In this case, any variable raised to a even power becomes the constant 1 while any variable raised to an odd power is equivalent to the original variable. For example, $X_1^2 \equiv 1$ and $X_1^3 \equiv X_1$; see Figure II-5(b) for more examples.

In addition to the redundancy among the features, the two encodings may results in the features interpreted in different manners. For instance, consider the feature X_1X_2 in both encodings. Figure II-5(a) and (a) shows the behavior of this feature with respect to its base variables. With the 0/1 encoding the feature is reflecting an AND relationship among its constituent variables. When the -1/+1 encoding is considered, the feature has an exclusive OR relationship among its constituent variables. In either case, it is the interaction of the two variables that define the feature, but how the interaction occurs depends on the encoding.

The heuristic method is developed to be a general purpose procedure that does not differen-

tiate its behavior depending on the type of input data and its supplied encoding. Consequently, the redundancies of the binary data are removed after a feature list is returned by the methods. Specifically, the list of features is processed to remove redundant features. For the case of binary data, all possible features are expressed with exponents of either zero or one (i.e., $\mathbf{x}^{\mathbf{q}} = x_0^{q_0} \cdots x_m^{q_m}$ is restricted s.t. $q_i \in \{0, 1\}$). In order to return the specified number of features, r, the heuristic procedure may need to continue its search (the procedure may be designed as an anytime algorithm to achieve this goal).

Comparison of Heuristic Methods

The two heuristic methods described in Section II.3.2 and II.3.3 are compared in terms of the time efficiency and a quality metric indicating their ability to return the top-weighted features. The quality metric is defined as the norm of the r features returned by the method relative to the norm of the true top r features sorted from the complete weight vector. That is, if the method is asked to return 100 features, the norm of those 100 features is compared to the norm of the 100 top-weighted features found in the entire weight vector. The heuristic methods are compared on the simulated data sets.

The `Heur1` and `Heur2` methods are compared with results summarized in Table II-3. For this comparison, the problem size for each problem is as follows: 400, 500, 600, and 700 variables for Circle, 337, 448, 559, and 707 variables for Double-XOR, and 70, 80, 100, and 125 variables for the Checkerboard problem. The `Heur1` and `Heur2` method were both run to construct $p = \{50, 100, 500, 1000, 5000\}$ features. The time ratio of the two methods is calculated for the different vales of p, problems, problem sizes (number of variables), and sample sizes. For each problem, the time ratio was averaged over values of p and either problem size (top portion of the table) or sample size (bottom portion of the table). A time ratio of greater than one indicates `Heur1` taking longer than `Heur2` while a ratio of less than one indicates `Heur2` taking longer than `Heur1`. The `Heur1` method is in general faster than `Heur2`. This observation meets expectations because the `Heur1` method requires less overhead than `Heur2`.

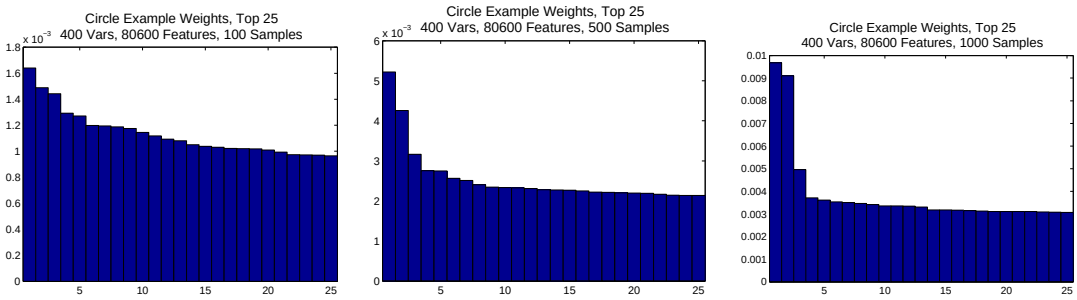
The results also reveal a general pattern that the speed gains of `Heur2` are affected by the number

Table II-3: Comparing `Heur1` and `Heur2`. The time and quality of the two heuristic methods, `Heur1` and `Heur2`, are compared. The time ratio of the two methods was measured for a number of runs with different values of p (number of features to construct), problems, problem sizes (number of variables), and sample sizes. For each problem, the time ratio was averaged over values of p and either problem size (top of the table) or sample size (bottom of the table). A time ratio of greater than one indicates `Heur1` being less efficient while a ratio of less than one indicates `Heur2` as the slower method. The quality ratio measured the ability of the two methods to return the top features. The quality measure is calculated for each instantiation over different values of p (number of features to construct), r (number of features to return), problems, problem sizes, and sample sizes. For each problem, the quality ratio was averaged over values of p, r, and either problem size (top of the table) or sample size (bottom of the table). A quality ratio of greater than one indicates `Heur1` returning the higher weighted features while a ratio of less than one indicates `Heur2` returning higher quality features.

	Time - `Heur1`/ `Heur2` Number of Samples				Quality - `Heur1`/ `Heur2` Number of Samples			
Data Set	50	100	500	1000	50	100	500	1000
Circle	0.620	0.753	0.960	0.975	0.812	0.778	0.796	0.794
Double-XOR	0.831	0.899	0.985	0.990	0.999	1.000	0.996	0.995
Checkerboard	0.782	0.870	0.991	0.995	0.746	0.717	0.718	0.813
	Number of Variables				Number of Variables			
Data Set	Increasing Num. Vars. $\longrightarrow$				Increasing Num. Vars. $\longrightarrow$			
Circle	0.844	0.834	0.823	0.807	0.838	0.801	0.777	0.763
Double-XOR	0.914	0.921	0.931	0.939	0.997	0.998	0.996	0.998
Checkerboard	0.905	0.907	0.915	0.910	0.843	0.755	0.715	0.682

of samples in the data. The sample size of course plays an important part in calculating the weights for both methods however, this calculation is the same for both methods. The sample size also influences the distribution of the components of the weight vector (see Figure II-6(a-c)). As sample size increases (across the row), the distribution of weights becomes more concentrated in a few important features. The change in the weight distribution can affect the entries of the contributions matrix. Figure II-6 illustrates the weight distribution (top 25 weights) for the Circle problem (the other problems show similar results and are presented in Appendix A.I). As the sample size grows the change in make-up of the contributions matrix will affect how new features are constructed in the search process and may explain this timing difference between the two heuristic methods.

In terms of the quality metric, the results are presented as a ratio of `Heur1` over `Heur2`, where a ratio of greater than one indicates `Heur1` increased ability to return the top-weighted features (less than one indicates `Heur2` is better at returning the top-weighted features). The quality ratio



(a) (b) (c)

Figure II-6: Distribution of Weights of Top 25 Features of the Circle example with increasing sample: (a) 100, (b) 500, and (c) 1000 samples.

was measured for the two heuristics for different values of numbers of features to construct $p = \{100, 500, 1000, 5000\}$, number of features to return $r = \{50, 100, 500, 1000\}$, problem, problem size, and sample size. For each problem also shown in Table II-3, the quality ratio was averaged over values for p, r (with $r < p$), and either problem size (top portion of table) or sample size (bottom portion of table). `Heur2` shows equal or better ability to return the top-weighted features in all cases (the method exhibits equal ability on the Double-XOR problem, and improved ability on the other two problems). Another trend to observe is that as the size of the problem gets larger the `Heur2` method improves over the `Heur1` method. There are tradeoffs between the two method, with no method showing dominance in both time and quality however, the difference in time is less significant than the potential ability to find the top-weighted features. Therefore of the two heuristic methods the `Heur2` will be used in future comparisons to the brute force approach and on real data sets.

Comparison of Brute Force to Heuristic Method

The brute force approach is compared to the `Heur2` heuristic method in terms of the execution time to complete each procedure (the execution time does not include the time to learn the SVM model - which is necessary and equal for both procedures). For this comparison, the problem size for each problem is as follows: 250, 350, and 450 variables for Circle, 226, 337, and 448 for Double-XOR, and 50, 70,and 80 variables for Checkerboard problems. The smaller data problem sizes were used again in order for the brute force to run and produce all features. The heuristic method was run

Table II-4: Comparing Brute Force and `Heur2`. The time and quality of the brute force and heuristic `Heur2` method are compared. The time ratio of the two methods was measured for a number of features to construct - p (all features for the case of brute force), problem, problem size (number of variables), and sample size. For each problem, the time ratio was averaged over values of p and either problem size (top portion of table) or sample size (bottom portion of table). The quality ratio measured the ability of the heuristic approach to return the top features for a each instantiation compared to the true top features over differing number of features to identify - p, number of features to return - r, problem, problem size, and sample size. For each problem, the quality ratio was averaged over values for p, r, and either problem size (top portion of table) or sample size (bottom portion of table).

	Time - Brute Force / `Heur2` Number of Samples				Quality - Brute Force / `Heur2` Number of Samples			
Data Set	50	100	500	1000	50	100	500	1000
Circle	396.0	144.8	6.9	2.1	1.02	1.03	1.08	1.04
Double-XOR	411.8	182.3	10.3	2.5	1.01	1.00	1.00	1.00
Checkerboard	1055.1	519.6	34.0	9.9	1.00	1.01	1.03	1.01
	Problem Size				Problem Size			
Data Set	Small	Mid	Large		Small	Mid	Large	
Circle	32.9	104.0	275.3		1.03	1.04	1.05	
Double-XOR	46.5	93.3	315.3		1.00	1.00	1.01	
Checkerboard	79.9	357.8	776.2		1.01	1.01	1.02	

for increasing values of p (number of features the method should construct) while the brute force approach constructed all features. Table II-4 summarizes the comparison of the two approaches. The timing results are presented as the ratio of brute force over `Heur2`. A time ratio of greater than one indicates the brute force approach taking longer than `Heur2`. The timing ratio was calculated for different values of the number of features to construct $p = \{50, 100, 500, 1000, 5000\}$ (for the heuristic approach), problem, problem size (number of variables), and sample size. For each problem, the time ratio was averaged over values of p (the brute force approach remains the same over this value) and either problem size (top portion of the table) or sample size (bottom portion of the table). Further visualizations of these results can be examined in Appendix A.II where the timing results are plotted by problem size, sample size, and number of features to construct.

Several trends in terms of the timing of each method individually and in comparison are observed from the table and graph in the appendix. First, in each of the problems presented the brute force approach requires more time than the heuristic method. This observation is expected as the brute

force is constructing all the features compared to a portion of the features and weight vector. Second, both the brute force and heuristic methods increase in time as the sample size increases for each problem (this is expected, with increasing sample the number of support vectors will likely increase causing an increase in the number of calculations for each method). Also, both methods increase in time as the size of the problem (number of variables, number of features) increase. For the heuristic method, the time results increase as the number of the weights to calculate increases (i.e., the value of p increases). However, the heuristic time increase is small as p increases compared to the increase in time for an increase in sample size or problem size for all but the smallest problem size; a majority of the heuristic method's time is spent in constructing the contributions matrix, with only a small percentage of time spent searching for the top p weights. Finally, the difference between the brute force and heuristic method grows as the size of the problem increases and the difference decreases as the sample size of the data increases.

The quality results are presented as a ratio of brute force over `Heur2`. The brute force approach has available the entire feature vector therefore it represents the ideal case and the quality ratio presented represents how close the heuristic method approaches the ideal. The closer the quality ratio is to one the better the heuristic method is at constructing all the top-weighted features. Since the heuristic method search via queries of large collections of features (the norm of the weight vector for specific variables and levels) it is expected that the returned features will have low weight features among them. The quality metric will therefore diverge from ideal as a number of low weighted feature are created. To counteract this situation, the two parameters, p - the number of features constructed and r - the number of features returned are both used where $r \leq p$. When a r is selected that is less than p, then those lower weighted features are removed from consideration. The heuristic method with different values of the number of features to construct $p = \{1000, 5000\}$ and number of features to return $r = \{5, 10, 25, 50, 100\}$ was compared to the brute force for the different problems, problem sizes, and sample sizes. For each problem also shown in Table II-4, the quality ratio was averaged over values for p, r, and either problem size (top portion of table) or sample size (bottom portion

of table). Additional graphs plotting the results under the different parameter values are shown in Appendix A.II.2.

From Table II-4 and the appendix several comments on the quality metric can be made. In general, the heuristic method diverges from the ideal quality metric of one as r increases. This observation can be explained by looking at the distribution of the weights. In Figure II-6(a-b), the weight distribution of the SVM model is shown. Focusing on Figure II-6(b), the Circle problem with 400 variables and 1000 samples is given in subplot showing only the top 25 weights. For this problem, two features hold the top weights after a gap in magnitude of the weights there exists a long tail of slowly diminishing weights. The heuristic method performs well to identify the top weights (the top two features). However, after identifying those top two features, the next highest weights are equal or close to the same value for many of the following features. The heuristic method does not perform as well in identifying the top feature within this noise. In order to achieve the ideal metric extensive searching for the top weights is required in the worst-case constructing all features and weights which degrades to the brute force approach. As with many other methods, there exist a trade-off between spending additional time searching and stopping the search with achieving a lower ideal in terms of finding all of the top weights.

An additional quality measure compares the classification performance of the full SVM model (brute force approach) to the returned features of the heuristic method (measured as AUC). This quality ratio is presented as a ratio of brute force over `Heur2`'s classification performance. A ratio of less than one indicates the heuristic method has a higher classification performance (a ratio of greater than one indicates the full SVM model has better classification performance). In Table II-5, this quality ratio was averaged over values for p, r, and either problem size (left portion of table) or sample size (right portion of table).

II.4.2 Related Methods

The heuristic method presented is the first of its kind to return the top features involved in the classification of a SVM model. Other methods do use SVM models to perform variable selection but

Table II-5: Comparing Brute Force and `Heur2`. The quality of the heuristic method was also assessed in terms of the classification performance of the full SVM model (brute force) compared to the classification performance of the returned features of the heuristic method. This quality ratio was measured for a number of features to construct - p (all features for the case of brute force), problem, problem size (number of variables), and sample size. For each problem, the classification performance ratio was averaged over values of p, r, and either problem size (left portion of table) or sample size (right portion of table).

	Classification Performance (AUC) - Brute Force / `Heur2`						
	Number of Samples				Problem Size		
Data Set	50	100	500	1000	Small	Mid	Large
Circle	0.927	0.942	0.866	0.774	0.895	0.884	0.853
Double-XOR	0.964	0.938	0.833	0.855	0.948	0.903	0.841
Checkerboard	1.005	1.016	0.863	0.794	0.891	0.930	0.937

do not identify the top features of the SVM; example methods include Recursive Feature Elimination (RFE) (Guyon *et al.*, 2002b), R^2W^2 (Weston *et al.*, 2000), l_0- and l_1- AROM (Weston *et al.*, 2003), and the methods of (Rakotomamonjy, 2003).

As a check of our methods, we compare the heuristic approach to identify the top-weighted features with the following: construct all features among the variables selected by a variable selection method (here we use the RFE as the variable selection method). RFE was run on the simulated data sets using a 1-fold 80/20 split on the data sets in order to train and test the performance of the SVM model. RFE was run using both linear (often the standard in practice) and polynomial kernels; for the polynomial kernel, the same kernel parameters as the heuristic method were used. Each iteration of the RFE algorithm eliminated the lowest ranked feature.

For the simulated data sets, the features constructed by the RFE variables are compared to the top-weighted features of entire weight vector. A comparison is also made to the `Heur2` method, where the same number of features that are created from the RFE variables are selected from the top `Heur2` features list (letting p=5000) and also compared to the top-weighted features from the entire weight vector. The quality is assessed as the norm of the features' weights made with RFE or `Heur2` divided by the norm of the same number of top weighted features of the entire weight vector. The features constructed from linear RFE variables performs poorly across all three data sets and is therefore excluded from further consideration. This result is not unexpected; the three simulated

data problems all have non-linear decision surfaces. The quality metric is plotted for the different problems and sample sizes for the heuristic method and polynomial RFE in Figure II-7. In general, the features of `Heur2` represent more of the top weights. Also, in addition the quality of the RFE method the efficiency of this method must be considered. RFE requires many iterations of learning a SVM model to select the variables whereas, the heuristic method learns the SVM model once (full timing results are available in Appendix A.III.1).

II.4.3 Motivating Example Problem

In the introduction, we briefly list the properties of the type of problems that this technique is aimed at:

- the SVM model is sparse or nearly sparse in feature space, that is, the weight vector $\mathbf{w}$ has relatively few components with large magnitude,
- there are high-order interactions without correlated low-order interactions that make it difficult for current variable selection methods to identify relevant variables,
- and the problem is too large to allow a brute force calculation of $\mathbf{w}$.

One particular problem where these properties are all true is the general parity problem. We have run our identification top weights method on 4-parity problems with variable of 250 variables (over 1.6×10^8 features) where we are able to identify the top feature. As the size of the 4-parity problem grows much beyond this point, the top-weighted features does not have a significant magnitude weight above the other features in order to fit the above criteria and be regularly found by our technique (in the future work section we discuss using a L0 or L1-norm SVM in order to create extremely sparse weight vectors that may be better suited to this and other tasks). This result is none-the-less important because most other variable selection/feature selection methods fail to detect this type of multivariate relationship (our method may be able to reveal new information for a domain).

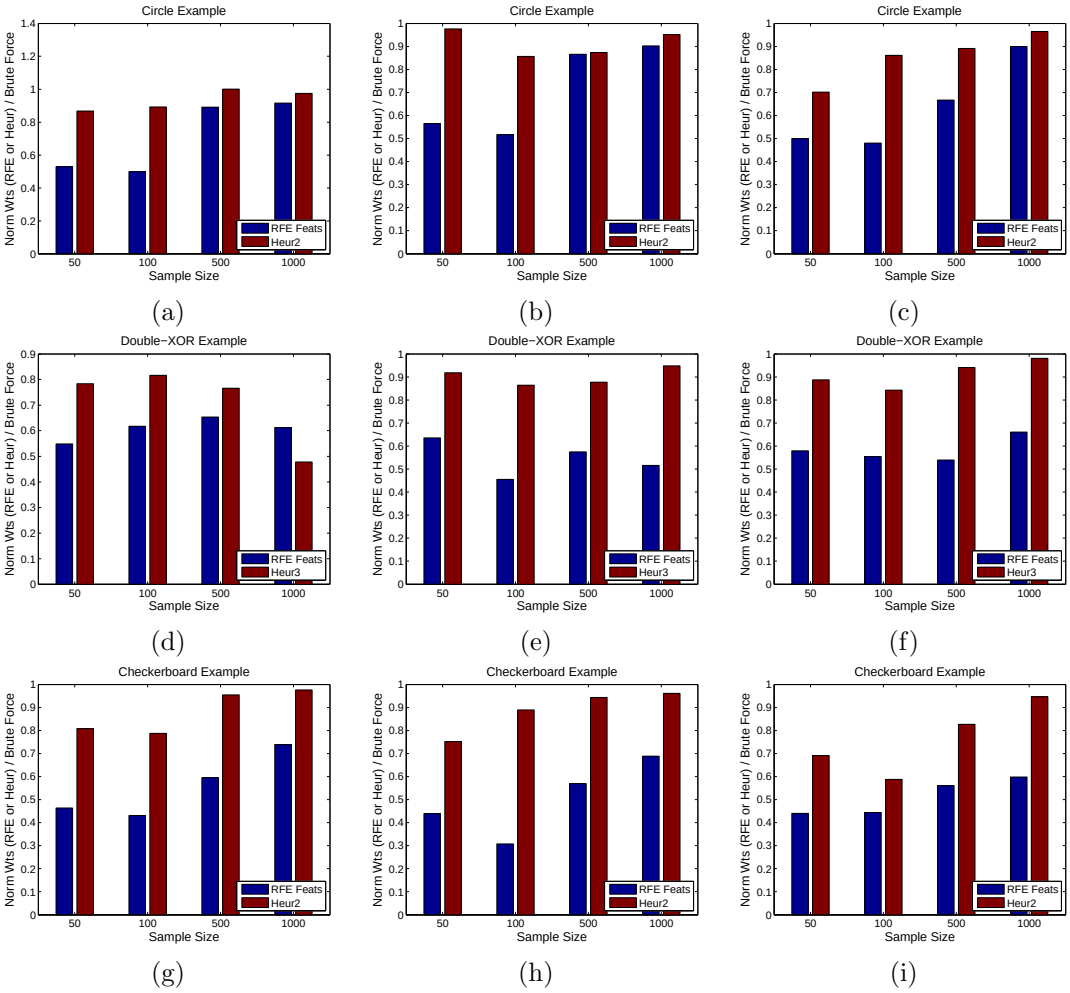


Figure II-7: Quality Results - RFE constructed Features: The quality results assess whether the variables returned by RFE can be used to construct the top-weighted features. The quality metric presented is the norm of the weights of the features constructed using the variables returned by RFE over the norm of the same number of the top-weighted features. The same number of features are selected from top of the features list returned by `Heur2` to use as comparison to the heuristic approach. The subplots present the quality metric over increasing problem sizes for the Circle (a-c), Double XOR (d-f), and Checkerboard (g-i) problems. The different color bars represent the quality metric the features from the RFE variables and the heuristic method. The different groupings within each plot show the quality metric for training data sets of increasing sample size.

Table II-6: Characteristics of Real Data sets

Data Set	Splice Site	Lung Cancer	Thrombin
Type	Splice Site Identification	Gene Expression Diagnosis	Drug Discovery
# of Vars	400	12,600	139,351
# of Samples	2000	160	2543
Design	10-fold c.v.	5-fold c.v.	1-fold c.v.

II.4.4 Real Data Sets

In addition to the simulated data analysis, the heuristic method was run on several diverse, real world data sets. In these data sets, the true classification function is unknown and the problems have high-dimensional data sets that do not allow the brute force approach to be run. Therefore, the method is evaluated in several indirect aspects. First, the features returned by the method are used to build a linear SVM model (that is, a linear model where the input variables are a subset of the features). The classification performance of this model is compared to that of a SVM model using all variables, and the variables selected by two methods: RFE and HITON (Aliferis *et al.*, 2003a, 2009a,b). The classification performance is used to gauge whether the selected features are informative to the classification decision. Also, the constituent variables of the top features are compared with the variables selected by RFE and HITON. Finally, the top features are detailed and compared against other published information on the data sets.

The first real-world data is in the drug discovery domain; the classification of whether bio-molecules are able (or not) to bind to thrombin (KDD Cup 2001, 2001). This data set illustrates the ability of our technique to scale to a very large number of variables (over 100,000) and present new information to the domain. The second task is diagnosis of lung cancer from oligonucleotide gene expression array data, specifically determining squamous versus adenocarcinoma types of cancer (Bhattacharjee *et al.*, 2001). The final task is to identify splice sites from a genomic sequence (Saeys *et al.*, 2003). For this task, we spend additional time relating the returned features found to other biological knowledge on this subject in the literature. The characteristics of the data sets are given in Table II-6. The results and details on the evaluation are presented in each of the following sections.

Table II-7: Classification Performance of Thrombin Data: The classification performance (measured by AUC) for SVM models using all available variables and the selected variables returned by either RFE or HITON is presented. The classification performance of the SVM model built using only the top 100 features (involving 16 variables) is also presented for comparison.

	All	RFE	HITON	Top 100 Feats
Num. Vars	139,531	8709	32	16
AUC	0.925	0.919	0.926	0.928

RFE was run with a linear kernel eliminating half of the variables at each iteration. HITON was implemented using either the G^2 (discrete data) of Fisher's z-test (continuous data) as the statistical test used.

Thrombin Data

A first experiment was performed on a very high-dimensional real data set - the Thrombin data set initially presented in KDD Cup 2001. The data set consists of 139,351 binary variables and a binary target. The data was split to train, validation and test sets. The parameters of the SVM were selected from the sets $d = \{1, 2, 3, 4\}$ and $C = 10^i, i = \{-8, ..., 3\}$ by optimizing the classification performance (measured as AUC) by training on train set and testing on the validation set. The heuristic method was then run on a SVM model with the optimal parameters trained on the train set to return the top 100 features. A new SVM classifier was trained on the train+validation set of the top 100 features; the performance is reported on the test set. The performance of the classifier was compared to a SVM model created using all variables and variable subset selected by two common feature selection methods HITON and RFE. The results of each method is reported in Table II-7.

The top features of heuristic method are listed in Table II-8. The variables in the top 100 features are also presented with the information of whether each variable is also selected by either RFE or HITON. Of the 16 variables in the top features 3 are also returned in variable set returned by RFE (a set of over 8000 variables). Also, none of the variables in the top 100 features are also returned by HITON (HITON selects 32 variables).

Table II-8: Top Features of Thrombin Data: The top 100 features constructed from the Thrombin data set. For each variable involved in the features listed whether this variables was also found by the variable selection methods of RFE and HITON is also presented.

Rank	Feature	Vars. in Top Features	Var. in RFE?	Var. in HITON?
1	$X_{16592}X_{16887}$	X_{6244}	N	N
2	$X_{16592}X_{17176}$	X_{6270}	N	N
3	$X_{16887}X_{17176}$	X_{6517}	Y	N
4	$X_{16592}X_{16597}$	X_{6523}	N	N
5	$X_{16597}X_{17176}$	X_{6526}	N	N
6	$X_{16895}X_{17176}$	X_{6737}	N	N
7	$X_{16592}X_{16895}$	X_{16558}	N	N
8	$X_{16597}X_{16887}$	X_{16592}	N	N
9	$X_{16597}X_{16895}$	X_{16597}	N	N
10	$X_{16887}X_{16895}$	X_{16837}	N	N
11	$X_{16865}X_{17176}$	X_{16847}	N	N
12	$X_{16592}X_{16865}$	X_{16865}	N	N
13	$X_{16597}X_{16865}$	X_{16887}	N	N
14	$X_{16865}X_{16895}$	X_{16895}	N	N
15	$X_{16865}X_{16887}$	X_{17176}	Y	N
16	$X_{16592}X_{17226}$	X_{17226}	Y	N
17	$X_{17176}X_{17226}$			
18	$X_{16895}X_{17226}$			
19	$X_{16887}X_{17226}$			
20	$X_{16597}X_{17226}$			

Table II-9: Classification Performance of Lung Cancer Data: The classification performance (measured by AUC) for SVM models using all available variables and the selected variables returned by either RFE or HITON is presented. The classification performance of the SVM model built using only the top 1000 features (involving 18 variables) is also presented for comparison.

	All	RFE	HITON	Top 1000 Feats
Num. Vars	12,600	19	16	18
AUC	0.991	0.986	0.978	0.993

Lung Cancer

The lung cancer data set is used to classify gene expression samples between squamous and adenocarcinoma types of cancer. The data was split following a nested 5-fold cross validation design (Aliferis *et al.*, 2003a) in order to estimate the performance of the model and optimize SVM parameters from the sets $d = \{1, 2, 3, 4\}$ and $C = 10^i, i = \{-8, \ldots, 3\}$.

The classification performance of the different methods on this data set are summarized in Table II-9. The classification performance of the SVM model with all variables is similar to that of the model built using only the top 1000 features (involving 18 variables). While the performance of the models using the subsets of variables selected by RFE (19 variables) and HITON (16 variables) is also similar but slightly lower.

The top features returned by the heuristic method are listed in Table II-10 as well as the variables returned by RFE and HITON. The features involve several variables not considered by HITON (zero variables intersect between the two sets) and RFE (two variables intersect between the two sets). In addition, the features typically involve combinations of 2 or 3 variables. Addition biological information about the selected variables is given in Appendix A.IV to allow further explorations by biologists and researchers in this domain.

Splice Data

The classification task of the splice data is to identify splice sites from DNA sequences. In most eukaryotic organisms, a gene is often not a continuous sequence of DNA; rather sections of DNA are spliced in and out to form the protein sequence. The regions of DNA that are coded into protein

Table II-10: Lung Cancer Data Top Features: The top 20 features returned by the heuristic method are listed alongside the variables returned by the variable selection method RFE and HITON.

Rank	Feature	RFE	HITON
1	$X_{23}X_{2515}X_{12097}$	X_{1060}	X_{288}
2	$X_{23}X_{3157}X_{12097}$	X_{8201}	X_{2093}
3	$X_{23}X_{4935}X_{12097}$	X_{6814}	X_{3119}
4	$X_{23}X_{1907}X_{12097}$	X_{7366}	X_{3255}
5	$X_{23}X_{11942}X_{12097}$	X_{12150}	X_{3676}
6	$X_{23}X_{4934}X_{12097}$	X_{8914}	X_{4525}
7	$X_{23}X_{205}X_{12097}$	X_{1376}	X_{4596}
8	$X_{2515}X_{3157}X_{12097}$	X_{8727}	X_{6686}
9	$X_{23}X_{11436}X_{12097}$	X_{6536}	X_{6905}
10	$X_{23}X_{4983}X_{11942}$	X_{8429}	X_{8843}
11	$X_{23}X_{3157}X_{4983}$	X_{1679}	X_{9071}
12	$X_{23}^2X_{12097}$	X_{6908}	X_{10139}
13	$X_{23}X_{1906}X_{12097}$	X_{11743}	X_{10525}
14	$X_{23}X_{205}X_{4983}$	X_{4786}	X_{10936}
15	$X_{23}X_{9977}X_{12097}$	X_{7756}	X_{11300}
16	$X_{2515}X_{4935}X_{12097}$	X_{11355}	X_{11359}
17	$X_{23}X_{1905}X_{12097}$	X_{10997}	
18	$X_{23}X_{2515}X_{8021}$	X_{1668}	
19	$X_{1907}X_{2515}X_{12097}$	X_{12414}	
20	$X_{3157}X_{4935}X_{12097}$		

Table II-11: Classification Performance on Splice Data: The classification performance (measured by AUC) for SVM models using all available variables and the selected variables returned by either RFE or HITON is presented. The classification performance of the SVM model built using only the top 1000 features (involving 11 variables) is also presented for comparison.

	All	RFE	HITON	Top 1000 Feats
Num. Vars	400	400	26	11
AUC	0.982	0.982	0.926	0.952

are referred to as exons and non-coding segments are referred to as introns. A splice site refers to the border of the exon/intron or intron/exon transition. Typically, the intron is marked by two consensus dinucleotides of GT at the 5' end (the donor site) and AG at the 3' end (the acceptor site). We focus on identification of the acceptor site, although a similar analysis could be made for the donor site. The prediction of many genetic markers and signals have been studied using many supervised learning algorithms (see (Haussler, 1997) for reviews and references).

For this analysis, sequence data from *Arabidopsis thaliana* is used to construct the data set as described in (Degroeve *et al.*, 2002) and (Saeys *et al.*, 2003). Each data sample consists of 50 nucleotides upstream and 50 nucleotides downstream of the consensus acceptor site. The nucleotides are converted to 400 binary features. The training data set consists of 1000 positive and 1000 negative instances. A testing set has 281 positive and 7643 negative instances.

Model parameters were selected via 10-fold cross validation from the sets degree $= \{1, 2, 3, 6, 9\}$ and $c = \{0.001, 0.05, 0.1\}$ (choice of the parameter options was influenced by previously published results). The best model parameters were selected via cross-validation by maximizing AUC and found to be degree 6 kernel with $c = 0.05$. With the parameters of the model selected, a final SVM model was created on the training data set to examine the top features and weights for each problem. The heuristic method was run on this model asking for the top t=100000 and r=1000 features (out of a possible 5.99×10^{12}). These features are used to map the data to a small subset of feature space. A new model was trained on the mapped data and compared against the SVM classifier on the original data. The classification performance was calculated for an SVM model using all variables, the variables selected by RFE, HITON, and the top 1000 features determined by

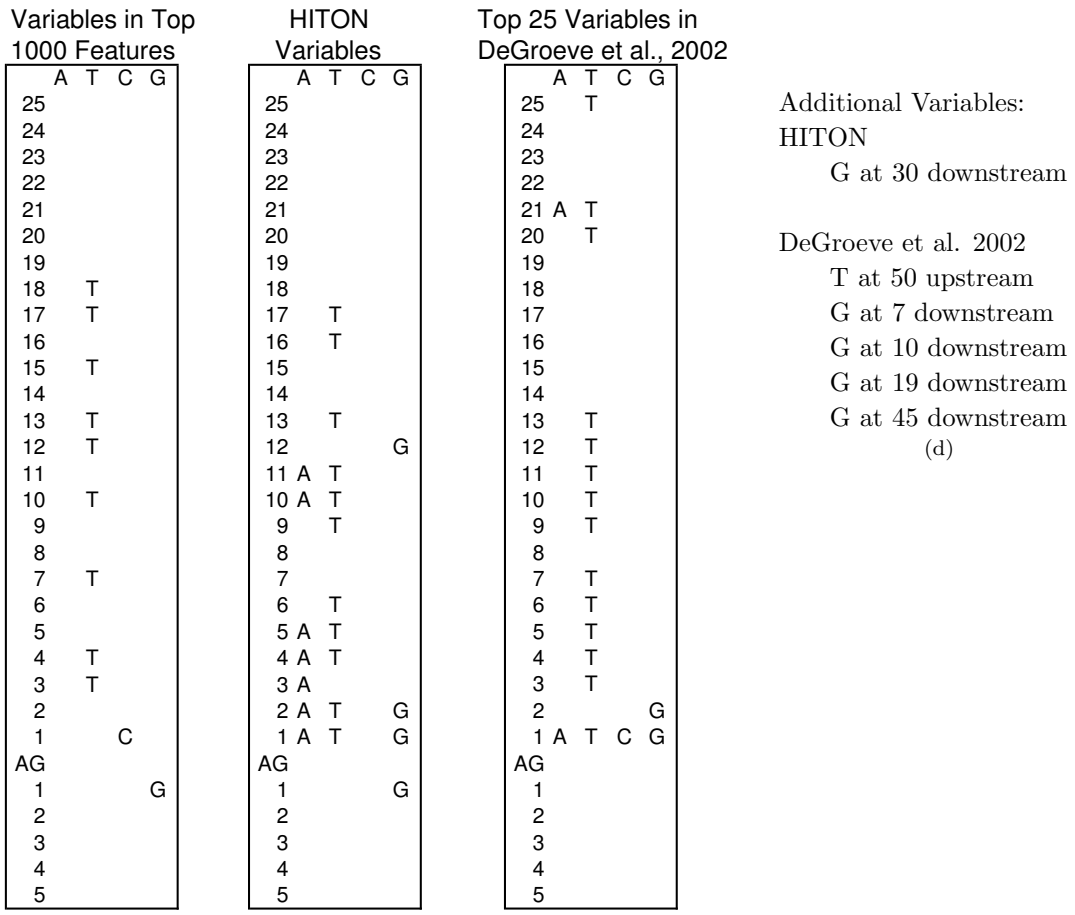


Figure II-8: Splice Site Identification: (a) The variables involved in the top 1000 features. (b) The variables selected by HITON (an additional variable is not shown, G at 30 downstream). (c) The top 25 variables reported in (Degroeve *et al.*, 2002) using a method similar to RFE (five additional variables not shown at 50 upstream (T) and 7, 10, 19, and 46 downstream (G)).

the heuristic method and shown in Table II-11. The top features returned by the heuristic method are able to accurately classify the test data, therefore we shall more closely examine the features returned by the method.

The constituent variables of the top features are visualized along with the variables selected by HITON and the top 25 variables found in (Degroeve *et al.*, 2002) using an method similar to RFE in figure II-8. Several observations can be made from this figure. First, the variables selected by all methods are generally close to the splice site. This suggests there is little interaction of variables at a distance with the splice site machinery; several other papers using this same data set have

made similar observations. In fact, (Degroeve *et al.*, 2002) suggest only the variables less than 10 nucleotides upstream of the splice site and 3 nucleotides downstream of the site are important for prediction.

Also, the top feature is the presence of the C nucleotide directly upstream of the splice site (a list of a selection of the top features is in Figure II-9). The next largest-weighted features consists of either pairs, triplets, and quartets of variables involving 1, 2, or 3 groupings of the upstream T's and the C or pairs, triplets, and quartets of the upstream T's alone. These results are again consistent with other published results. In (Lim & Burge, 2001), the authors show that the C nucleotide directly upstream of the splice site is the most frequent at this position (over 60%). Additionally, the many features involving combinations of the upstream T's is corroborated. In *A. thaliana* (and also humans), the presence of a small subset of pentamers applies a large contribution to splice site recognition. The pentamers associated with *A. thaliana* are all heavily based on sequences of T's, e.g., TTTTT, TCTCT, TTCTT, TTTTA, etc. These short sequences are not dependent on relative position to the splice site.

Another observation can be made on the features receiving positive or negative weights. A negative weight suggest that this feature when active indicates that there is not a splice site present, while an active positive weighted feature indicates there is a splice site present. The variable, corresponding to a C in the position before the splice site, only occurs in positive features suggesting this nucleotide in this position is particularly indicative of a splice site. In contrast, the variable, corresponding to a G in the same position, only occurs in negative features therefore suggesting this nucleotide is not indicative of a splice site.

II.4.5 Summary of Real Data Sets

The method of selecting the top features are compared to variable selection through the indirect measure of classification performance. The features are used to create a linear SVM model, while alternative SVM models are build using all variables, and the variables selected by RFE and HITON. The number of variables returned by the variable selection method (and the number of variables

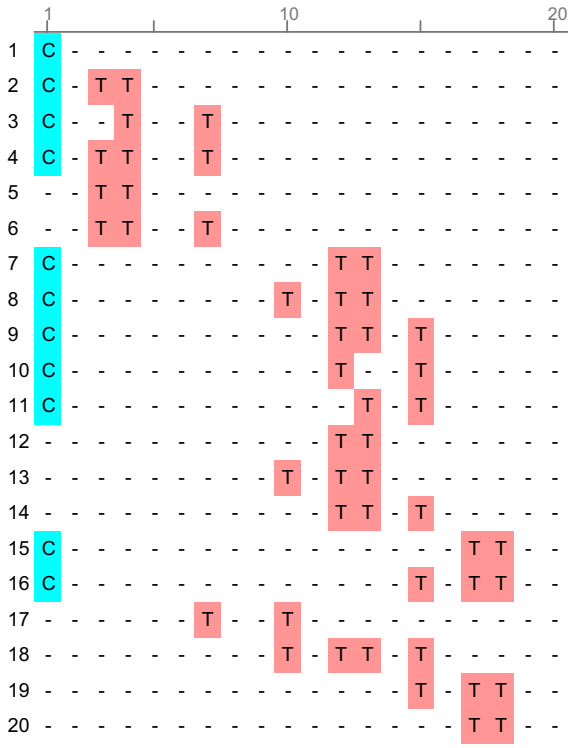


Figure II-9: Splice Site Identification: The top 20 features are listed. Each feature (numbered down the list) is a combination of the variables in the row. The numbers across the columns indicate the position from the splice site.

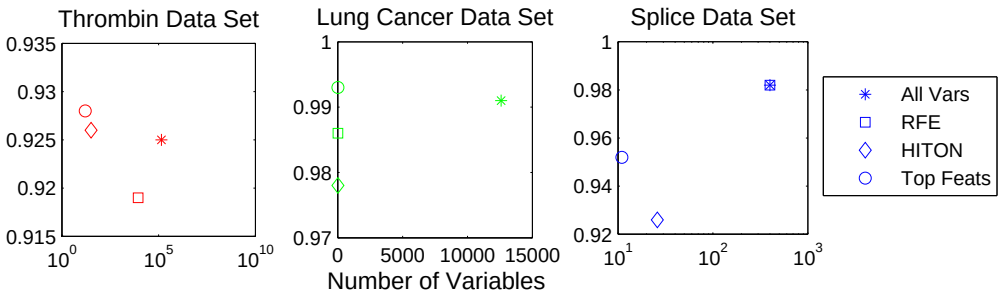


Figure II-10: Real Data Sets Results: Plots of the classification performance versus the number of variables used in the model. The variables are selected via four methods: all variables included, RFE, HITON, and top features selected.

involved in the top features) are plotted by the classification performance for each data set in Figure II-10. This figure illustrates this new method to create effective models for classification involving a few number of variables, while providing additional information about the top features (combinations of variables) that may be important to a domain.

II.5 Discussion

In this Chapter, we present an efficient, heuristic method for identifying the largest weights of a polynomial support vector machine model. This algorithm provides a new ability to understand polynomial SVM models. Prior to this work in order to understand how an SVM model decides on a classification either the entire feature weight vector, $\mathbf{w}$, would need to be identified or some variable selection method would be applied to identify a smaller subset of the "best" variables (where best can be defined via several criteria). Explicating the entire weight vector is a prohibitively expensive process, that becomes intractable for many problems that SVM models are aimed at. Whereas, a particular variable selection method may not provide any new insight into the functionality of the SVM model.

The experimental results presented here are over several different problems involving continuous and discrete data. However, the scope of the experiments is still limited and several questions in the application of this new method are not fully addressed in the work. First, the choice of the kernel parameters is set for the simulated data sets so that the minimal degree kernel is selected to

include all features important for classification. In the real world data sets, the kernel parameters are selected by cross validation classification performance (a standard technique in many experimental designs). However, it should be noted that as the degree of the kernel defines the features of the SVM choosing a degree that is too low may result in not finding a feature that may be important in classifying the data and a degree that is too high may results in redundant features identified and an increased search space for the algorithm to parse.

Also, the heuristic method developed here is general purpose restricted only to the polynomial kernel and does not consider the specifics of the data type or how it is encoded. We employ the standard practices of typically normalizing continuous data to a mean of zero and standard deviation of 1 and encoding binary data as 0/1. The choice of the input to train the SVM does not affect how the heuristic method is run, however, it may impact the SVM model trained and consequently the feature list returned. For instance, the effect of different binary encodings is described in Section II.4.1. Future iterations of the heuristic method could be developed to tailor to a specific data type or encoding, in order to remove redundancy in the search procedure.

In addition to the practical application of the procedure there are theoretical questions raised. For instance, the heuristic method returns the top-weighted features however, when are the most "relevant" features guaranteed to be the highest weighted? Are there distributions where the features important for classification will not be found and is it possible to assess if a specific data set falls into such a case? Questions such as these are outside the scope of this work, but are important for the understanding and application of SVM models to specific domains.

II.5.1 Future Directions

We consider this research a first step in attempting to identify the top-weighted features of an SVM model; there are many future directions of this work, several of which will be discussed here. First, the SVM models used within this Chapter all use the standard L2 SVM models learned via the widely-used LibSVM package. Please note that the L2 norm tends spread the weighting across

the features. In the future, we plan to investigate the use of L1 or L0 SVMs which may provide more sparsity in the weight vector allowing for easier searching by the heuristic method.

The search procedure itself can be extended and explored. Here, the search was guided by groupings of features involving a variable v at a level l; however, collections of variables could be considered. In section II.2.1, several general formulations are presented to consider sets of features and the norm of the weight vector over those features. Additional subsets of features where the grouping is by both variable v, degree of the feature l, and specific exponent p of variable v can be constructed:

$$\|\mathbf{w}_l^{v,p}\|^2 = \binom{d}{l} \sum_{j,k=1}^{n} \alpha_k \alpha_j H_l(\mathbf{x}_j^{v,p}, \mathbf{x}_k^{v,p}) H_l(\mathbf{x}_j^{\setminus v}, \mathbf{x}_k^{\setminus v}). \tag{II.46}$$

Another direction is to extend the process here of selecting the top weighted features with Markov-blanket based variable selection algorithms. A new algorithm Feature Space Markov Blanket (FSMB) attempts to combine these approaches (this algorithm will be discussed in Chapter III with only a short introduction here). The main idea of FSMB is to identify the Markov Blanket of T in *feature space* where multivariate associations become pairwise associations instead of in the original variable space. FSMB employs an SVM to dictate which features may have pairwise association with T in feature space. To avoid explicitly computing all features, FSMB uses the heuristic method described here to identify the top features. A subset of the top-weighted features returned are selected and the original data is mapped to the subset of feature space defined by the selected features. Finally, the new feature space data set is passed to a Markov Blanket identification method (MMMB, HITON, PCMV, etc.) to select the Markov Blanket in feature space.

This new approach has been run on several of the real data sets used in this Chapter and shown to have ability to return features that have good classification performance. For instance on the lung cancer data set, the classification AUC for the SVM with all variables, variables selected by RFE and variables selected by HITON is 0.991, 0.986, and 0.978 respectively. FSMB returns 4 features that provide a classification performance of 0.979. For the Thrombin data set, RFE selected over 8000 variables, HITON selected 32 variables, FSMB selects 5 features while providing the following classification performances respectively 0.919, 0.926, 0.939.

The FSMB method works well compared to many MB-based feature selection on two problems in particular. The first example is the general parity problem where traditional MB-based variable selection algorithms are often unable to detect the variables involved in the high-order interaction relevant to the target. The second example is for problems with redundant mechanisms. In this type of problem (see Figure II-3), variable A when "on" causes variable C to be "on". When variable A is not on this causes variable B to be on. Also, when variable B is on this causes variable C to be on. The nature of this network is to assure the variable C is always on either by the variable A or variable B. For this example, the SVM weights the feature involving A, B, and AB highly and the FSMB algorithm will return AB. This problem is difficult for traditional MB-based variable selection algorithms which will not be able to properly detect that A and B are parents of C and therefore in the Markov Blanket. These examples and other problems will be more formally explored in the next Chapter focusing on the FSMB method.

II.6 Conclusions

Support Vector Machines (SVMs) models have been widely used to classify data. However, the reasoning behind the classification is complex, and previously unavailable to the user. This Chapter examines a method to explicitly determine the decision function used to classify data for polynomial SVMs. In particular, a heuristic method was designed to identify the highly weighted features of this decision function. These features may give insight into how the SVM classifies data and provide information on the features and variables relevant to the target class.

CHAPTER III

MARKOV BLANKET-BASED VARIABLE SELECTION IN FEATURE SPACE

Variable selection (a.k.a. feature selection) for a target variable of interest, T, is an important problem in prediction modeling that has drawn significant attention. A new variable selection algorithm is presented. This algorithm, Feature Space Markov Blanket (FSMB), exhibits two attractive properties under certain conditions: (i) it is able to select multivariately-predictive variables even when these variables have a small or no pairwise association with T (e.g., they are associated with T via a parity function), and (ii) it is able to identify a minimal variable subset required for optimal prediction. FSMB combines ideas from kernel-based and Markov Blanket-based variable selection methods to borrow the theoretical properties from each; to our knowledge, it is the first such filtering algorithm. The advantages of FSMB are empirically shown over previous approaches in simulated and real, large data sets and illustrate its potential for principled, efficient, and high-quality variable selection. For some cases, FSMB is able to identify 2 or 3 features which can then be used to visualize the discriminative power of the features. Additionally, data sets where Markov Blanket-based methods perform poorly compared to FSMB suggest the existence of multivariate relationships in the underlying data.

III.1 Introduction

Variable selection for predictive modeling (also called feature selection in the literature) has received considerable attention during the last three decades both in statistics and in machine learning (Guyon & Elisseeff, 2003). Intuitively, variable selection for prediction aims to select only a subset (proper or not) of variables for constructing a diagnostic or predictive model for a given classification or regression task. Reasons for performing variable selection include: (i) improving the prediction power and addressing the curse-of-dimensionality, (ii) reducing the cost of observing, storing, and

using the predicting variables, and finally (iii) gaining an understanding of the underlying modeling task.

There are many ways to formally define the problem of variable selection giving preferences to different variable subsets and predictive models. In this Chapter, the following formalism for the problem of variable selection is defined. Let $\mathbf{x}$ be a random vector $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, let T be a random variable such that $T = \{+1, -1\}$, where the random variables $\langle \mathbf{x}, T \rangle$ follows a joint probability distribution P. Let $\mathcal{D} = \{\langle \mathbf{x}_k, T_k \rangle\}_{k=1}^{n}$ consist of n independent samples of $\langle \mathbf{x}, T \rangle$. We further assume a given learner A is provided that can construct a predictive model $M_{\mathbf{F}}$ for T using the the sample $\mathcal{D}$ projected on a subset of all the variables $\mathbf{F} \subseteq \mathbf{x}$. Finally, a performance metric $E(M_{\mathbf{F}}, \mathbf{F})$ is given that scores the model and the selected variable subset $\mathbf{F}$. The problem of variable selection is to select the variable subset $\mathbf{F}$ that maximizes the performance $E(M_{\mathbf{F}}, \mathbf{F})$.

In a learning setting where variable selection is not performed, the performance function only depends on the prediction power of the model, e.g., the accuracy or expected loss. However, when variable selection is desired, the performance may also depend on the number or cost of the variables selected for inclusion in the model; hence the evaluation function accepts the second parameter $\mathbf{F}$. Typical performance functions prefer (scores highly) the smallest variable subset that can be used to construct the model with the highest prediction accuracy. Other performance functions may try to achieve a balance between prediction power and cost of observing the variables.

Markov Blanket-based and kernel-based methods illustrate two prominent paradigms in variable selection. The former follows a principled approach to variable selection and is able to guarantee some desirable theoretical properties such as optimality under certain broad conditions (e.g., data is i.i.d., Markov condition, faithfulness condition, etc.). Two examples of the conditions being violated are: (i) the optimal variable subset contains multivariate associations whose participant variables have no detectable univariate associations with T and (ii) the target variable is caused by variables from specific redundant mechanism distributions (see section III.3 for further details). The kernel-based approach is able to capture the multivariate and redundant relationships in such situations even in very high dimensional data sets. In this Chapter, we introduce a new variable selection

algorithm that combines the advantages of both approaches in a non-trivial way, this new algorithm we call Feature Space Markov Blanket (FSMB).

In the following sections, the Markov Blanket-Based variable selection approach is reviewed (Section III.2). The prototypical problems for which the FSMB algorithm is designed are discussed in Section III.3 (these problems are cases where Markov Blanket-based techniques fail). Section III.4 reviews kernel-based variable selection from which the new method draws upon. A discussion on general variable selection methods and their applicability to the prototypical problems is given in Section III.5. Section III.6 presents the new FSMB algorithm. Finally, in section III.7 the experimental evaluation and comparison of the algorithms are presented. We conclude in section III.8 with final remarks on the method including its limitations and future directions.

III.2 Markov Blanket-Based Variable Selection

A principled approach in variable selection is based on identifying the Markov Blanket of the prediction variable T. A Markov Blanket of T, denoted as $MB(T)$, is defined as a minimal set conditioned on which all other variables in $\mathbf{x}$ become independent of T (the Markov Boundary in the terminology of Pearl (1988)):

$$P(T|\,\mathbf{x}) = P(T|MB(T)). \tag{III.1}$$

Thus, all information for predicting T is contained within the $MB(T)$ and therefore, intuitively it seems that these should be the only variables required for optimal prediction. The latter statement is not true in general however, as the learner and the performance metric used are important. For the $MB(T)$ to be the solution to the variable selection problem as it was defined above, two conditions are sufficient (Tsamardinos & Aliferis, 2003):

1. *The algorithm A constructing the prediction model can learn the distribution* $P(T|MB(T))$.

   For example, if the $MB(T)$ predicts T via a highly non-linear function but the learner employed is linear, then *for this specific learner* it may be preferable to select a different set of variables.

2. *The performance metric is such that perfect estimation of the probability distribution of T is required with the smallest number of variables.*

   Consider for example the following distribution where T and x_1 are both binary: $P(T = 1|x_1 = 1) = 0.6$ and $P(T = 1|x_1 = -1) = 0.7$ and assume the marginal $P(T = 1) = 0.65$. $MB(T) = \{x_1\}$ and knowledge of the value of x_1 is necessary for optimal density estimation or calibrated accuracy. However, if the goal is to maximize accuracy (i.e., percentage of correct classifications) then x_1 is not necessary in the model: whether we know its value or not, T is always classified as $T = 1$. In this case, the $MB(T)$ will be a superset of the minimal subset required for optimal performance. In addition, a preference for selecting the smallest possible number of variables is important. If for example, it is not the minimum-size but the minimum-cost maximally-predictive variable subset that is sought, then the $MB(T)$ maybe a poor approximation.

Frequently in variable selection applications the above conditions hold or hold approximately. In these cases, it makes theoretical sense to identify the $MB(T)$ as a first approximation of the variable subset to select. Many time- and sample-efficient *MB*-identifying algorithms appear in the literature, including HITON (Aliferis *et al.*, 2003a, 2009a,b), MMMB (Tsamardinos *et al.*, 2003c), IAMB (Tsamardinos *et al.*, 2003a), PCMB (Peña *et al.*, 2007), GS (Margaritis & Thrun, 1999), Koller-Sahami (Koller & Sahami, 1996), among others. Most, if not all of these algorithms are based on the theory of Bayesian Networks (Pearl, 1988). We assume the reader is familiar with the Bayesian Network formalism and theory, although a few important concepts are re-iterated here.

Definition III.1. *Let P be a discrete joint probability distribution of the random variables*[1] *in some set $\mathcal{V} = \mathbf{x} \cup T$ and $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a Directed Acyclic Graph (DAG). We call $\langle \mathcal{G}, P \rangle$ a (discrete)* Bayesian network *if $\langle \mathcal{G}, P \rangle$ satisfies the Markov Condition: every variable is independent of any subset of its non-descendant variables conditioned on its parents (Pearl, 1988; Spirtes* et al.*, 1993; Glymour & Cooper, 1999; Pearl, 2000; Spirtes* et al.*, 2000; Neapolitan, 2003).*

[1] Variables are also interchangeably called nodes or vertices in the context of a Bayesian network.

The graph of a network in conjunction with the Markov Condition directly encode some of the independencies of the probability distribution and entail others (see Neapolitan, 2003, pp. 70 for a definition of entailment). The faithfulness condition below, asserts that the conditional independencies observed in the distribution of a network are not accidental properties of the distribution, but instead due to the structure of the network.

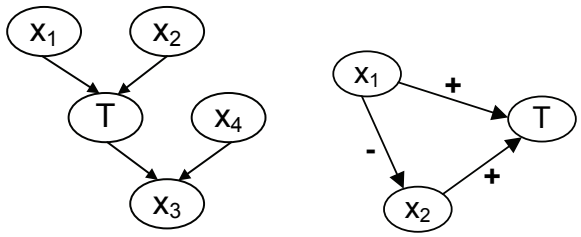
Definition III.2. *If all and only the conditional independencies true in the distribution P are entailed by the Markov condition applied to $\mathcal{G}$, we will say that P and $\mathcal{G}$ are* faithful to each other *(Spirtes* et al.*, 1993, 2000; Neapolitan, 2003). Furthermore, a distribution P is* faithful *if there exists a graph, $\mathcal{G}$, to which it is faithful.*

Definition III.3. *A Bayesian network $\langle \mathcal{G}, P \rangle$ satisfies the* faithfulness condition *if P embodies only independencies that can be represented in the DAG $\mathcal{G}$ (Spirtes* et al.*, 1993). We will call such a Bayesian network a* faithful network*.*

The following theorem is utilized in most constraint-based algorithms such as the ones presented here:

Theorem III.1. *In a faithful BN $\langle \mathcal{G}, P \rangle$ on variables $\mathcal{V}$ there is an edge between the pair of nodes x_1 and x_2 in $\mathcal{V}$ iff $Dep_P(x_1; x_2 | \mathbf{x_k})$, for all $\mathbf{x_k} \subseteq \mathcal{V}$ (Spirtes* et al.*, 1993).*

Faithfulness is important for $MB(T)$ identification. Consider this reinterpretation of the theorem: *a BN is faithful if a dependency (association) exists between any pair of nodes connected by an edge, conditioned on any other subset of variables (Spirtes* et al.*, 2000).* Thus, all direct (to T) multivariate associations can be discovered incrementally, since if an edge $x_i \rightarrow T$ exists, (i.e., x_i participates in a multivariate dependency) there should be a pairwise (conditional or not) association too. In addition, *in a faithful Bayesian Network, the $MB(T)$ (i) is unique and (ii) has a graphical interpretation: it is the set of parents, children, and spouses of (i.e., nodes with common children with) T* (Neapolitan, 2003). An example graph of a Bayesian Network is shown in Fig. III-1(a) whose $MB(T)$ is $\{x_1, x_2, x_3, x_4\}$.



(a) (b)

Figure III-1: Example Bayesian Networks: (a) The parents, children, and spouses (nodes with common children) of T are the $MB(T) = \{x_1, x_2, x_3, x_4\}$; these are the only nodes required for perfect estimation of the distribution of T. (b) A small 3 variable network example illustrating a redundant mechanism for activating variable T.

For the rest of this Chapter, the *MB*-based variable selection method used is HITON; a successful *MB* algorithm that is efficient and selects a highly predictive variable subset as shown empirically on a variety of problems (Aliferis *et al.*, 2003a, 2009a,b). HITON first identifies the $MB(T)$ to significantly reduce the number of variables to consider for inclusion in the output subset. It then performs a backward search for eliminating variables from the $MB(T)$ that do not affect the predictive performance (e.g., as this is measured by accuracy or the Area Under the Receiving operating characteristic curve of a Support Vector Machine trained on the sample projected on the specific variable subset).

Apart from time-efficiency and quality of output, one of HITON's attractive properties is its theoretical guarantees: if the data distribution is faithful, then it will provably, in the sample limit, identify the $MB(T)$.[2] Thus, if both conditions mentioned at the start of this section also hold, HITON will optimally solve the variable selection problem.

III.3 Problems of Interest

In this work, we focus on problems where Markov Blanket-based variable selection methods fail. Two prototypical examples that illustrate this property will be discussed throughout the chapter,

[2]Actually, some false positives may enter; see Tsamardinos *et al.* (2006b) for an analysis of this case; these will be removed by the backward search in the next phase.

from both theoretical basis and also experimentally to verify the new algorithm's ability on such problems.

HITON and all other *MB*-based algorithms mentioned may not identify variables as belonging in the $MB(T)$ when the data distribution is not faithful. Consider again for example the network in Fig. III-1(a). Let us assume that all variables are binary taking values $\{-1, 1\}$ and that T is the XOR of variables x_1, x_2, i.e., $T = 1$ when x_1 and x_2 take different values, and $T = -1$ otherwise (the generalization of this example is a parity function). In addition, let us suppose the x_1 and x_2 are independent of each other (conditioned on the empty set) and their marginal probability of taking the value of 1 is 0.5. In this extreme case, there is no pairwise association between either of these two variables and T. Only when both of them are examined together, a strong multivariate association with T emerges. Thus, this is a non-faithful distribution.

HITON and the other *MB*-algorithms mentioned depend on a parent or a child variable of T having a detectable pairwise association with T. *Thus, in the XOR example above, HITON will fail to identify any of the four variables* $\{x_1, x_2, x_3, x_4\}$ *as belonging in the* $MB(T)$. In practice, when the sample is finite HITON may also miss variables that have a small pairwise association with T, even when they have a strong multivariate association. This problem is a specialization of the general problem of parity functions.

Another example of *MB*-based algorithms failing to identify the $MB(T)$ when the data not faithful is illustrated in the redundant mechanism example of Figure III-1(b) (Scheines, 2009). In this type of problem, variables x_1 when "on" causes variable T to be "on". When variables x_1 is not on this causes variable x_2 to be on. Also, when variable x_2 is on this causes variable T to be on. The nature of this network is to assure the variable T is always on either by variable x_1 or variable x_2 . In this example, the probability distribution may be set such that there is no detectable association between x_1 and T consequently, HITON will miss identifying variable x_1 as belonging in the $MB(T)$ for those distributions.

III.4 Kernel-Based Variable Selection

The invention of kernel-based methods was a breakthrough step towards addressing the problem of detecting multivariate associations of a group of variables that exhibit no univariate association with T. These methods involve mapping the data from variable space to a constructed feature space possibly containing interaction terms, where the multivariate associations become pairwise associations between the features and T. By performing the mapping implicitly, the constructed features do not have to be computed.

In recent years, many researchers have worked on the problem of variable selection with SVMs[3]. There are methods that rank the variables by scaling factors, where scaling factors are added into the kernel and are optimized in the training of the model (Weston *et al.*, 2000). The Recursive Feature Elimination method (RFE) ranks each variable by removing each variable from consideration in turn to construct a score, removes the lowest ranked variables, and iterates through this process (Guyon *et al.*, 2002b). Recently, methods for constructing SVMs with sparse weight vectors have been developed (cf. l_0- and l_1 AROM (Weston *et al.*, 2003) and the methods of (Rakotomamonjy, 2003)). For the most part, these methods have been developed for linear SVMs. In (Weston *et al.*, 2003), the authors also describe minimizing the zero-norm with non-linear kernels. We will consider RFE as an examplar from this set of techniques due to its prominent use across many domains as a variable selection method.

Before delving into the details of kernel-based variable selection methods, some background and notation on Support Vector Machines is presented (Vapnik, 1998). In this Chapter, we focus on a soft-margin 1-norm Support Vector Machine with full polynomial kernel of degree d for a binary classification problem (Schölkopf *et al.*, 1999).[4] We will work with the canonical polynomial mapping $\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^f$ of degree d that satisfies for $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^m$, $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$. A component of the vector $\mathbf{x} \in \mathbb{R}^m$ is called a variable (denoted as x_i) and a component of a feature vector $\Phi(\mathbf{x}) \in \mathbb{R}^f$ is a feature, denoted as $\Phi_i(\mathbf{x})$. For this mapping, each component consists of a multivariate monomial of at most degree d there are $f = \binom{m+d}{d}$ such components (features). The

[3]Often called feature selection in the literature, but here we distinguish between variable selection (selecting a subset of the input variables) and feature selection (selecting a subset of the features).

[4]The polynomial kernel is the only kernel discussed here because it is needed for the FSMB method.

SVM model determines a linear function in feature space (via the mapping Φ) of the following form,

$$h(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b. \tag{III.2}$$

A sample vector $\mathbf{x}$ is classified by the decision function $g(\mathbf{x}) = \text{sgn}\,(h(\mathbf{x}))$. The weight vector $\mathbf{w}$ is given by the equation,

$$\mathbf{w} = \sum_{k=1}^{n} a_k t_k \Phi(\mathbf{x}_k) = \sum_{k, a_k \neq 0}^{s} a_k t_k \Phi(\mathbf{x}_k), \tag{III.3}$$

where the second summation is over the support vectors that is, the data samples with $a_k \neq 0$ (we let s denote the number of support vectors). In the equations, the a's are the minimizers for the optimization problem,

$$\begin{aligned} &\min_a \frac{1}{2} \sum_{k=1,l=1}^{n} a_k a_l t_k t_l K(\mathbf{x}_k, \mathbf{x}_l) - \sum_{k=1}^{n} a_k \\ &\text{s.t. } \sum_{k=1}^{n} t_k a_k = 0, C \geq a_k \geq 0, k = 1, \ldots, n \end{aligned} \tag{III.4}$$

and the a_i's are called the Lagrange multipliers.

The weight vector w defines a decision hyperplane in feature space that balances the margin of separation between the two classes (equal to $2/\|w\|_2$) and the 1-norm of the distances ξ_k of the data falling on the wrong side of the margin of separation. The classification function g can be written as:

$$g(x) = (\sum_{k=1}^{n} a_k t_k K(\mathbf{x}_k, \mathbf{x}) + b) \ . \tag{III.5}$$

By using the kernel function, K, and the dual formulation of the optimization problem, the explicit mapping to feature space, Φ is never computed. As a result of the "kernel trick", a linear decision surface is constructed in an extremely high dimensional feature space without explicitly mapping to the feature space. Consequently, the SVM model consists of the Lagrange multipliers and support vectors rather than the features and weights of the decision surface. The weight vector $\mathbf{w}$ is never explicitly constructed.

Letting $x_0 = 1$, the polynomial kernel can be written in the form,

$$K(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{m} x_i x_i' + 1 \right)^d = \left(\sum_{i=0}^{m} x_i x_i' \right)^d. \tag{III.6}$$

One choice of Φ corresponding to this kernel maps the input variables to a set of features consisting of all products of the variables up to degree d. Consider the space $Q_{m,d}$ to index the features; let $Q_{m,d} = \{\mathbf{q} = \langle q_1, ..., q_m \rangle \mid q_i \in \{0, ..., d\} \text{ for } i = 1, ..., m \text{ and } \sum_{i=1}^{m} q_i \leq d\}$ then, $\Phi(\mathbf{x}) = (\Phi_{\mathbf{q}}(\mathbf{x}))_{\mathbf{q} \in Q_{m,d}}$ where

$$\Phi_{\mathbf{q}}(\mathbf{x}) = c_q \mathbf{x}^{\mathbf{q}} = \sqrt{\binom{d}{\mathbf{q}}} \prod_{v=1}^{m} x_v^{q_v}. \tag{III.7}$$

We use the multinomial notation $\mathbf{x}^{\mathbf{q}} = x_0^{q_0} \cdots x_m^{q_m}$ and $\binom{d}{\mathbf{q}} = \frac{d!}{q_0! \cdots q_m!}$ for $\mathbf{q} \in Q_{m,d}$. The $\mathbf{q}$-th feature is $\Phi_{\mathbf{q}}(\mathbf{x}) = c_{\mathbf{q}} \mathbf{x}^{\mathbf{q}}$ with $c_{\mathbf{q}} = \sqrt{\binom{d}{\mathbf{q}}}$ or equivalently Eq. III.7. Then, for the polynomial kernel K and this choice of Φ, the following holds:

$$\begin{aligned}
K(\mathbf{x}, \mathbf{x}') &= \left(\sum_{i=0}^{m} x_i x_i' \right)^d && \text{(III.8)} \\
&= \sum_{\mathbf{q}} \frac{d!}{q_0! \cdots q_m!} (x_0 x_0')^{q_0} \cdots (x_m x_m')^{q_m} && \text{(III.9)} \\
&= \sum_{\mathbf{q}} \binom{d}{\mathbf{q}} x^{\mathbf{q}} x'^{\mathbf{q}} && \text{(III.10)} \\
&= \sum_{\mathbf{q}} \Phi(x^{\mathbf{q}}) \cdot \Phi(x'^{\mathbf{q}}). && \text{(III.11)}
\end{aligned}$$

Consider an example: a data set consisting of 2 variables and a polynomial kernel with a degree

of 2 results in 6 features. A vector in the data space, $\mathbf{x} = \langle x_1, x_2 \rangle$ maps to the following features

$$\Phi(\mathbf{x}) = \begin{pmatrix} c_{0,0}\ x_1^0 x_2^0 \\ c_{1,0}\ x_1^1 x_2^0 \\ c_{0,1}\ x_1^0 x_2^1 \\ c_{1,1}\ x_1^1 x_2^1 \\ c_{2,0}\ x_1^2 x_2^0 \\ c_{0,2}\ x_1^0 x_2^2 \end{pmatrix} = \begin{pmatrix} 1 & x_1^0 x_2^0 \\ \sqrt{2} & x_1^1 x_2^0 \\ \sqrt{2} & x_1^0 x_2^1 \\ \sqrt{2} & x_1^1 x_2^1 \\ 1 & x_1^2 x_2^0 \\ 1 & x_1^0 x_2^2 \end{pmatrix} \tag{III.12}$$

In our prototypical variable selection algorithm RFE, an SVM is trained on the data and then the "importance" of each variable for classification is scored (Guyon *et al.*, 2002b). The half of the variables corresponding to the smallest scores are eliminated (the number or percentage of variables eliminated vary in different implementations of this method). The process is repeated recursively with the remaining variables. Of the $log_2 |\mathbf{x}|$ SVMs models and corresponding variable sets produced this way, the one with the maximum prediction performance (e.g., accuracy) is selected.

In RFE the score of each variable x_i corresponds to the difference of the value of the objective function in (III.4) when all variables are included with the value of the objective function with x_i is removed from the data. Roughly, this is indicative of the difference between the separation margins between the classes with and without the inclusion of x_i. To allow for efficient computation of this difference, RFE assumes that the Lagrange multipliers a_i's do not change when removing a variable x_i and resolving the optimization problem. Under this assumption, the score of each variable x_i can be calculated as:

$$s_i = \frac{1}{2} \sum_{k=1,l=1}^{n} a_k a_l t_k t_l (K(\mathbf{x}_k, \mathbf{x}_l) - K(\mathbf{x}_k^{\setminus i}, \mathbf{x}_l^{\setminus i})) \tag{III.13}$$

where $\mathbf{x}_k^{\setminus i}$ denotes vector $\mathbf{x}_k$ with the i component removed. In fact, it can be shown that

$$s_i = \sum_{\mathbf{q}, q_i > 0} w_{\mathbf{q}}^2 \tag{III.14}$$

and for the polynomial kernel s_i *is the sum of the squares of the weights* $w_{\mathbf{q}}$ *of all constructed features* $\Phi_{\mathbf{q}}$ *where variable* x_i *appears in the product* $\prod_{v=1}^{m} x_v^{q_v}$ *with a non-zero degree* $q_i > 0$.

Let us now recall the example of Fig. III-1(a), considering that T is XOR of x_1 and x_2. The SVM will implicitly construct the feature $\Phi_{\mathbf{q}} = c_{\mathbf{q}} x_1 x_2$ with $\mathbf{q} = \langle 1, 1 \rangle$ that corresponds to the product of the variables. Notice that, since the variables take values in $\{-1, 1\}$, the product $x_1 x_2 = -T$. Thus, the feature $\Phi_{\mathbf{q}} = c_{\mathbf{q}} x_1 x_2$ is adequate to perfectly classify T and will be given a high weight w_q. This weight w_q will be included and increase the scores s_i's of both variables x_1 and x_2. Hopefully then, they will be highly ranked and returned by RFE.

Thus, depending on the kernel used, RFE is in principle able to identify parity functions (as shown in preliminary experiments of Guyon *et al.*, 2002b) and, in general, multivariate associations of variables with small or no pairwise association with T. Moreover, it performs this task efficiently (quadratically to the number of training examples n and linearly to the number of variables m) and without explicitly constructing all possible interaction terms, i.e., products of the original variables. In contrast, explicitly searching for a parity function of exactly d variables among m variables, would require checking all $\binom{m}{d}$ such terms.

Unfortunately, it has been shown theoretically that an SVM will not only assign non-zero weights to variables necessary for optimal classification, but it may also assign a non-zero weights to superfluous variables (Hardin *et al.*, 2004). In the terminology of Kohavi & John (1997), these are the weakly-relevant variables: informative for prediction, but superfluous to optimality. This helps explain why algorithms such as RFE, based on SVMs, tend to output numerous false positives, as shown in our experimental section.

A pictorial example is shown in Fig. III-2. In this example, there are two variables x_1 and x_2 while the class T of each sample is denoted with circles or squares. Variable x_2 is superfluous to perfect classification (weakly-relevant) but will still be given a weight arbitrarily close to that of x_1. In other words and in this example, the SVM cannot separate between strongly-relevant (absolutely necessary) variables and superfluous variables. Notice that, the SVM will still output the same result independently of whether there are exactly three training examples (samples) provided or millions

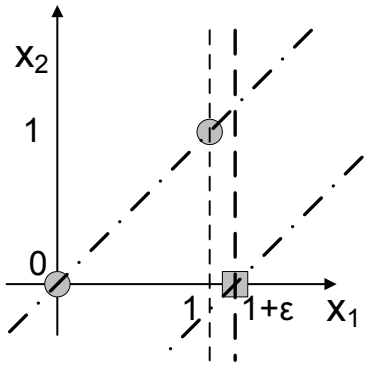


Figure III-2: An example where the weakly relevant variable x_2 receives a non-zero weight by the maximum margin classifier (dashed diagonal lines). The gap corresponding to the classifier that assigns a zero weight to x_2 (dashed vertical lines) can have an arbitrarily smaller gap.

of samples that all fall onto the depicted points of the graph. Approaches that use statistical tests however, such as HITON, would have been able to determine that with high statistical confidence, x_2 is superfluous, i.e., x_2 is independent of T given x_1 with sufficient sample and x_2 is removed from consideration.

This is not a contrived example but often occurs in practice. This pattern is observed in extensive experiments of Statnikov *et al.* (2006), the experimental results of presented here, and the following illustrative problem (Fig. III-3). Given a network consisting of 10 tiled copies of the ALARM network (from Tsamardinos *et al.*, 2006b), a target variable was selected randomly (shown as unfilled circle) and RFE was run with a polynomial kernel of degree 2. The variables selected by RFE have 4 true positives (shown as filled circles in the figure), 7 false positives (shown as triangles in the figure), and 1 false negative (shown as a square) which are scattered all over the network. HITON typically perfectly identifies the $MB(T)$ for the variables of this network. *Unlike the Markov Blanket-based methods, RFE and similar kernel-based algorithms provide no theoretical guarantees regarding their output.*

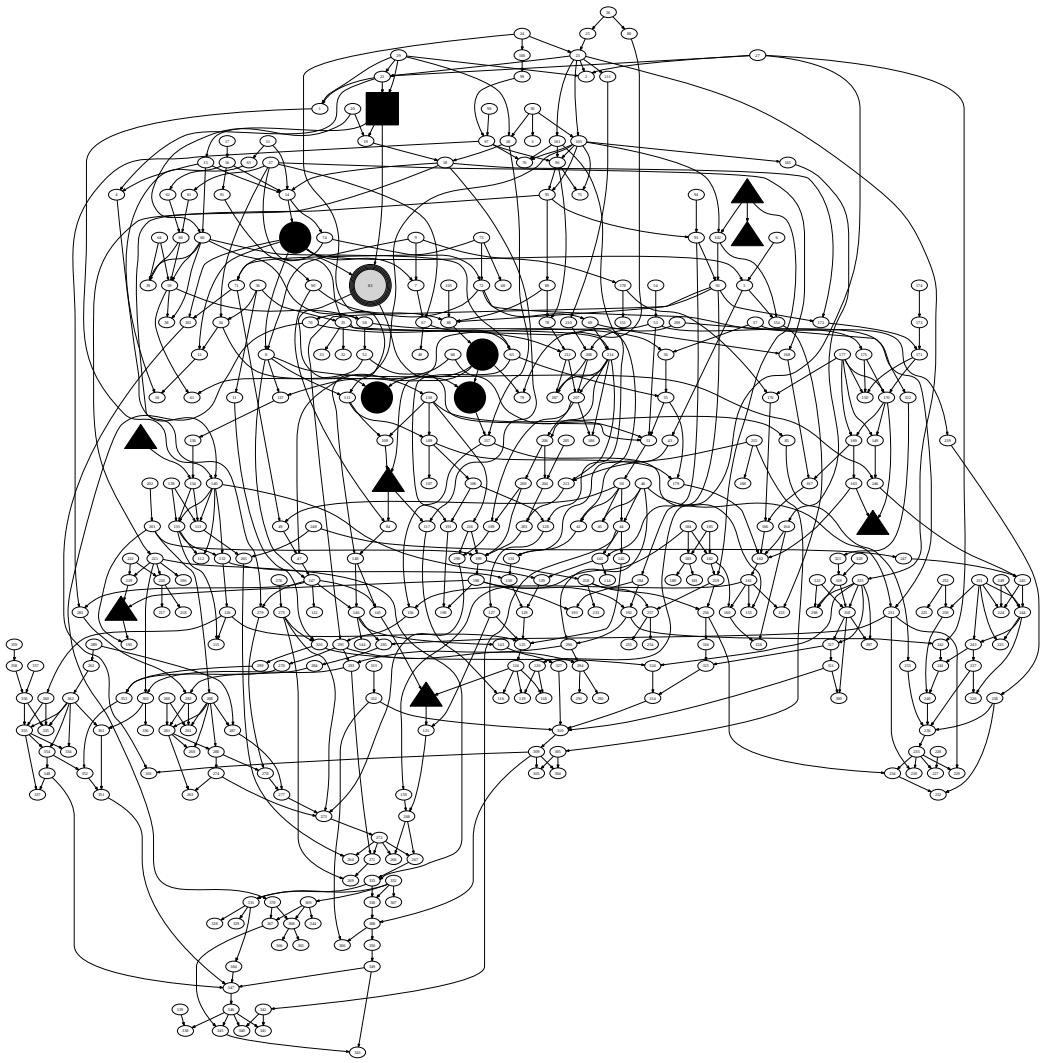


Figure III-3: An example network of 10 tiled copies of ALARM; a data set of 2500 instances was sampled from the distribution of the network. RFE was run (with a polynomial kernel of degree 2) on a target node selected randomly (unfilled circle) and returned 7 false positive (triangles) and 1 false negative (square), scattered all over the network. HITON typically perfectly identifies the $MB(T)$ on this network.

III.5 Related Work

Variable selection (also called feature selection in the literature) has received considerable attention during the last three decades both in statistics and in machine learning (Guyon & Elisseeff, 2003). Variable selection methods are typically divided into three classes: filters, wrappers, and embedded methods (Guyon & Elisseeff, 2003; Kohavi & John, 1997). In filter methods, variable selection can be viewed as a pre-processing step, that is independent of the learner A to produce a model. In the wrapper approach, the learner is included in the variable selection method; a search through the space of subset is executed with an objective function to guide the search is evaluated on the subsets of variables considered. Embedded methods incorporate the variable selection process directly in the learning method.

Filter Methods

Filter methods are thought of as a pre-processing step, that screens (or filters) extraneous variables. Two main classes of filter methods exist: variable ranking and Markov Blanket-based approaches. In variable ranking techniques, each variable is given some score (assume a high score indicates that the variable is useful for the learning task). Then, the scores can be ranked and a threshold used to select the top variables. Where the threshold can be selected by the user, have statistical (a standard 5% cutoff), be a hard count (select the top m variables), or other meaning.

Variable ranking methods can be separated as univariate or multivariate methods. A standard univariate approach is to calculate the pairwise association (or correlation) of each variable with the target value. Many different univariate measure may be employed depending on the data type and learning task. These methods include: signal-to-noise ratio, Fisher's criterion (Furey *et al.*, 2000), the T-test criterion (Tusher *et al.*, 2001; Hastie *et al.*, 2001), odds ratio (Mladenic & Grobelnik, 1999), and other information theoretic measures. The univariate methods will not perform well on the class of prototypical problems discussed in this paper, because the variables relationship to the target are in multivariate relationships with no (or little) univariate association.

Multivariate variable ranking methods use criterion that involve subsets of variables in the scoring function. A historical multivariate variable ranking method is the Relief algorithm (Kira & Rendell,

1992) and its many extensions (Kononenko, 1994; Florez-Lopez, 2002; Robnik-Sikonja & Kononenko, 2003; Guyon *et al.*, 2003; Gilad-Bachrachy *et al.*, 2004). In general, this method calculates a weight for each variable based on a randomized nearest neighbor algorithm (randomized in the sense that a random data instance is selected and used to update the weight vector using the nearest hit, closet sample from the same class, and nearest miss, closest sample from the opposite class). The Relief family of algorithms has been applied to the parity problem. However, its application is most often tested on small toy examples to prove its applicability (e.g., 5Parity + 5, a problem of 10 variables with 5 involved in a parity function; or a 3-parity example on 10 variables). Another historical variable ranking approach is the FOCUS algorithm, which recovers the minimum and sufficient subset of variables that is necessary to determine the labels for all training data (Allmuallim & Dietterich, 1994). This algorithm is highly sensitive to noise (a single misclassification can cause bogus results, Koller & Sahami, 1996). Other filtering methods are based on correlation measures (Hall, 2000; Yu & Liu, 2003); however, these methods may not work well on the prototypical problems of this Chapter because a single variable might not exhibit correlation with the target. The method of Zhao & Liu (2007) is designed to identify interacting variables (however, is not tested on synthetic examples to illustrate its properties beyond the Monks and Corral problems John *et al.*, 1994). The second class of filter methods is based on identifying the Markov Blanket (which is discussed in detail in Section III.2.

Wrapper Methods

In the wrapper methodology, a search through the space of subsets of variables is performed where the search is guided by an objective function applied to the output of a learner. Wrapper methods are classed according to their search methodology. For any non-trivial size problem, all subsets of variables can not be search therefore greedy heuristic strategies are employed namely: forward selection, backward selection, and general stepwise selection. Enumerable wrapper algorithms may be developed by selection of different search strategies, performance measures, and learners. The RFE algorithm discussed in detail in Section III.4 is an example of a wrapper method. Other methods

employing SVMs are the learner uses a variant of sensitivity analysis with the leave-one-out-error as the objective function to rank and eliminate variables (Rakotomamonjy, 2003).

Embedded Methods

Embedded methods incorporate the variable selection process directly into the learning procedure. Examples of such methods include: shrinkage regression methods, decision trees, and specific kernel method formulations. Shrinkage regression methods include ridge regression techniques (Hoerl & Kennard, 1970; Hastie *et al.*, 2001), methods combining shrinkage with variable selection (e.g., nonnegative garrote) (Breiman, 1995), least absolute shrinkage and selection operator (lasso) (Tibshirani, 1996; Efron *et al.*, 2004; Zou & Hastie, 2005), bridge regression (Frank & Friedman, 1993; Fu, 1998), sure independence screening (Fan & Lv, 2008), and elastic nets (Zou & Hastie, 2005). In decision trees, the "best" variables is chosen for the next node in the tree (information gain is often the criterion used to select the best variables). With pruning only a small subset of the possible variables may be included in the tree structure, consequently an implicit variable selection is performed. The classical criterion for selecting variables would have difficulties on the prototypical problems of this Chapter due to the multivariate associations (Tuv *et al.*, 2009).

Recently, several researcher have looked at alternative formulations of SVMs to implicitly perform variable selection. Often the l_0 or l_1 norm is employed (these norms concentrate weights on several sparse variables) (Bi *et al.*, 2003), or an approximation is formed by repeated application and scaling of the weights with a l_2 formulation (Weston *et al.*, 2003).

III.6 Feature Space Markov Blanket Algorithm

The Feature Space Markov Blanket (FSMB) algorithm is the first attempt to construct an algorithm combining the theoretical properties of the two approaches. The main idea of FSMB is to identify the Markov Blanket of T from *feature space* instead of in the original variable space, where multivariate associations become pairwise associations. FSMB employs a SVM to identify which features may have large pairwise association with T in feature space, so as to avoid considering all features.

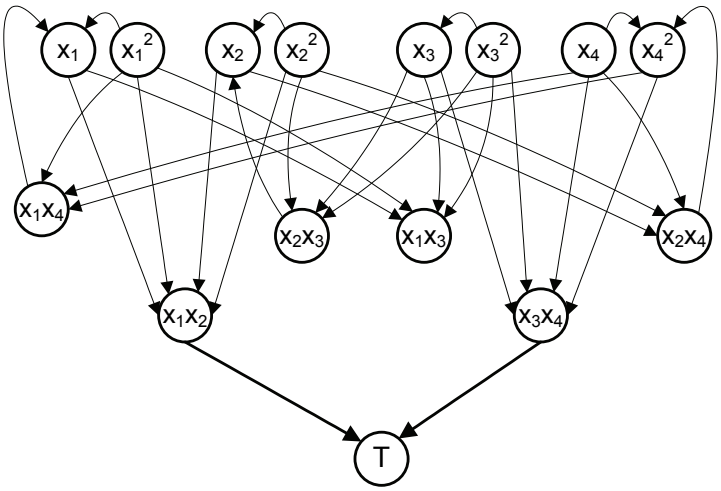


Figure III-4: The BN of feature space with the Fig. III-1(a) variables using a degree 2 polynomial kernel. T is the noisy XOR of x_1 and x_2, and x_3 is the noisy XOR of T and x_4. The two necessary features for perfect estimation of T, x_1x_2 and x_3x_4, appear as the parents of T and have a large pairwise association with T.

Example III.1. *Consider again the network in Fig. III-1(a), let us assume that T is the* noisy *XOR of x_1 and x_2 (i.e., $T = 1$ with probability 0.7 when x_1 and x_2 are different, and $T = 1$ with probability 0.3 otherwise). The same functional relation holds for x_3 and the pair T and x_4. Also assume the prior of x_1, x_2 and x_4 is 0.5.* No pair of variables has a non-zero association. *The pairs x_1, x_2 and x_3, x_4 have a multivariate association with T.* All variables are required for perfect estimation of the distribution of T. *Figure III-4 shows the BN in feature space of the original variables when a polynomial kernel of degree 2 is used, consisting of all products of the original variables of degree up to 2. The network was reconstructed using the MMHC algorithm (Tsamardinos* et al.*, 2006b). A Markov Blanket of T in feature space, denoted as $MB_\Phi(T)$ has as expected the features x_1x_2 and x_3x_4 containing all the variables of the original $MB(T)$.* □

The network in feature space is certainly not faithful because of the deterministic construction of the features: notice that feature x_1x_2 is independent of T given features x_1 and x_2 even though there is a direct edge between T and x_1x_2. Thus, we do not expect the $MB_\Phi(T)$ to necessarily be unique. Even though the distribution is still not faithful, HITON should be able to discover a

$MB_\Phi(T)$ since the features containing the original $MB(T)$ variables now have detectable pairwise associations with T.

In general, there is an exponential number of features to the number of variables. For FSMB to be scalable it should avoid explicitly computing all features. To this end, FSMB first trains an SVM model on the data that implicitly learns a weight vector $\mathbf{w}$. *A key assumption is that a low absolute weight* $|w_\mathbf{q}|$ *implies low association of the corresponding feature* $\Phi_\mathbf{q}$ *with the target. FSMB uses a heuristic method to compute the top* k *weights* $|w_\mathbf{q}|$ and subsequently, the corresponding features $\Phi_\mathbf{q}$ are calculated by (III.7) and passed to HITON. The algorithm is shown in Algorithm 4.

Algorithm 4

procedure FSMB($\mathcal{D} = \{\langle \mathbf{x}_i, t_i \rangle\}_{i=1}^n$)
// Train an SVM on the data and obtain the Lagrange multipliers $\mathbf{a}$ and Support Vectors $\mathbf{SV}$.
$\{\mathbf{a}, \mathbf{SV}\}$ = TrainSVM($\mathcal{D}$)
// Identify the largest magnitude weights, $|w_\mathbf{q}|$, and corresponding features, $\mathbf{q}$.
$\{\langle w_\mathbf{q}, \mathbf{q} \rangle\}$ = IdentifyTopWeights($\mathbf{SV}, \mathbf{a}, p, r$)
// Project the data onto the features identified see Eq. (III.7).
For each $\mathbf{q}$ and $\mathbf{x}_i$, $\Phi_\mathbf{q}(\mathbf{x}_i) = c_\mathbf{q} \prod_{v=1}^n \mathbf{x}_{i,v}^{\mathbf{q}_v}$
// Run HITON on the constructed features and obtain the $MB_F(T)$
$MB_F(T)$ = HITON($\{\langle \Phi_\mathbf{q}(\mathbf{x}_i), t_i \rangle\}_{i=1}^m$)
$MB(T)$ = All variables participating in some feature of $MB_F(T)$
return $\langle MB(T), MB_F(T) \rangle$
end procedure

Identify Top Weights Method. The heuristic method IdentifyTopWeights is a new polynomial algorithm that aims to identify the largest (in magnitude) weights of an SVM model (details and experimental results showing the ability of this algorithm to identify the top weights are presented in Chapter II, the main approach of the algorithm is summarized here). Without the heuristic, a brute force approach of constructing and sorting the entire weight vector is possible for small problems, but quickly grows intractable as the degree and/or number of variables increase. Also, notice that directly optimizing the quantity $|w_q| = |\sum_{k=1}^n a_k t_k c_q \prod_{v=1}^m x_{k,v}^{q_v}|$ seems to be a hard problem since the vector elements q_v are integers and the t_k can be negative, so the quantity is not a posynomial. Therefore, the heuristic approach avoids the expensive explicit construction of the weight vector by conducting a search for the top weights of the SVM model, guided by the norm of the weights summed over various subsets of features.

Recall, the quantity s_i of (III.13) corresponding to the sum of the squared weights of all features

containing variable x_i. Alternatively, the norm of the weight vector, can be partitioned into values $s_{i,j}$ corresponding to the sum of the squared weights of all features that contain variable x_i and are of degree j,

$$s_{j,i} = \binom{d}{j} \sum_{k,l=1}^{n} a_k a_l t_k t_l \left(H_j(\mathbf{x}_k, \mathbf{x}_l) - H_j(\mathbf{x}_k^{\setminus i}, \mathbf{x}_l^{\setminus i}) \right) \quad , \tag{III.15}$$

where $H_j(\mathbf{x}_k, \mathbf{x}_l) = (\mathbf{x}_k \cdot \mathbf{x}_l)^j$ is the homogeneous polynomial kernel of degree j. If a variable i participates in only one feature at degree level j with a non-zero weight $w_{\mathbf{q}}$ then $s_{i,j} = |w_{\mathbf{q}}|$. When a variable i participates in more than one feature at level j then $s_{i,j} > |w_{\mathbf{q}}|$. In either case, the quantity $s_{i,j}$ is an upper bound on the largest (in magnitude) weight for any feature involving variable i at level j. The search for the top weights uses these quantities as a guide to selectively calculate the weights of suspected top features.

The method IdentifyTopWeights takes as inputs the support vectors of the SVM model $\mathbf{SV}$, the alpha values $\boldsymbol{\alpha} = \mathbf{at}$, p - the number of features to construct, and r - the number of features to return. The method begins with the construction of the $d \times m$ *contributions matrix*, $\mathbf{B}$ where $\mathbf{B}_{j,i} = s_{j,i}$ for $j = 1, \ldots, d$ and $i = 1, \ldots, m$. After the calculation of the initial contributions matrix, $\mathbf{B}$, the heuristic search loops through the following 3 sub-procedures: (1) select the next level and variable(s) to focus construction, (2) explicitly construct the features and calculate their weights for the selected variables and level, and (3) update the bounds of the contribution matrix. Once the search procedure constructs p features, the features are sorted by their absolute weight and the top r features are returned.

The first sub-procedure selects the next level and variables to focus construction. The method normalizes the contributions matrix by the number of features in the sum of each cell of the matrix. After normalization the selection function returns the level with the maximum normalized value, l, and a sorted variable list for that level $\mathbf{v}$.

The next sub-procedure explicitly constructs the features and weights selected in the previous step. The construction of new features always includes all combinations with any variables already used to construct features at the level under consideration. Initially, there are no such variables. As the algorithm proceeds, the features to be constructed consist of all combination of variables

already selected at the level l and the next highest ranked variable in $\mathbf{v}$ that has not previously been selected. Once the new feature(s) are determined, then the weights are calculated.

The final sub-procedure of the loop involves updating the bounds on the top weight, that is updating the contributions matrix. For example, if the feature $x_1 x_3^2$ was constructed then its feature index $\mathbf{q} = \langle 102 \rangle$ and weight $w_{\mathbf{q}}$ is calculated and the contributions matrix is updated: $\mathbf{B}_{j,i} = \mathbf{B}_{j,i} - w^2_{\langle 102 \rangle}$ where $j = 3$, the degree of the feature, and $i = 1\ textand\ 3$, the variables in the feature.

The search loop continues until p features and weights have been computed. This list is sorted by the magnitude of the weights. The top r weights and corresponding feature index vectors are returned. Overall, the calculations of the top p weights via the heuristic method is $O(dms^2 + sdp)$, with s support vectors, m variables, d degree of the kernel, and p features to construct.

III.7 Experimental Evaluation

The new method, FSMB, was evaluated against other variable selection methods: the two main approaches that influenced its design, HITON and RFE, and the Relief method, which is capable of selecting variables in the difficult distributions discussed. All four methods were implemented in Matlab. The implementation of HITON follows that used in Aliferis *et al.* (2003a, 2009a,b) (HITON-MB and HITON-PC are available in the Causal Explorer library, Aliferis *et al.*, 2003b; Statnikov *et al.*, 2009). The RFE implementation was also used in Aliferis *et al.* (2003a, 2009a,b). Relief was implemented (from the CLOP toolkit, Guyon *et al.*, 2009) with a backwards wrapper. The SVMs were trained using LibSVM software (Chang & Lin, 2001).

Parameter selection for the given algorithms followed generally accepted choices. The statistical test for independence used in both HITON and FSMB was G^2 test of Fisher's z test using a 5% threshold and maximum conditioning set of 3. RFE was run reducing the variables considered by a half each iteration and selecting the subset with the maximum performance score on a test set (AUC measured performance on a test split of 20% of the data). [5]. Relief was run with $k = 5$ neighbors ($k = 1$ neighbors was also run on limited problems, but found to have worse results and

[5] Alternative parameterizations of RFE were considered, i.e., reducing the variables by 20% each iteration, removing one variable iteration; however, this parameters selected in general had the best and most efficient results

was discontinued). FSMB was run searching for the top 20,000 features (p=20,000) then the top 10, 25, 50, or 1000 features were returned and used to construct feature data. The reported results are for the case of 50 features (there is often no difference between considering the top 10, 25, or 50 features, upwards of that some additional features may not always be removed as begin related to the target). The methods are first compared on simulated data sets where the true Markov Blanket is known and different difficult distributions can be explored. Finally, the methods were evaluated on several large, real world data sets.

III.7.1 Simulated Data

The four methods were evaluated on several simulated data sets. For each problem, FSMB and RFE use SVM models; a polynomial kernel was used and its parameters selected with a high cost C-value and the degree set to match the problem (that is, for the double noisy-xor problem a degree 2 kernel was selected, the redundant mechanism example a degree 2 kernel was used, and the noisy 3-parity problem a degree 3 kernel was used).

The metrics used to compare the methods are the sensitivity and specificity in identifying the true Markov Blanket. An additional metric, the distance from the true Markov Blanket, was calculated that combines the sensitivity and specificity as $d = \sqrt{(1 - \text{sensitivity})^2 + (1 - \text{specificity})^2}$. When the algorithms identify the Markov Blanket and only the Markov Blanket, then the sensitivity and specificity should be 1.000 and the distance measure should be 0.000. For each problem, the methods were run for 10 data samplings and the mean and standard deviation of the metrics are presented for each method.

Double Noisy-XOR:

The first example is the 5-variable network described in Example 1 and Figure III-1(a). Recall in this example, there are two noisy XOR relationships with the probability distribution set to be such that there is no detectable pairwise association with the target. The true Markov Blanket of the target, T, is all other variables: $\text{MB}(T) = \{x_1, x_2, x_3, x_4\}$. The total number of features in the problem is 15 therefore, FSMB converts the data for all features before running HITON.

The results on this prototype problem are summarized here with a detailed table of the results in Appendix B.I. For this problem the specificity is always undefined since every variable is included in the Markov Blanket. Therefore, values of 1.00 were entered for each method. HITON rarely identified any of the variables in the Markov Blanket (sensitivity values of 0.05 and 0.125). This results is expected because any detectable pairwise associations between the variables and target is due to chance in the data sampling. In general, Relief is able to identify the Markov Blanket (sensitivity values of 0.875, 1.000, and 0.975 with increasing sample). RFE was also able to identify all four members of the Markov Blanket (sensitivity values of 0.80, 0.95, 0.975 with increasing sample). FSMB can be thought of as running HITON on the distribution of the network shown in Fig. III-4. The method generally returns two features x_1x_2 and x_3x_4 involving all variables of the Markov Blanket. For the larger sample sizes, FSMB correctly identifies all four variables reliably (at the lowest sample size it occasionally will miss a feature, but performs similarly to the other methods: sensitivity of 0.900).

Embedded Double Noisy-XOR:

The second example embedded the 5 variable Double Noisy-XOR network of Fig. III-1(a) in a larger network for a total of 153 variables (including the target variable), as shown in Fig. III-5. The embedding was such that all dependencies and independencies between the 5 variables were maintained (Tsamardinos *et al.*, 2006a). FSMB converted the top 100 weighted features into feature data on which to run HITON (out of the 11780 total number of features). The variables returned by each algorithm were compared to the true $MB(T) = \{x_1, x_2, x_3, x_4\}$.

The mean and standard deviation of the sensitivity, specificity, and distance measure is reported for the four algorithms in Table III-1. HITON rarely includes the correct members of the Markov Blanket as shown by the low sensitivities. RFE occasionally misses a member of the Markov Blanket and often includes a number of false positives. Relief identifies the Markov Blanket correctly for the larger sample sizes (as does FSMB), but introduces slightly more false positives than FSMB. These results are also presented in Figure III-6. The distance measure is plotted versus increasing sample size for the different algorithms.

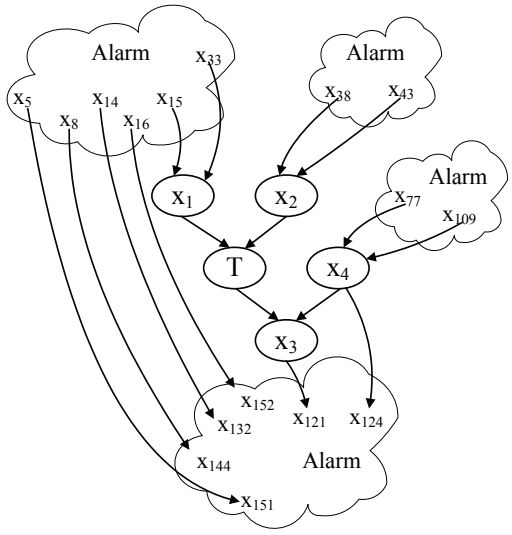


Figure III-5: The BN for the Embedded Double Noisy-XOR data set, where the network of Fig. III-1(a) is embedded into a larger network. The subgraphs with label *Alarm* are not shown in detail and are copies of the ALARM network (Beinlich *et al.*, 1989).

Table III-1: Results on Embedded Double-XOR Problem of 153 variables.The sensitivity, specificity, and distance metric for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3, x_4\}$) by each algorithm on the network of Fig. III-5. In this network, all parent-child relationships involving T are noisy-XOR. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Data		Sensitivity		
Size	HITON	Relief	RFE	FSMB
100	0.025 ± 0.08	0.450 ± 0.33	0.725 ± 0.32	0.400 ± 0.32
500	0.075 ± 0.12	1.000 ± 0.00	0.875 ± 0.21	1.000 ± 0.00
1000	0.100 ± 0.13	1.000 ± 0.00	0.925 ± 0.17	1.000 ± 0.00
Data		Specificity		
Size	HITON	Relief	RFE	FSMB
100	0.792 ± 0.12	0.811 ± 0.18	0.984 ± 0.01	0.960 ± 0.02
500	0.959 ± 0.04	0.963 ± 0.04	0.932 ± 0.15	0.991 ± 0.01
1000	0.980 ± 0.01	0.976 ± 0.02	0.972 ± 0.01	0.986 ± 0.01
Data		Distance		
Size	HITON	Relief	RFE	FSMB
100	1.003 ± 0.09	0.627 ± 0.29	0.285 ± 0.31	0.605 ± 0.31
500	0.927 ± 0.12	0.037 ± 0.04	0.191 ± 0.22	0.009 ± 0.01
1000	0.900 ± 0.13	0.024 ± 0.02	0.099 ± 0.16	0.014 ± 0.01

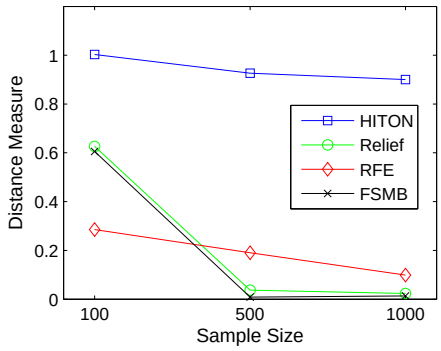


Figure III-6: Results on Embedded Double-XOR Problem. This figure is plotting the distance metrics for each algorithm versus increasing sample size. A distance measure of zero indicates perfect identification of the Markov Blanket. At the smallest sample size, RFE performs the best; while for the larger sample sizes Relief and FSMB perform well, with FSMB having slightly fewer false positives.

At this point, we would like to make the following observation: *the embedded double noisy XOR example is harder than typical toy examples in the literature.* A common practice is to try new variable selection algorithms on toy problems that contain a set of necessary-for-optimality variables and a set of completely independent-to-the-target variables. Usually, these sets are named the "relevant" and the "irrelevant" variables. However, most real data sets contain informative but superfluous variables (weakly-relevant). *This is evident by the fact that most BNs reconstructed from real data are connected.* Most algorithms are favored when there are no superfluous variables, such as RFE as shown in Statnikov *et al.* (2006). In our example, all variables not in the $MB(T)$ are not irrelevant, but superfluous given $MB(T)$.

Redundant Mechanism:

The four methods were run on the 3-variable network pictured in Figure III-1(b), with a redundant mechanism distribution with no pairwise association between x_1 and the target. The true Markov Blanket of the target, T, is all other variables: $\text{MB}(T) = \{x_1, x_2\}$. The total number of features in the problem is 6 therefore, FSMB converts the data for all features and then runs HITON. Mean and standard deviation of the sensitivity, specificity, and distance results are summarized here and detailed table of results in Appendix B.II. For this problem the sample sizes used increased to

500, 1000, 1500, and 2000 all methods did not perform well at the lowest sample size considered, 100 samples).

For this problem, the specificity is again undefined since every variable is included in the Markov Blanket. Therefore, values of 1.00 were entered for each method. HITON rarely identified the members of the Markov Blanket (sensitivity values of 0.00 to 0.40). Relief is able to identify the MB in general (sensitivity values of 0.70-0.95). RFE was also able to identify the MB in general with sensitivity values of 0.65 - 0.95. FSMB is able to identify the MB perfectly for all but the lowest sample size (sensitivity of 0.70 for 500 samples, and 1.00 for larger sample sizes).

Embedded Redundant Mechanism:

This example embedded the 3 variable network of Fig. III-1(b) in a larger network of increasing sizes (the total number of variables goes from 262 to 447 variables, created with tiling 7 and 12 copies of the Alarm network). FSMB constructed a feature data set on the top 50 weighted features (out of the 25878 and 100576 total number of features). The variables returned by each method were compared to the true $MB(T)\{x_1, x_2\}$. The mean and standard deviation of the sensitivity, specificity, and distance results are presented in Table III-2 and III-3. Note, in this example larger sample sizes were used because all methods did not perform well at the lowest sample sizes. For FSMB at the lower sample sizes the SVM model does not give the features of interest a large weight, therefore the algorithm will not select the feature of interest to convert to feature data.

On this problem, HITON has a low sensitivity and high specificity because it rarely finds the true positives, but includes very few false positive results. Both Relief and RFE have both sensitivities and specificities ranging from 0.50 - 0.80. This indicates the methods occasionally miss the true members of the Markov Blanket but may also return many false positives. The range of variables selected by either method goes from 2 variables to the set of all possible variables. FSMB has quite high sensitivity and specificities (resulting in the lowest distance measures). For the smaller problem size with the two larger sample sizes and the larger problem size with the largest sample sizes, FSMB has perfect sensitivity.

The distance measure for each algorithm is plotted versus increasing sample sizes in Figure III-7.

Table III-2: Results on Embedded Redundant Mechanism Problem of 262 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2\}$) for each algorithm on the network of Fig. III-1(b) embedded into a larger network of 262 total variables (similar to Fig. III-5). The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 262

Data Size	Sensitivity HITON	Relief	RFE	FSMB
1000	0.200 ± 0.42	0.750 ± 0.35	0.750 ± 0.42	0.850 ± 0.34
1500	0.200 ± 0.42	0.800 ± 0.42	0.750 ± 0.42	1.000 ± 0.00
2000	0.300 ± 0.48	0.500 ± 0.47	0.550 ± 0.50	1.000 ± 0.00
Data Size	Specificity HITON	Relief	RFE	FSMB
1000	0.989 ± 0.00	0.525 ± 0.47	0.729 ± 0.31	0.984 ± 0.00
1500	0.988 ± 0.00	0.707 ± 0.33	0.610 ± 0.45	0.990 ± 0.00
2000	0.988 ± 0.00	0.796 ± 0.32	0.800 ± 0.32	0.988 ± 0.01
Data Size	Distance HITON	Relief	RFE	FSMB
1000	0.801 ± 0.42	0.706 ± 0.34	0.512 ± 0.37	0.162 ± 0.33
1500	0.802 ± 0.42	0.492 ± 0.39	0.630 ± 0.42	0.010 ± 0.00
2000	0.703 ± 0.48	0.678 ± 0.37	0.626 ± 0.43	0.012 ± 0.01

The different colored lines illustrate the results for the different methods ("blue" - HITON, "green" - Relief, "red" - RFE, "black" - FSMB). The plot on the left is for the smaller problem (226 variables). The plot on the right is for the larger problem (447 variables). The figure confirms FSMB superiority to other methods on this problem and the general trend of the method increases its performance as sample size increases.

Noisy 3-Parity:

The data for this classification problem is determined by a noisy 3-input parity function. The parity function is a generalization of the XOR function and is difficult to detect the multivariate relationship. The noisy portion of the function allows for on average 30% of the target values to be switched. The margin probabilities of the true causes of the target were set such that there is no detectable pairwise association to the target. The data was sampled from a uniform binary distribution taking values of $\{-1, +1\}$. With problems of sizes of 60, 80, and 100 variables, FSMB

Table III-3: Results on Embedded Redundant Mechanism Problem of 447 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2\}$) for each algorithm on the network of Fig. III-1(b) embedded into a larger network of 447 total variables (similar to Fig. III-5). The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 447

Data Size	Sensitivity HITON	Relief	RFE	FSMB
1000	0.200 ± 0.42	0.750 ± 0.35	0.750 ± 0.42	0.900 ± 0.32
1500	0.200 ± 0.42	0.800 ± 0.42	0.750 ± 0.42	0.950 ± 0.16
2000	0.300 ± 0.48	0.500 ± 0.47	0.550 ± 0.50	1.000 ± 0.00
Data Size	Specificity HITON	Relief	RFE	FSMB
1000	0.989 ± 0.00	0.525 ± 0.47	0.729 ± 0.31	0.976 ± 0.01
1500	0.988 ± 0.00	0.707 ± 0.33	0.610 ± 0.45	0.982 ± 0.00
2000	0.988 ± 0.00	0.796 ± 0.32	0.800 ± 0.32	0.985 ± 0.00
Data Size	Distance HITON	Relief	RFE	FSMB
1000	0.801 ± 0.42	0.706 ± 0.34	0.512 ± 0.37	0.121 ± 0.31
1500	0.802 ± 0.42	0.492 ± 0.39	0.630 ± 0.42	0.066 ± 0.15
2000	0.703 ± 0.48	0.678 ± 0.37	0.626 ± 0.43	0.015 ± 0.00

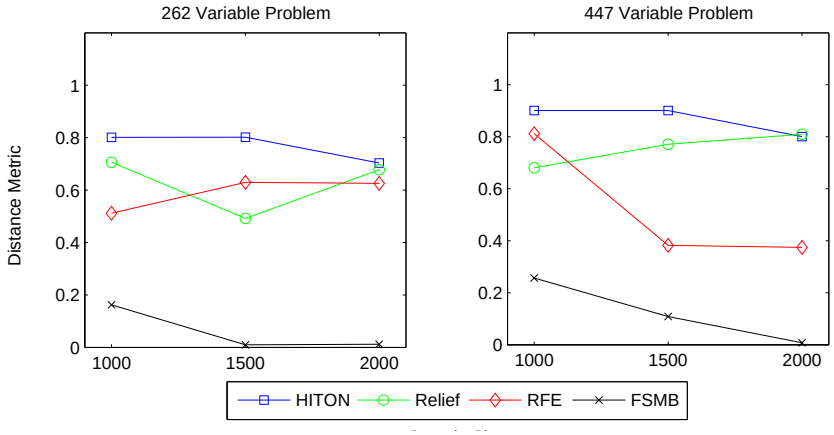


Figure III-7: Results on Embedded Redundant Mechanism Problems. This figure is plotting the distance metrics for each algorithm versus increasing sample size. The figure on the left is for the smaller problem (226 variables), the figure on the right is for the larger problem (447 variables). FSMB outperforms all algorithms for all sample sizes and problem size.

Table III-4: The Results on Noisy 3-Parity Problem of 60 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 60 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 60				
Data	Sensitivity			
Size	HITON	Relief	RFE	FSMB
100	0.000 ± 0.00	0.500 ± 0.39	0.367 ± 0.40	0.700 ± 0.43
500	0.033 ± 0.11	0.500 ± 0.45	0.633 ± 0.48	1.000 ± 0.00
1000	0.000 ± 0.00	0.767 ± 0.42	1.000 ± 0.00	1.000 ± 0.00
Data	Specificity			
Size	HITON	Relief	RFE	FSMB
100	0.975 ± 0.01	0.537 ± 0.40	0.709 ± 0.38	0.712 ± 0.03
500	0.968 ± 0.02	0.811 ± 0.21	0.802 ± 0.30	0.782 ± 0.04
1000	0.977 ± 0.01	0.718 ± 0.39	0.863 ± 0.31	0.775 ± 0.04
Data	Distance			
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	0.850 ± 0.17	0.847 ± 0.22	0.502 ± 0.31
500	0.967 ± 0.10	0.587 ± 0.43	0.504 ± 0.48	0.218 ± 0.04
1000	1.000 ± 0.00	0.472 ± 0.47	0.137 ± 0.31	0.225 ± 0.04

heuristically selected the top 100 features to then process with HITON (out of a total of 39711, 91881, and 176851 features).

The variables returned by each algorithm were compared to the true $MB(T) = \{x_1, x_2, x_3\}$. The mean and standard deviation of the sensitivity, specificity, and distance value for identifying the Markov Blanket are reported in Table III-4, III-5, and III-6. HITON again did poorly on this data set; most often missing all members of the Markov Blanket. Relief occasionally misses members of the Markov Blanket and includes additional variables in the set identified. RFE performs well for the smallest problem and largest sample size (in one class outperforming FSMB). However, for the larger problems RFE begins identifying more false positives and eventually misses the true members of the Markov Blanket. FSMB performs best for all but the smallest problem with the largest sample size.

To help visualize the trends in the results, the distance measure for each algorithm is plotted versus increasing sample sizes in Figure III-8. The three plots are for the three problems sizes increasing from left to right: 60, 80, and 100 variables. The colored lines indicate the method: blue

Table III-5: The Results on Noisy 3-Parity Problem of 80 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 80 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 80				
Data	Sensitivity			
Size	HITON	Relief	RFE	FSMB
100	0.000 ± 0.00	0.467 ± 0.36	0.433 ± 0.45	0.633 ± 0.37
500	0.000 ± 0.00	0.367 ± 0.43	0.500 ± 0.42	0.933 ± 0.21
1000	0.000 ± 0.00	0.533 ± 0.45	1.000 ± 0.00	1.000 ± 0.00
Data	Specificity			
Size	HITON	Relief	RFE	FSMB
100	0.978 ± 0.01	0.648 ± 0.37	0.643 ± 0.38	0.782 ± 0.03
500	0.974 ± 0.01	0.794 ± 0.18	0.697 ± 0.32	0.809 ± 0.04
1000	0.983 ± 0.01	0.787 ± 0.29	0.787 ± 0.41	0.810 ± 0.03
Data	Distance			
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	0.769 ± 0.25	0.846 ± 0.21	0.479 ± 0.29
500	1.000 ± 0.00	0.702 ± 0.40	0.686 ± 0.37	0.236 ± 0.17
1000	1.000 ± 0.00	0.592 ± 0.44	0.213 ± 0.41	0.190 ± 0.03

Table III-6: The Results on Noisy 3-Parity Problem of 100 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 100 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 100				
Data	Sensitivity			
Size	HITON	Relief	RFE	FSMB
100	0.000 ± 0.00	0.533 ± 0.42	0.233 ± 0.42	0.367 ± 0.40
500	0.000 ± 0.00	0.433 ± 0.39	0.367 ± 0.46	0.800 ± 0.32
1000	0.000 ± 0.00	0.500 ± 0.48	0.733 ± 0.38	0.933 ± 0.21
Data	Specificity			
Size	HITON	Relief	RFE	FSMB
100	0.980 ± 0.01	0.567 ± 0.43	0.701 ± 0.39	0.825 ± 0.03
500	0.979 ± 0.01	0.670 ± 0.38	0.762 ± 0.40	0.860 ± 0.03
1000	0.986 ± 0.01	0.828 ± 0.30	0.799 ± 0.32	0.837 ± 0.02
Data	Distance			
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	0.822 ± 0.24	0.985 ± 0.05	0.684 ± 0.35
500	1.000 ± 0.00	0.806 ± 0.22	0.836 ± 0.32	0.296 ± 0.27
1000	1.000 ± 0.00	0.619 ± 0.45	0.410 ± 0.43	0.212 ± 0.17

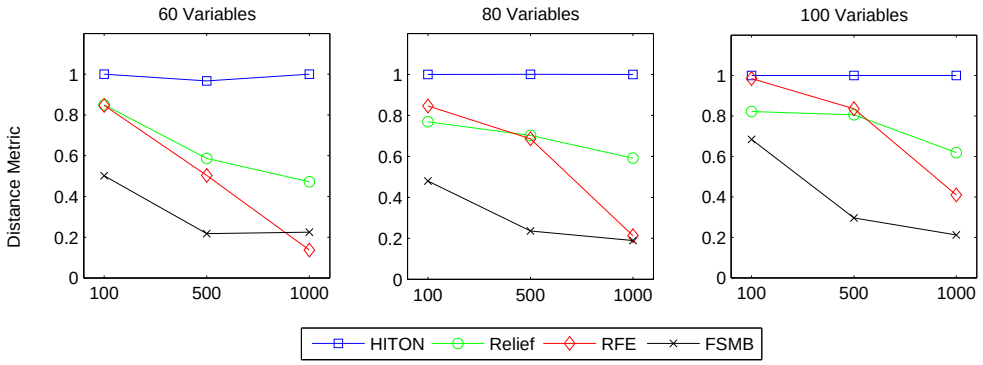


Figure III-8: Noisy 3-Parity Results Summary. This figure is plotting the distance metrics for each algorithm versus increasing sample size. The three plots are for the three problems sizes increasing from left to right: 60, 80, and 100 variables. The colored lines indicate the method: blue - HITON, green - Relief, red - RFE, and black - FSMB.

- HITON, green - Relief, red - RFE, and black - FSMB. The figure shows FSMB general superiority to the other methods. Only for the largest sample size and smaller problem sizes does RFE beat or compete with FSMB.

The same synthetic problem was also run with the noise for the parity function set to 20%. The results for this analysis are presented in Tables B-3, B-4, B-5 in the Appendix B.III. On this problem, HITON performs porrly. Relief and RFE do as well or better than FSMB on the smaller problems with large sample. As the problem size gets larger, FSMB begins to meet and then exceed the other methods performance. Once again, the distance metrics are plotted and presented in Figure B-1.

High-Dimensional Noisy 3-Parity:

A final experiment was performed with the noisy 3-parity problem to show that our method scales to higher-dimensional problems. We ran FSMB on a noisy 3-parity problem of 1000 variables and 5000 samples. The problem has over 10^8 features to search among. FSMB was run on three data samplings and took on average 31 minutes. Compare this to a brute force approach if 1000 test/s were possible, then to test all features would take approximately 2 days time. For this problem, FSMB is able to detect and return the interaction feature of interest with very few 3-8 extra variables returned as false positives.

Table III-7: Summary of Variables and Features Identified. The number of variables selected by each method where FSMB also reports the number of features returned. The number is averaged across the ten samplings and sample sizes for each simulated problem.

	Number of Variables				Num. of Feats
Problem	HITON	Relief	RFE	FSMB	FSMB
Small Double XOR	0.30	3.80	3.63	3.87	2.03
Small Redundant Mechanism	0.40	1.63	1.63	1.85	1.06
Embedded Double XOR	13.57	15.57	8.87	6.27	3.57
Embedded Redundant Mechanism					
226 Variables	3.57	85.23	75.67	5.17	2.83
447 Variables	3.57	107.53	94.77	5.87	3.20
Noisy 3-Parity					
60 Variables	1.33	19.53	13.90	16.57	6.57
80 Variables	1.67	21.17	24.33	17.93	6.80
100 Variables	1.77	31.70	25.20	17.57	6.87

Summary of Variables Selected

A final summary is given in Table III-7 reporting the number of variables identified by each method. Additionally, for FSMB the number of features found is reported. These numbers are averaged over the ten data samplings and sample sizes for each problem. For the Small Double-XOR and Small Redundant Mechanism problem, the optimal number of variables to return is 4 and 2 respectively. For the other problems, the number of variables returned often exceeds the optimal number of variables (2-4). HITON often returns the fewest variables, but that is expected since the problems are designed so that the variables have no association with the target. Relief often returns the greatest number of variables. RFE generally returns fewer variables than Relief, but often more than FSMB. FSMB returns the fewest number of variables (excluding HITON) while selecting the optimal variables (as illustrated by the tables above). The number of features FSMB returns is also reported for each problem.

Timing Results

The focus of this chapter is on the development of a new algorithm for variable selection that works on several difficult problems that other techniques fail or produce sub-optimal results. The emphasis is on the quality of the performance of the new method - FSMB, but a short presentation on the efficiency of the method is now presented. In terms of execution time, FSMB was the slowest

algorithm presented. Table III-8 summarizes the execution time on the simulated problems. First, notice the time for HITON on the embedded redundant mechanism and noisy 3-parity problems is extremely low. This results is because the first step of HITON is to filter only those variables associated with the target. These two problems were designed to exploit this property and cause HITON to fail. So while the method completes very quickly, the results are equally as poor. The times for Relief and RFE are in the middle with RFE taking longer (the scoring function for RFE requires doing computations with the kernel that take longer). Note, that the timing results of these methods would be affected by how the wrapper was designed. Recall, these methods eliminate half of the variables each iteration; if the variable reduction was lessened to 20% or only one variable at a time, then the methods computation time would be increased. For FSMB, the algorithm spends the majority of its time searching for the top-weighted features. The parameters selected for the methods were meant to be generous to ensure catching all top features (that is, FSMB was run with the parameter set to construct the top 20,000 features, from this the top 50 were then passed to HITON). The time complexity of the searching portion of FSMB is $\Theta(dms^2 + sdp)$, where m is the number of variables, d the degree of the kernel, s the number of support vectors, and p the number of features the algorithm is allowed to search (see Chapter II.3.3 for detailed analysis on this result). It is important to note, that even for greatly increased scales of problems FSMB does not become intractable. For example, the Thrombin data set (discussed in the next section) has over 139,000 variables and 2,000 samples is analyzed by FSMB in approximately 2 hours.

III.7.2 Real World Data

In addition to the simulated data analysis, comparisons of the methods were performed on several diverse, real world data sets. The evaluation will first look at the FSMB, RFE, and HITON algorithms on several data sets. Several of these data sets were analyzed previously for many variable selection methods Aliferis *et al.* (2003a, 2009a,b). The evaluation was constructed following these previous analysis in order to compare the results of various variable selection methods.

All of the real-world data sets and their characteristics are listed in Table III-9. These sets cover

Table III-8: Summary of Execution Time on Simulated Problems. The execution time (in seconds) of the variable selection methods on different simulated problems. The time reported is the mean over the ten samplings and sample sizes.

Execution Time (seconds)				
Problem	HITON	Relief	RFE	FSMB
Embedded Double XOR	70.6	3.5	12.7	121.5
Embedded Redundant Mechanism				
226 Variables	1.9	31.5	107.8	626.5
447 Variables	3.8	37.8	186.9	785.2
Noisy 3-Parity				
60 Variables	0.3	4.9	19.0	363.2
80 Variables	0.3	10.2	35.7	499.6
100 Variables	0.4	17.3	48.0	397.5

many different domains and data types. The first data set is on the diagnosis of lung cancer from oligonucleotide gene expression array data, specifically determining squamous versus adenocarcinoma types of cancer (Bhattacharjee *et al.*, 2001). The second data set is on splice site prediction, that is the identification of splice sites from a genomic sequence (Saeys *et al.*, 2003). The next task was on prediction of infant mortality within one year from clinical values (Mani & Cooper, 1999). This is followed by a task of text categorization; text (Medline) documents from the OHSUMED corpus (version from Joachims, 2002) are labeled relevant or non-relevant to neonatal disease (Hersh *et al.*, 1994). Also included were data sets from different public challenges (NIPS 2003 Challenge, 2003; WCCI 2006 Challenge, 2006): Gisette, Sylva, Hiva, and Thrombin. The domain of each of these data sets is digit recognition, ecology, and two on drug discovery respectively. Where the final data set is for the classification of whether biomolecules are able (or not) to find to thrombin (KDD Cup 2001, 2001).

For the real data sets, the true *MB* is not available to assess the quality of each variables selection method. Therefore, two metrics were used to assess variable selection methods. First, the number of variables returned by the method (listed as an absolute number or percentage of variables returned) was reported. Also, the classification performance measured by the area under the receiver operator characteristic curve (AUC) was presented.

Table III-9: Characteristics of Real Data sets

Data Set Name	Problem Domain	Number Vars.	Number Samples	Evaluation Design
Lung Cancer	Genomic	12,600	160	5-fold c.v.
Splice Site	Genetic	400	2000	10-fold c.v.
Infant Mortality	Clinical	86	5337	1-fold c.v.
OHSUMED	Text Cat.	14,373	5000	1-fold c.v.
Gisette	Digit Rec.	5000	7000	1-fold c.v.
Sylva	Ecology	216	14394	1-fold c.v.
Hiva	Drug Disc.	1617	4229	1-fold c.v.
Thrombin	Drug Disc.	139,351	2543	1-fold c.v.

Performance of RFE, HITON, and FSMB

The first two data sets analyzed were the Lung Cancer and Splice Site data. These data sets were also discussed and used in the evaluation of the IdentifyTopWeights method of Chapter II. Results on the next six data sets using several variable selection methods were reported in Aliferis *et al.* (2009a,b). The data were split following the same cross validation design exactly as in *ibid.* A nested stratified cross validation design was employed: in the outer loop the performance estimate was calculated for the optimal model, in the inner loop the choice of parameter and variables subsets was selected. The best parameters of the SVM classifier were selected from the sets $d = \{1, 2, 3, 4\}$ and $C = 10^i, i = \{-8, \ldots, 3\}$. For FSMB, the SVM model with all variables and trained with the optimal parameters is used by the FSMB algorithm to heuristically identify the top 1000 features (100 features for Thrombin). The top features were passed to HITON in order to identify the $MB_{\Phi}(T)$. The results comparing the three main methods: RFE, HITON, and FSMB are presented here.

The results of the variable selection methods are presented in Table III-10. The classification performance and number of variables selected is given for each method. For FSMB, the number of features is also presented. A second table (Table III-11) illustrates the simplicity of the FSMB results. The number of features FSMB using in its classifier model is given alongside the number of features used to evaluate the other methods. For these methods, the number of features, $\binom{v+d}{d}$, is

Table III-10: Results on Real World Data: The classification performance for model built on all variables and the subsets selected by the three variable selection method is given. In addition, the number of variables selected is presented for each data set.

Data Set	Evaluation Metric	Variable Selection Method None	RFE	HITON	FSMB
Lung Cancer	Num. Vars/Feats	12,600	19	16	4 / 2
	AUC	0.991	0.986	0.978	0.979
Splice Site	Num. Vars/Feats	400	400	26	10 / 21
	AUC	0.982	0.982	0.926	0.961
Infant Mortality	Num. Vars/Feats	86	5	7	15 /37
	AUC	0.820	0.748	0.865	0.823
OHSUMED	Num. Vars/Feats	14,373	112	34	40 / 43
	AUC	0.905	0.807	0.829	0.811
Gisette	Num. Vars/Feats	5,000	625	226	48 / 38
	AUC	0.997	0.998	0.997	0.994
Sylva	Num. Vars/Feats	216	27	50	29 / 41
	AUC	0.998	0.998	0.997	0.997
Hiva	Num. Vars/Feats	1617	51	8	14 / 15
	AUC	0.717	0.640	0.527	0.702
Thrombin	Num. Vars/Feats	139,531	8709	32	5 / 3
	AUC	0.925	0.919	0.926	0.939

calculated using the number of variables selected, v, and the degree of the kernel, d. FSMB provides simpler models in all but one case.

Lung Cancer Data Set.

The lung cancer data set is used to classify gene expression samples between squamous and adenocarcinoma types of cancer. Previous results on these data sets using RFE and HITON have been reported (Aliferis *et al.*, 2003a) (this data set was also used in the analysis of Chapter II.4.4). The data were split following the same cross validation design exactly as in *ibid* in order to estimate the performance of the model and optimize SVM parameters from the sets $d = \{1, 2, 3, 4\}$ and $C = 10^i, i = \{-8, \ldots, 3\}$. For FSMB, the SVM model with all variables and trained with the optimal parameters is used by the FSMB algorithm to heuristically identify the top 1000 features. The top features were passed to HITON in order to identify the $MB_\Phi(T)$.

Table III-11: Number of Features with Variable Selection Methods. The number of features in final models is calculated from the number of variables selected by the methods (None, RFE, and HITON) and degree of the SVM model. The number of features for FSMB is directly determined by what FSMB returns.

Data Set	Degree of Kernel Metric	Variable Selection Method None	RFE	HITON	FSMB
Lung Cancer	d = 3	3.33×10^{11}	1540	969	2
Splice Site	d = 6	5.99×10^{12}	5.99×10^{12}	906192	21
Infant Mortality	d = 2	3828	21	36	37
OHSUMED	d = 2	103313125	6441	630	43
Gisette	d = 2	12507501	196251	25878	38
Sylva	d = 2	23653	406	1326	41
Hiva	d = 2	1309771	1378	45	15
Thrombin	d = 2	9.73×10^{9}	37936405	561	3

The results of the different methods on the Lung Cancer data set are summarized in Table III-10. For this data set, the best classification performance is when all variables are included in the model. However, there is only a slight reduction in performance when models of much smaller subsets of variables are selected. The number of variables selected by RFE is 19 (0.15% of the total number of variables) and by HITON is 16 (0.13% of the total number). FSMB returns 2 features $\{X_{2515}X_{3157}X_{12097},\ X_{205}X_{2515}X_{12097}\}$ defined over 4 variables (0.03% of the total number). On this data set, FSMB performed well with fewer variables selected. The discriminative performance of FSMB can be visualized by plotting the data over these two features (Figure III-9).

Splice Site Data Set.

The second data set is for the classification task identifying splice sites from DNA sequences (Degroeve *et al.*, 2002; Saeys *et al.*, 2003). This data set was also used in the analysis of Chapter II.4.4 where more details on the data can be found. Model parameters were selected via 10-fold cross validation from the sets degree = $\{1, 2, 3, 6, 9\}$ and $c = \{0.001, 0.05, 0.1\}$ (choice of the parameter options was influenced by previously published results). The best model parameters were selected via cross-validation by maximizing AUC and found to be degree 6 kernel with $c = 0.05$. With the parameters of the model selected, a final SVM model was created on the training data set to examine the top features and weights for each problem. In FSMB, the top 1000 features were passed

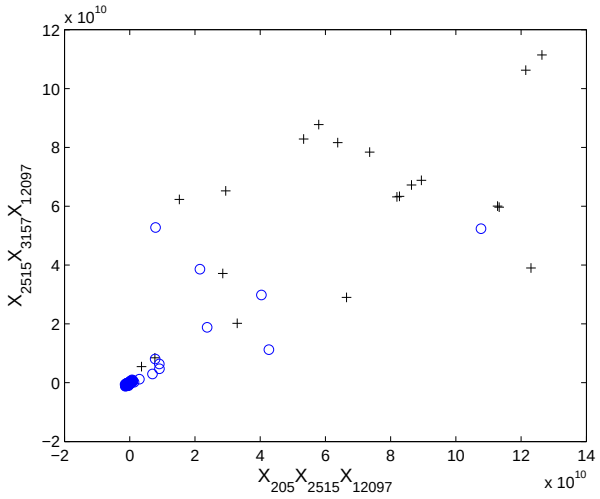


Figure III-9: Lung Cancer Data on FSMB Features. The lung cancer data is plotted on the two FSMB features. The "o" points are negative class examples and "+" are positive class examples.

to HITON to identify the $MB_{\Phi}(T)$. The classification performance was calculated for an SVM model using all variables, the variables selected by RFE, HITON, and FSMB with results shown in Table III-10.

The best classification performance resulted when using all variables. However, FSMB had only a slight performance reduction while reducing the number of variables to only 2.5% of the total. RFE did not reduce the number of variables on this problem and HITON selected 26 variables but had a lower classification performance. FSMB does well on this data set and looking at the domain this is not unexpected. The top features identified often involve pairs, triplets, and quartets of variables involving groupings of the upstream T's. These results are consistent with biologist beliefs in recognizing splice sites (see Chapter II.4.4 for additional discussion). FSMB is able to identify the features (variable combinations) rather than each variable individually.

Infant Mortality Data Set.

The results of the different variable selection methods on the Infant Mortality data set are presented in Table III-10. The best classification performance is achieved via the model that uses HITON's variables. This outperforms the model that uses all variables, FSMB, and RFE. FSMB also outperforms RFE and the model using all variables. The reduction in the number of variables

is not as great for this data set (since the original data set has only 86 variables). However, the number of variables returned is less than 20% for all methods.

OHSUMED Data Set.

For the OSUMED results (Table III-10), the best classification performance is achieved via the model that uses all 14,000+ variables. For the models using only a subset of the variables (selected by either RFE, HITON, or FSMB), the classification performance decreases. In terms of the number of variables found by the variables selection methods, RFE retains the most with 112 variables (0.78% of the total number of variables). HITON returns 34 variables (0.24% of the total), while FSMB returns 43 features defined over 40 variables (0.28% of the total number of variables).

On this data sets, none of the variable selection methods are effective compared to the performance with the full data set. One possible explanation of this results it the sparseness of the data. Recall, this data comes from the text categorization domain where the documents are represented using a bag of words approach. There might exist words or phrase (corresponding to features in the FSMB approach) that are indicative to a specific target class. However, if the words only appear in a very limited number of documents then the whether using a statistical test or SVM model there will be little chance of this variables (feature being selected).

Gisette Data Set.

The Gisette data set results all illustrate good classification performance by all models. The model built with all 5000 original variables has a classification performance of 0.997. HITON illustrates the same performance with only 226 variables. RFE has 625 variables yield a classification performance of 0.998. FSMB returns only 48 variables yet still achieves a classification performance of 0.994.

Sylva Data Set.

The results on the Sylva data set of the three main variable methods are presented in Table III-10 for comparison. Models build with all variables and the three variable subsets all have high classification performance. In terms of the number of variables selected, RFE and FSMB had the

greatest reduction in number of variables selected returning 12.5% and 13.4% of the total number of variables.

Hiva Data Set.

The Hiva data set comes from the drug discovery domain specifically whether a drug compound is active against AIDS HIV infection. The variables represent properties of the molecule inferred from its structure. On the Hiva data set, the top classification performance came from the model involving all variables (AUC = 0.717), with the model built with the variables of FSMB exhibiting the next best performance (AUC = 0.702). This model was built using only 0.87% of the total number of variables. The models build using the variables selected with RFE or HITON had lower classification performance. On this data set, FSMB doing so much better than HITON suggest that there might be epistatic multivariate relationships in this domain. For instance there are 9 variables in the features FSMB returns that do not have large enough association with the target to be considered by the HITON method. Those variables solely make up 8 of the 15 features returned by FSMB. Looking at the feature data the variables with no univariate association now have detectable association between the target and the features.

Thrombin Data Set.

The general results for the variable selection method on the Thrombin data set are also shown in Table III-10. The classification performance was highest on the variables subset of FSMB (although the performance for the other methods is not much less). The number of variables returned by RFE is more than 8,000 (5.73% of the total number of variables). The number of variables returned by HITON is 32 (0.023% of the total). FSMB returns 3 features $\{X_{16598}X_{17177}, X_{6524}X_{16896}, X_{16888}\}$ defined over 5 variables (0.0036% of the total). The discriminative performance of FSMB can be visualized by plotting the data over the two top features (Figure III-10).

III.7.3 Other Variable Selection Methods Results

The data sets of Infant Mortality, OHSUMED, Gisette, Sylva, Hiva, and Thrombin were all in the analysis of Aliferis *et al.* (2009a) and Aliferis *et al.* (2009b). This evaluation followed the same

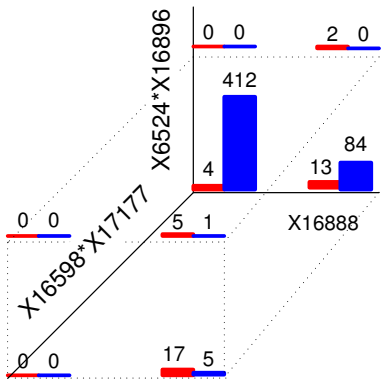


Figure III-10: Thrombin Data Split by FSMB Features. The Thrombin data distribution is shown on the two FSMB features.

experimental design so as to include further comparison with additional variable selection methods. A selection of 26 variable selection instances (a single method may have been run with several different parameters) are reported here (the original evaluation had additional parameter combinations). The performance of each method (including FSMB) was reported for the number of variables selected, the percentage of variables selected and the classification performance in Appendices B.IV, B.V, and B.VI.

In general, FSMB follows the trend of other causal discovery approaches in it parsimonious selection of variables. Furthermore, it is able to achieve this reduction in variables while often not sacrificing the classification performance when compared to the full model or other variable selection methods.

III.8 Conclusions

The Feature Space Markov Blanket algorithm for variable selection was presented. FSMB marries two different approaches to the variable selection problem, the kernel-based and the Markov Blanket-based strategies, combining their strengths. Similar to the latter approach, FSMB identifies the Markov Blanket as the smallest, most-predictive variable set. However, the search for the Markov Blanket is performed in the feature space, where multivariate associations may become pairwise associations and thus detectable. This overcomes a significant limitation of all present MB-based

algorithms that depend on the existence of pairwise association of the MB-variables and the target. The feature space is implicitly constructed by a Support Vector Machine. A heuristic sub-algorithm explicitly constructs only the interesting portion of the feature space, those features with a high absolute weight and possibly, a high association with the target. This work demonstrates the advantages of FSMB over two standard algorithms HITON and RFE, and shows promising results on both simulated data and on a large, real data sets.

The results of this research have several limitations. First, the choice of parameters for the FSMB method is not addressed (for which the degree of the kernel may greatly affect the performance): either the kernel parameters are given (for the simulated data sets) or selected via a cross validation design (for the real data sets). Also, the number of features K to pass to HITON is also given as a small default value (e.g., 50, 100, or 1000); whether these are reasonable values needs further investigation. Additionally, the heuristic method offers a simple, efficient method to identify the top weights, but other search methods could be employed with tradeoffs in efficiency and/or quality of the results. Finally, the FSMB relies on the assumption that the features with the largest magnitude weights are most important (relevant) for the classification task; whether this assumption holds and why it may fail will be examined. These theoretical properties of FSMB should examined and the heuristic for identifying the top-weighted features improved in future work.

Additional future work in this area should be completed to first formally define the class of distributions where the FSMB algorithm is expected to exceed other methods. Once this completed, then a examination of the prevalence of such distributions should be explored (Dash, 2005). Also, the performance of the method should be more fully tested for various aspects of the underlying problem, e.g., training set sample size, number of relevant variables, number of irrelevant variables, number of relevant features, redundant features. This last concept connects to the idea of there may be many equivalent sets of variables that make-up the Markov Boundary (Statnikov, 2009). Will this method identify all possible members of the Markov Boundary or what possible subsets can be found? Questions such as these should receive additional attention in the future work.

Finally, this work has solely used SVM models for binary classification. In the future, we would

like to explore extending the theory and methods to SVM models for regression or multi-categorical SVMs. We believe such extensions are possible; the methods rely on the properties of the polynomial kernel to construct and consider features. This kernel can be used in the alternative SVM formulations. The promising results of this paper support many future theoretical and empirical investigations into this new method and its properties.

CHAPTER IV

LEARNING BAYESIAN NETWORK REGIONS

The extremely large data sets emerging from a multitude of domains have exceeded the limits of traditional Bayesian network learning algorithms. Often however, an analyst may be only interested in the Bayesian network structure (region) around a target variable of interest. One approach to learning the region (called the global method throughout the Chapter) first induces the full network and then prunes or extracts the region of interest. A second approach directly learns just the desired region without making inductive inferences for unrelated parts of the network. A global approach has the advantage that it may use information induced from remote parts of the network to better learn the region of interest. At the same time however, erroneous statistical inferences may also propagate and affect the induction of the region. In this Chapter, one of the best algorithms for learning Bayesian networks (MMHC) is extended to locally learn a region. The resulting local method is compared to the global MMHC in an empirical evaluation. As expected, the local method takes only a fraction of the time to learn the region compared to MMHC. Interestingly, the empirical results also show that the local technique learns a region with equal or better quality compared to the global one. In other words, propagation of errors from remote parts of the network often outweigh the benefits from propagation of useful information. Thus, within the scope of the evaluation and current methods for learning Bayesian networks, it is possible to learn a local structure of interest in reasonable time and without sacrificing the quality of learning.

IV.1 Introduction

In recent years there has been increasing interest in the use of Bayesian networks (BNs) for causal and predictive modeling in several domains including learning genetic regulatory pathways (Friedman *et al.*, 2000; Hartemink *et al.*, 2002; Friedman, 2004), discovering protein signaling networks (Sachs *et al.*, 2005; Woolf *et al.*, 2005), and modeling and generating hypotheses for biomedical

researchers. These domains require techniques that scale to handle the large data sets consisting of thousands to hundreds-of-thousands of variables and samples. Unfortunately, learning the most probable a-posteriori Bayesian network is an NP-Hard problem. Until recently, learning the complete Bayesian network in reasonable time with hundreds or thousands of variables was beyond the reach of any algorithm. Thus, BN-based data analysis had to be restricted to domains of relatively small dimensionality.

However, researchers may be interested in the local area of the BN around a target variable of interest. In this Chapter, we focus on methods for learning a BN region defined as the subgraph within the radius depth of d edges about a target node. The most simplistic approach, referred to as the *global approach*, for learning such a BN region is to first learn the entire network and then prune the edges to the desired region depth. This global approach lacks efficiency by learning parts of the network that may be greatly removed from the target region (especially for problems with a large number of variables and small depth). However, the quality of the network learned may be aided by information propagating from one part of the network to a distant area. Alternatively, errors may also be propagated reducing the network quality. For example, if the learner identifies a collider, i.e., the substructure $A \rightarrow C \leftarrow B$, the orientation of the edges may propagate and help orient the other edges in the region. Conversely, if C is not truly a collider, statistical fluctuations result in a false inference. Then the constraints and information propagated by this induction may negatively affect the quality of the network structure.

An alternative intuitive approach, referred to as the *local approach*, is to first identify members of the region out to the specified depth then learn the network structure for this subset of variables. Variables belonging to the region about a target could be identified by recursively applying local learning techniques, e.g., methods for finding the parents and children or the Markov blanket of a node. Any number of traditional BN learning techniques can then be applied to reconstruct the network region structure using only the regional subset of variables. As the local approach is restricted to using only closely related variables, the propagation of information, whether correct

or erroneous, is limited in its scope. Whether and how this propagation of information affects the quality of a learned structure is unclear.

This research compares the global and local approaches in terms of time-efficiency and structural quality. We expect the comparison will lend insight into whether the global approach benefits from or is hindered by the propagation of information. For the study, the global BN learning algorithm used is Max-Min Hill-Climbing, *MMHC* (Tsamardinos *et al.*, 2006b). *MMHC* is one of the most competitive BN learning algorithms today and can be naturally modified to take a local approach. This new algorithm, *RegionMMHC* presented in Section IV.3.3, works by first identifying the set of parents and children of the target variable in the data-generating BN, by using constraint-based techniques. This process is then repeated recursively to the set of parents and children, identifying the nodes and edges, without their orientation, within the targeted region. A search-and-score procedure is subsequently applied to orient this BN region skeleton and the final graph of the region is returned.

By using *MMHC* and *RegionMMHC* as the global and local approaches to learning BN regions respectively, the comparisons between the two approaches can be made with algorithms that use the same assumptions, have similar learning mechanisms, and consist of the same functional basis. The results show that in general, (1) *RegionMMHC* takes a fraction of the time to learn a specified region and (2) the learned structure of *RegionMMHC* is of the same quality as the structure learned by the global approach. In other words, the propagation of errors from remote parts of the network offsets the benefits from propagation of useful information. Thus, by using the local method one can scale learning to domains with unprecedented sizes, hundreds of thousands of variables, while retaining the quality of learning. Additionally, the local approach is compared to another method that uses the concept of learning a local region from a target node, *AlgorithmGPC* (Peña *et al.*, 2005). When the *AlgorithmGPC* learns the network (for larger networks and sample sizes *AlgorithmGPC* does not complete), it is in general faster than *RegionMMHC* however the regions learned are of lower quality.

IV.2 Background

First a few notational definitions, a variable is denoted with an upper-case letter (e.g., A, V_i) and a state or value of that variable by the same lower-case letter (e.g., a, v_i). A set of variables is denoted by upper-case bold-face (e.g., $\mathbf{Z}$, $\mathbf{Pa_i}$) and the corresponding lower-case bold-face symbol is an assignment of state or value to each variable in the given set (e.g., $\mathbf{z}$, $\mathbf{pa_i}$). Calligraphic fonts are used for special sets of variables such as the set of all variables considered $\mathcal{V}$. In this Chapter, only discrete probability distributions and complete data sets are considered (i.e., all modeled variables in all training instances obtain an observed known value). We denote conditional independence of X and Y given $\mathbf{Z}$ according to distribution P as $Ind_P(X;Y|\mathbf{Z})$ and dependence as

$$Dep_P(X;Y|\mathbf{Z}) \equiv \neg Ind_P(X;Y|\mathbf{Z})$$

Definition IV.1. *Let P be a discrete joint probability distribution of the random variables*[1] *in some set $\mathcal{V}$ and $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a Directed Acyclic Graph (DAG). We call $\langle \mathcal{G}, P \rangle$ a (discrete)* Bayesian network *if $\langle \mathcal{G}, P \rangle$ satisfies the Markov Condition: every variable is independent of any subset of its non-descendant variables conditioned on its parents (Pearl, 1988; Spirtes* et al.*, 1993; Glymour & Cooper, 1999; Pearl, 2000; Spirtes* et al.*, 2000; Neapolitan, 2003).*

We denote the set of the parents of variable V_i in the graph $\mathcal{G}$ as $\mathbf{Pa}_i^{\mathcal{G}}$. By utilizing the Markov Condition, it is easy to prove that for a Bayesian network $\langle \mathcal{G}, P \rangle$ the distribution P of the variables $\mathcal{V}$ can be factored as follows:

$$P(\mathcal{V}) = P(V_1, \ldots, V_n) = \prod_{V_i \in \mathcal{V}} P(V_i | \mathbf{Pa}_i^{\mathcal{G}})$$

The graph of a network in conjunction with the Markov Condition directly encode some of the independencies of the probability distribution and entail others (see Neapolitan 2003, pp. 70 for a definition of entailment). A graphical criterion for entailment is that of d-separation (Pearl, 1988, 2000). It is defined on the basis of blocked paths:

[1] Variables are also interchangeably called nodes or vertices in the context of a Bayesian network.

Definition IV.2. *A node W of a path p is a* collider *if p contains two incoming edges into W.*

Definition IV.3. *A path p from node X to node Y is* blocked *by a set of nodes $\mathbf{Z}$, if there is a node W on p for which one of the following two conditions hold:*

1. *W is not a collider and $W \in \mathbf{Z}$, or*
2. *W is a collider and neither W or its descendants are in $\mathbf{Z}$ (Pearl, 1988).*

Definition IV.4. *Two nodes X and Y are* d-separated *by $\mathbf{Z}$ in graph $\mathcal{G}$ (denoted as $Dsep_{\mathcal{G}}(X;Y|\mathbf{Z})$) if and only if every path from X to Y is blocked by $\mathbf{Z}$. Two nodes are* d-connected *if they are not d-separated.*

A pair of nodes d-separated by a variable set in network $\langle \mathcal{G}, P \rangle$ is also conditionally independent in P given the set (Verma & Pearl, 1988). The faithfulness condition below, asserts that the conditional independencies observed in the distribution of a network are not accidental properties of the distribution, but instead due to the structure of the network.

Definition IV.5. *If all and only the conditional independencies true in the distribution P are entailed by the Markov condition applied to $\mathcal{G}$, we will say that P and $\mathcal{G}$ are* faithful to each other *(Spirtes* et al.*, 1993, 2000; Neapolitan, 2003). Furthermore, a distribution P is* faithful *if there exists a graph, $\mathcal{G}$, to which it is faithful.*

Definition IV.6. *A Bayesian network $\langle \mathcal{G}, P \rangle$ satisfies the* faithfulness condition *if P embodies only independencies that can be represented in the DAG $\mathcal{G}$ (Spirtes* et al.*, 1993). We will call such a Bayesian network a* faithful network*.*

The following theorem is utilized in most constraint-based algorithms such as the ones presented here:

Theorem IV.1. *In a faithful BN $\langle \mathcal{G}, P \rangle$ on variables $\mathcal{V}$ there is an edge between the pair of nodes X and Y in $\mathcal{V}$ iff $Dep_P(X;Y|\mathbf{Z})$, for all $\mathbf{Z} \subseteq \mathcal{V}$ (Spirtes* et al.*, 1993).*

We define the distance between a node X and T, $\delta(X,T)$ as the length of the shortest undirected path between the two nodes. We denote an edge from node X to node Y by $X \rightarrow Y$.

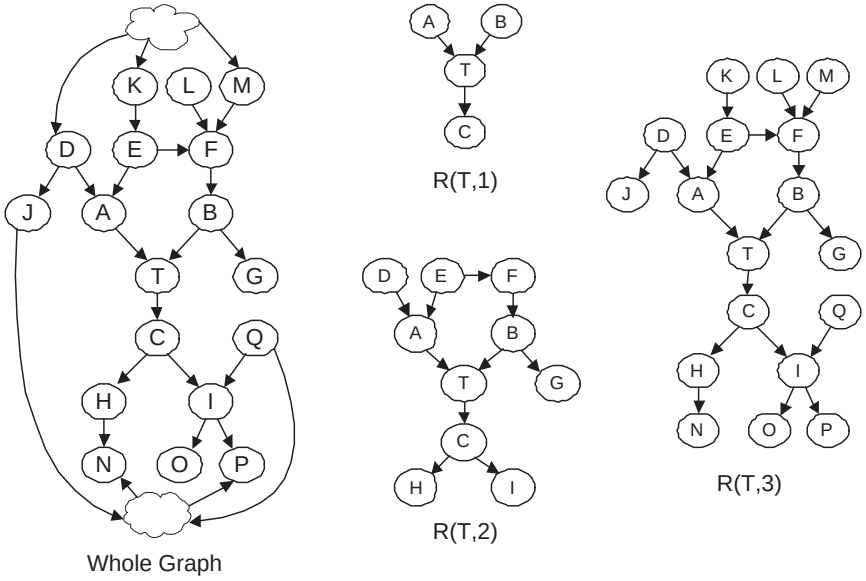


Figure IV-1: The regions, $R(T,1), R(T,2)$ and $R(T,3)$ extracted from the whole graph on the left of the figure.

Definition IV.7. *A region of depth d around node T of the BN with DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$is denoted as $R_{\mathcal{G}}(T,d)$. The region $R_{\mathcal{G}}(T,d)$ is a subgraph of $\mathcal{G}$ consisting of the nodes with distance at most d from T and all edges in $\mathcal{E}$ between said nodes. $R_{\mathcal{G}}(T,d) = \langle \mathcal{V}_{\mathcal{R}}, \mathcal{E}_{\mathcal{R}} \rangle$ where,*

1. $\mathcal{V}_{\mathcal{R}} = \{V_i \in \mathcal{V} : \delta(V_i, T) \leq d\}$, *and,*
2. $\mathcal{E}_{\mathcal{R}} = \{V_i \rightarrow V_j : V_i, V_j \in \mathcal{V}_{\mathcal{R}}, V_i \rightarrow V_j \in \mathcal{E}\}$.

The subscript $\mathcal{G}$ is dropped when it can be inferred from the context. To illustrate this concept, Figure IV-1 shows three regions of increasing depth, $R(T,1)$, $R(T,2)$ and $R(T,3)$, extracted from the whole graph on the left side of the figure.

Several specialized regions have been studied extensively, namely algorithms for finding the parents and children of a node (region with depth, $d = 1$) and the Markov Blanket (the parents, children and spouses) of a target node (subset of a region with depth, $d = 2$) (Koller & Sahami, 1996; Margaritis & Thrun, 1999; Tsamardinos *et al.*, 2003a,c; Yaramakala & Margaritis, 2005). Other local learning approaches include the LCD algorithm (Cooper, 1997) and the CCC algorithm (Silverstein *et al.*, 2000); both of which can only identify local features of the underlying BN if they exhibit certain special structural properties.

In Tsamardinos *et al.* (2003c), the authors describes the first algorithm that could learn a local BN region of arbitrary size; their algorithm returns the BN skeleton in a radius d edges around a target or "seed" node. This method was shown to reconstruct skeletal regions from a network of 10,000 variables and was the first to scale to domains of such dimensionality. A similar general methodology has also been used to learn boolean networks (Hashimoto *et al.*, 2004). In Peña *et al.* (2005), a corrected version of the algorithm of Tsamardinos *et al.* (2003c) is used (the correction is also found in Tsamardinos *et al.* (2006b)) to recursively grow out the skeleton region a given depth then edge orientations are added to return an equivalence class. Also, Bai *et al.* (2008) learn and orient the Markov Blanket for a target of interest. The purpose of this research is not to find the "best" method for learning regions of a BN (if one could even proof such a method existed), but to compare the two approaches (local vs. global) on this problem. Consequently, the focus of the Chapter will be on the *MMHC* and *RegionMMHC* algorithms (although we include results with Pena's *AlgorithmGPC* as additional comparison).

IV.3 Learning Regions of Bayesian Network

The main problem examined in this research is that of learning a local region $R_{\mathcal{G}}(T, d)$ of a BN $\langle \mathcal{G}, P \rangle$, given T, d and statistical data $\mathcal{D}$ following the joint distribution P of the network (in other words the data-generating graph $\mathcal{G}$ is unknown or the problem would be trivial). Since there may be numerous graphs $\mathcal{G}$ such that $\langle \mathcal{G}, P \rangle$ is a Bayesian network the above problem is not well-defined. Several definitions are possible for the problem of learning $\mathcal{G}$ or $R_{\mathcal{G}}(T, d)$ from the data, giving preferences to inducing different structures. Here, we adapt Neapolitan (2003), pp. 533 to define the problem as follows:

Definition IV.8. *Let P be a faithful distribution and $\mathcal{D}$ a statistical sample following P. The problem of learning the structure of a region with nodes of at most distance d from T given $\mathcal{D}$ is to induce $R_{\mathcal{G}}(T, d)$ where $\langle \mathcal{G}, P \rangle$ is a faithful Bayesian network.*

We now present the global and the local approach for solving the above problem. In the global approach a BN learning algorithm reconstructs from the data $\mathcal{D}$ a complete BN $\langle \mathcal{G}, P \rangle$. Subsequently,

with the graph $\mathcal{G}$ given, the region $R_{\mathcal{G}}(T, d)$ is extracted using simple graph operations. In the local approach, the region is found considering only the subset of variables identified to belonging to the region. In this Chapter, the two approaches use the same base algorithmic foundation, which aids in their comparison. This base algorithm, Max-Min Parents and Children, is discussed followed by the global and local approaches built upon this foundation.

IV.3.1 The Max-Min Parents and Children Algorithm

Max-Min Parents and Children (*MMPC*, first published in Tsamardinos *et al.* (2003c), with a correction described in Tsamardinos *et al.* (2006b), a short description is presented here for self-containment) is a local discovery algorithm. From the algorithm's name, "Max-Min" refers to the heuristic used in the algorithm while "Parents and Children" refers to the algorithm's output. A few notational conventions are as follows: the set of parents and children of a node T in a graph $\mathcal{G}$ is denoted as $\mathbf{PC}_T^{\mathcal{G}}$, the parents of T in $\mathcal{G}$ is denoted as $\mathbf{Pa}_T^{\mathcal{G}}$. In two faithful Bayesian networks (to the same distribution), $\langle \mathcal{G}, P \rangle$ and $\langle \mathcal{G}', P \rangle$, then any variable T has the property $\mathbf{PC}_T^{\mathcal{G}} = \mathbf{PC}_T^{\mathcal{G}'}$ (Verma & Pearl, 1990, 1991; Tsamardinos *et al.*, 2003d). In other words, the set of parents and children of T is unique among the Bayesian networks faithful to the same distribution. Consequently, the superscript is dropped and the parents and children set is denoted as $\mathbf{PC}_T$. For example, $\mathbf{PC}_T = \{A, B, C\}$ in the network in Figure IV-1.

MMPC is shown given a target variable, T, and observational data, $\mathcal{D}$, to return $\mathbf{PC}_T$ if there is a graph faithful to the data distribution and all statistical tests are reliable. Note, a variable may be a child of T in one network and a parent in another network both of which are faithful to the same distribution, e.g., $X \rightarrow T$ and $X \leftarrow T$. The set of parents and children remain the same, e.g., $\{X\}$ is the $\mathbf{PC}_T$ in both networks.

Running *MMPC* for a target node T is a method of identifying the edges into and out of T, without knowing the orientation of the edges. Assessing the results of invoking *MMPC* with all the variables as targets, all edges in the network can be identified (this un-oriented network is referred to as the skeleton of the network). Full reconstruction require the orientation of the edges; an

Algorithm 5 $\overline{MMPC}$ Algorithm

```
1: procedure MMPC (T,D)
   Input: target variable T; data D
   Output: the parents and children of T in any Bayesian
   network faithfully representing the data distribution
   %Phase I: Forward
2:     CPC = ∅
3:     repeat
4:         ⟨F, assocF⟩ = MaxMinHeuristic(T; CPC)
5:         if assocF ≠ 0 then
6:             CPC = CPC ∪ F
7:         end if
8:     until CPC has not changed

   %Phase II: Backward
9:     for all X ∈ CPC do
10:        if ∃S ⊆ CPC, s.t. Ind(X; T|S) then
11:            CPC = CPC \ {X}
12:        end if
13:    end for

14:    return CPC
15: end procedure

16: procedure MaxMinHeuristic(T,CPC)
   Input: target variable T; subset of variables CPC
   Output: the maximum over all variables of the minimum association with T relative to CPC,
   and the variable that achieves the maximum
17:    assocF = max_{X∈V} MinAssoc(X; T|CPC)
18:    F = arg max_{X∈V} MinAssoc(X; T|CPC)
19:    return ⟨F, assocF⟩
20: end procedure
```

algorithm that completes the full reconstruction is the Max-Min Hill-Climbing (*MMHC*) algorithm and is discussed as part of the global approach for learning a region.

A simplified version of the algorithm we call $\overline{MMPC}$ is presented first for clarity. $\overline{MMPC}$ may return false positives depending on the structure, i.e., it may return a superset of $\mathbf{PC}_T$. The complete and sound *MMPC* is a simple extension of this base algorithm.

The simplified algorithm is shown in Algorithm 5. $\overline{MMPC}$ makes use of functions for independence tests, $Ind_P(X;T|\mathbf{Z})$, and measures of association, $Assoc(X;T|\mathbf{Z})$. The function $Ind_P(X;T|\mathbf{Z})$ uses a statistical test on the training data, $\mathcal{D}$[2], to estimate and return the values *true* if X and T are conditionally independent given $\mathbf{Z}$. The function $Assoc(X;T|\mathbf{Z})$ estimates the strength of the

[2]For simplicity of notation, $\mathcal{D}$ is omitted from the parameter list of all functions that use the data.

association between X and T given $\mathbf{Z}$. The following assumption is made relating the two functions, $Ind_P(X;T|\mathbf{Z}) \Leftrightarrow (Assoc(X;T|\mathbf{Z}) = 0)$. Finally, the function MinAssoc$(X;T|\mathbf{Z})$ is defined as the minimum association between X and T considering all subsets of $\mathbf{Z}$.

From Theorem IV.1, the identification of a subset $\mathbf{Z}$ such that $Ind_P(X;T|\mathbf{Z})$ results in the knowledge of no edge existing between X and T in the learned graph. $\overline{MMPC}$ makes use of this property to try to quickly identify the conditioning set $\mathbf{Z}$ that results in the independence relationship between variables X and T, proving $X \notin \mathbf{PC}_T$.

The algorithm $\overline{MMPC}$ discovers $\mathbf{PC}_T$ with a two-phase design. In the first (forward) phase, a candidate $\mathbf{PC}_T$, $\mathbf{CPC}$, is created with variables entering sequentially via a heuristic function. The Max-Min heuristics function *selects the variables that maximizes the minimum association with T relative to **CPC**.* The intuition behind the heuristic is the variable that despite all efforts, conditioning upon all subsets of $\mathbf{CPC}$, continues to be strongly associated with the target T should be selected. The Max-Min heuristic is admissible; all variables adjacent to T, i.e., with an edge to of from T, and possibly additional variables will enter $\mathbf{CPC}$. The first phase is completed once the remaining variables are found independent of T by a subset of $\mathbf{CPC}$.

The second (backward) phase aims to remove any false positives that may have entered the $\mathbf{CPC}$. Each variable, X, in $\mathbf{CPC}$ is tested for a subset $\mathbf{S} \subseteq \mathbf{CPC}$ such that $Ind_P(X;T|\mathbf{S})$ holds, therefore X can be removed from $\mathbf{CPC}$. The search over all subsets at lines 10, 17, and 18 on the algorithm is in practice bounded by the available sample.

The output of $\overline{MMPC}$ will include all members of $\mathbf{PC}_T$ in its output assuming faithfulness. However, $\overline{MMPC}$ may return false positives in certain cases. Since the relation $\mathbf{PC}$ should be symmetric, a break of symmetry in the output of the algorithm is an indication of a false positive member. *MMPC* in Algorithm 6 checks whether $T \in \overline{MMPC}(X, \mathcal{D})$ for all $X \in \overline{MMPC}(T, \mathcal{D})$; if this is not the case it removes X from its output. *MMPC* is theoretically sound and will return $\mathbf{PC}_T$ when the sample is adequate for no errors to occur in the tests of independence and the network is faithful (Tsamardinos *et al.*, 2006b).

Algorithm 6 *MMPC* Algorithm

procedure MMPC(T,$\mathcal{D}$)
  CPC = $\overline{MMPC}(T, \mathcal{D})$
  for every variable $X \in$ **CPC do**
    if $T \notin \overline{MMPC}(X, \mathcal{D})$ **then**
      CPC = **CPC** $\setminus X$
    end if
  end for

  return CPC
end procedure

Algorithm 7 *MMHC* Algorithm

1: **procedure** MMHC($\mathcal{D}$, T, d)
  Input: data $\mathcal{D}$, target node T, distance d
  Output: $R_{\mathcal{G}}(T, d)$
  % Restrict
2: **for** every variable $X \in \mathcal{V}$ **do**
3: **PC**$_X$ = MMPC($X, \mathcal{D}$)
4: **end for**
  % Search
5: Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge* $Y \rightarrow X$ if $Y \in$ **PC**$_X$.
6: Let $\mathcal{G}$ be the highest scoring DAG found
7: Extract and return region $R_{\mathcal{G}}(T, d)$
8: **end procedure**

IV.3.2 Global Structure Learning

Any algorithm for learning BNs, such as PC (Spirtes *et al.*, 1990), TPDA (Cheng *et al.*, 2002), OR (Moore & Wong, 2003), GES (Chickering, 2002), among others, can be used in conjunction with pruning to address the learning regions problem. However, most of the general BN learning algorithms cannot be easily adapted to learn the region, without first inducing the complete graph. A recent BN learning algorithm called the Max-Min Hill-Climbing (*MMHC*) algorithm (Tsamardinos *et al.* 2006b, Algorithm 7) is shown to be (a) one of the most competitive algorithms for learning BNs in extensive studies against a plethora of other state-of-the-art algorithms and (b) easily and naturally adapted for local learning of BN regions. By use of the global and local versions of the algorithm we can compare the two approaches and identify their respective strengths, since the underlying learning mechanisms are similar.

Algorithm 7 presents the *MMHC* algorithm (adapted to return a region). It takes as input a data set, $\mathcal{D}$, a target T and a distance d, learns a network $\mathcal{G}$ from the data $\mathcal{D}$, and returns the

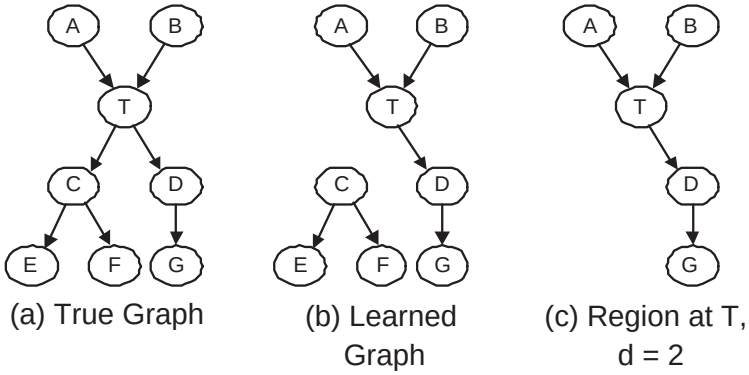


Figure IV-2: Finding a region using the global approach and *MMHC*: (a) the original graph to be learned, (b) the graph actually learned by *MMHC* (with finite sample), (c) the region extracted from the global graph, about node T up to distance 2.

region $R_{\mathcal{G}}(T, d)$. *MMHC* uses the parents and children sets returned for each variable to restrict the search space of a search-and-score procedure. Specifically, a greedy hill-climbing search extended with a TABU list starting from the empty network is run. The search using operators of adding, deleting, and reversing an edge is restricted to only considering adding an edge $Y \rightarrow X$ if $Y \in \mathbf{PC}_X$. Once the complete network is induced from the data, the region $R_{\mathcal{G}}(T, d)$ is extracted using graph operations only and returned. An example of using *MMHC* is in Figure IV-2.

The data-generating graph structure (labeled true graph) is shown in Figure IV-2(a). The highest scoring DAG (from line 6 of the *MMHC* Algorithm 7) is shown in (b). Finally, the extracted region about T with depth 2 is illustrated in (c). This example displays a potential negative consequence to the global approach for learning a region, with errors propagating and compounding in regions of increasing depths (please note, local approaches have no guarantees on also not committing the same errors). An error in creating the learned graph (i.e, missing the edge $T \rightarrow C$) can propagate greater depths when extracting a region. The missing edge is at depth 1, but when the region to depth 2 is extracted three edges are now missing ($T \rightarrow C$, $C \rightarrow E$, and $C \rightarrow F$).

IV.3.3 Local Structure Learning

The new algorithm, *RegionMMHC*, is an extension of *MMHC* to learn the region of a BN around a target node T. *RegionMMHC* takes as its inputs a data set $\mathcal{D}$, a target node T, and a depth d.

Algorithm 8 *RegionMMHC* Algorithm

procedure REGIONMMHC($\mathcal{D}$, T, d)
Input: data $\mathcal{D}$, target node T, distance d
Output: $R_{\mathcal{G}}(T, d)$
% Restrict
    $nodes_0 = T$, $i = 1$
    while $i \leq d + 1$ **do**
        for every variable $X \in nodes_{i-1}$ **do**
            $PC_X = \text{MMPC}(X, \mathcal{D})$
            $nodes_i = nodes_i \cup PC_X$
        end for
        $i = i + 1$
    end while
% Search
    Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge* $Y \rightarrow X$ if $Y \in PC_X$.
    DAG = the highest scoring DAG found, pruned around T to depth of d.
    Return DAG
end procedure

Similarly to *MMHC*, *RegionMMHC* also uses *MMPC* to create candidate parents sets to restrict the search-and-score procedure.

In detail, the algorithm begins with discovering the parents and children of the target node, $\mathbf{PC}_T$. The members of $\mathbf{PC}_T$ are the variables at depth 1, $nodes_1$. For each member of the set, $X \in nodes_1$, the parents and children of the variable are found, $\mathbf{PC}_X$. These variables are added to the set $nodes_2$, the variables at depth 2. This continues recursively, for every variable $Y \in nodes_{i-1}$, the parents and children set of the variable are found $\mathbf{PC}_Y$ and the members are added to the set of nodes at depth i, $nodes_i = nodes_i \cup \mathbf{PC}_Y$. This process continues out to a depth of $d + 1$.

The greedy search procedure is started to orient and find the highest scoring network. The same search procedure as in *MMHC* is run except the variables are limited to $totalnodes = \bigcup_i nodes_i$. The highest scoring DAG is returned and the region can be found by pruning back to the given depth d.

In finding a region of depth d, note that the *RegionMMHC* algorithm runs the local learning and the search-and-score procedure to depth $d + 1$. The reason for this choice is the following: let X and Y be two top (minimal) nodes in the region with no edge to each other, i.e., two nodes with no parents within the region. If there is a common ancestor Z of both X and Y then the two variables are d-connected though a path $X \leftarrow \ldots \leftarrow Z \rightarrow \ldots \rightarrow Y$ and thus dependent. The only subsets

that would render X and Y conditionally independent include ancestors of X and Y that are not in the region. Thus, *MMPC* will fail to identify a d-separating subset and will assume there is an edge between X and Y. This would severely impair the accuracy of the reconstruction of the region by introducing several false positive edges: all pairs of nodes of the region that are connected through ancestors in the original graph would be connected by an edge.

IV.4 A Theoretical Comparison: Global vs Local

In the global approach, we first learn the skeleton of the graph, then orient *all* the edges[3] and finally, we extract the region of interest. In the local approach, we learn the skeleton of the region (to depth d+1) and orient the edges of the region only.

In terms of the efficiency of the two approaches, what are the expected differences between the two methods? First consider just the construction of the network (or region) skeleton, ignoring the search-and-score orientation portion of the algorithms. The performance of the two methods can be compared by the number of conditional independence tests and association calculations. A single operation of $\overline{MMPC}$ for a target will calculated the association of every variable with the target conditioned on all subsets of **CPC** (in the worst case) (Tsamardinos *et al.*, 2006b). The number of tests is bounded by $O(|\mathcal{V}| \cdot 2^{|\mathbf{CPC}|})$. In the global approach, identifying the network skeleton requires calling *MMPC* with all targets. By caching the calls to $\overline{MMPC}$ the overall cost is bounded by $O(|\mathcal{V}|^2 \cdot 2^{|\mathbf{CPC}|})$ in the worst case ($|\mathbf{CPC}|$ is the maximum found over all variables). The local approach is bounded by $O(|R| \cdot |\mathcal{V}| \cdot 2^{|\mathbf{CPC}|})$, where $|R|$ is the number of variables in the region (the term $|\mathbf{CPC}|$ is the maximum found within variables of the region). Note, in practice the size of the conditioning sets is limited by the available sample.

The relative difference between the two approaches can be more closely examined under the following two assumptions: first, the maximum size of the **CPC** used in both approaches is approximately the same and second, the upper bounds are tight. With those assumptions, the relative time efficiency (local / global) is expected to be $\approx \frac{|R|}{|\mathcal{V}|}$. That is to say the relative time of the local technique to the global technique is proportional to the size of the region discovered.

[3]Notice, that the hill-climbing search-and-score phase is allowed to remove some of the edges found.

In terms of quality of the regions structure produced by both approaches, the global approach has available to it *all* edges of the network to decide their orientation, while the local one has a myopic view of the network structure to use for orientation. The global approach can both be helped and hindered by information propagated to the region by inferences made at parts of the network at an arbitrarily distance from T. An example is shown in Figure IV-3. Suppose that the true network is the one shown in Figure IV-3(b) and contains a collider at distance d from the target that can be used to orient all edges in the network[4]. The local approach will not be able to make this inference, unless it reconstructs the complete network at distance $d+1$ from T. In particular, it will not be able to differentiate between the structures in Figure IV-3(a). Note that the distance d can be arbitrarily near or far from T.

Alternatively, if node C in the figure is mistaken for a collider by the global approach, this error will be propagated to the rest of the network. Such errors can occur both because of the greedy nature of the hill-climbing algorithm trapped in local minima and often, because of false positives and negatives in the independence and associations tests, or statistical fluctuations of the score used in the search-and-score greedy search due to finite available sample.

Thus, in the global approach, orientations of the edges in the local region are the result of a compound of inferences made not only locally, by propagated from all parts of the network. Since greedy search in *MMHC* is allowed to also remove edges, the reconstructed skeleton, and not just the orientations, of the region is affected by global inferences. It is currently unknown, whether in typical networks, distributions, and learning problems, global inferences are a positive, negative, or negligible influence on the average and so, whether locally restricting learning will impair, improve, or not affect (respectively) learning. The empirical investigation of this question is the subject of the next section.

[4]See the PC algorithm (Spirtes *et al.*, 1990) for a set of sound edge-orientation rules that will allow these orientations to be inferred given enough sample. The greedy search-and-score with Bayesian scoring will also provide the same orientations.

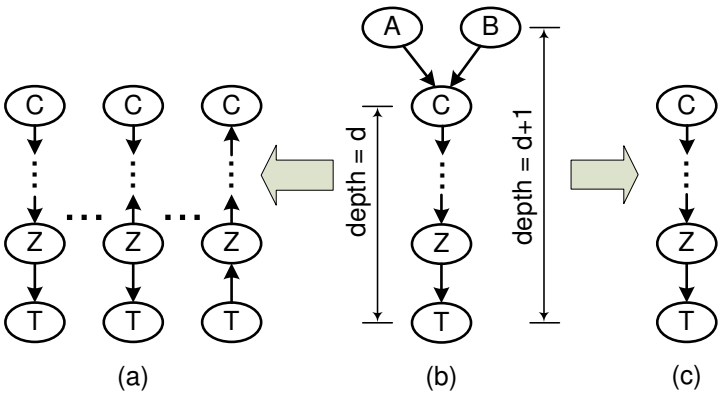


Figure IV-3: Intuition into learning out to depth $d+1$. (a) Algorithm runs only to depth d (b) Algorithm runs to depth $d+1$.

IV.5 Experimental Evaluation

The subsequent experiments were run with implementations of Max-Min Hill-Climbing (*MMHC*) and *RegionMMHC* in Matlab 6.5. Both algorithms used the standard 5% statistical threshold for tests of independence. The search-and-score procedure used BDeu scoring with the equivalent sample size of 10. The computations were run on Pentium Xeons, 2.4GHz with 2GB RAM running Linux.

The networks used in evaluating the algorithms consist of real or tiled networks and random BNs. The real BNs used are decision support BNs commonly cited in the literature: namely, ALARM (Beinlich *et al.*, 1989) and PIGS (Jensen, 1997). In addition tiled networks were created using a base of the ALARM, HAILFINDER (Abramson *et al.*, 1996), and INSURANCE (Binder *et al.*, 1997) networks. The tiled networks are created by tiling several copies of a smaller BN together where the tiling is performed in such a way that the structural and probabilistic properties of the original network are maintained (Tsamardinos *et al.*, 2006a). The tiled networks used in the evaluation ALARM10, HAIL5 and INS10 (where the number denotes the number of tiles). The network selection criteria was made to choose networks from a wide range of disciplines, with varying network properties, and consisting of several hundreds of variables (this problem size was selected in that it was not too small as to become a trivial problem, where exact solutions may be applied, but not too large a problem

Table IV-1: Characteristics of the Bayesian networks used in the evaluation: number of variables, number of edges, maximum and minimum domain values for the variables, and diameter of the graph (the greatest distance between any pair of nodes).

Network	Num. Vars	Num. Edges	Domain Range	Graph Diameter
ALARM	37	46	2 - 4	12
INS10	270	556	2 - 5	12
HAIL5	280	458	2 - 11	10
ALARM10	370	570	2 - 4	15
PIGS	441	592	3	19
RN50	50	138	2 - 3	5
RN100	100	284	2 - 3	6
RN500	500	1497	2 - 3	7

where the computational time becomes prohibitive in a thorough evaluation). Note, ALARM is an exception to the final criteria and was included as a baseline comparison.

Additionally, several random BNs were created with 50, 100, and 500 variables. The networks were created with a maximum fan-in of 5 (the exact number of parents for a given node is sampled from a discrete uniform distribution). The variables were randomly selected to have a domain of either 2 or 3. The random BNs were referenced by the number of variables, i.e., RN50 is the random network with 50 variables. The characteristic properties of these networks used in the evaluation are given in Table IV-1. Also, a larger random BNs was created with 100,000 variables in order to illustrate the scalability of new technique.

The training data sets were generated from the networks above with sample sizes (SS) of 500, 1000, and 5000. Five data sets were created for each sample size; the results then were averaged across the five samplings.

IV.5.1 Results of Evaluation

MMHC, referred to in the results tables as the global approach, was run for each sample size and network's five different data set samplings. *RegionMMHC*, referred to in the results tables as the

local approach, was run for each sample size and network on 10 randomly chosen target variables, each for depths of 1, 2, 3, 4, and 5 for the five different samplings.

Table IV-2: Execution Time Results for the local and global approaches. The global method's execution time is averaged over the 5 data samples and given for each network and sample size (the extraction of the region is dominated by learning the complete network therefore, the global time remains the same for each depth of region and is reported once). The local method execution time is averaged over the 5 data samples and the 10 random target nodes for each network and sample size. The relative execution time (local / global) is then reported for each network, sample size, and depth of region. The local method is statistically significantly faster than the global method (calculated using permutation testing) at a 0.05 level in all cases except for RN100 with 5000 samples at depths of 3, 4, and 5 indicated with †.

		Global	Relative Time (Local / Global)				
Data	SS	Time (sec)	d=1	d=2	d=3	d=4	d=5
Alarm	500	6.86	30.8%	40.4%	48.1%	59.0%	70.0%
Alarm	1000	8.90	28.8%	42.1%	53.5%	68.2%	82.7%
Alarm	5000	23.26	26.8%	42.0%	57.0%	73.6%	86.0%
Ins10	500	385.94	2.7%	4.5%	6.6%	9.2%	12.4%
Ins10	1000	610.62	2.8%	4.6%	6.3%	8.4%	10.7%
Ins10	5000	1529.38	4.0%	6.5%	10.0%	14.5%	20.5%
Alarm10	500	829.74	1.1%	2.0%	3.2%	4.5%	6.3%
Alarm10	1000	1026.06	1.4%	2.5%	3.9%	5.6%	7.8%
Alarm10	5000	1702.34	1.8%	3.5%	5.9%	9.2%	13.7%
Hail5	500	685.37	12.7%	31.0%	53.6%	69.4%	80.3%
Hail5	1000	1058.09	7.7%	15.6%	25.2%	37.3%	49.3%
Hail5	5000	4725.03	9.9%	14.9%	28.8%	49.7%	74.5%
Pigs	500	1710.44	4.6%	9.5%	19.9%	32.0%	47.0%
Pigs	1000	2029.42	7.1%	12.9%	26.8%	39.1%	54.3%
Pigs	5000	76753.42	28.5%	40.0%	76.6%	87.1%	97.6%
RN50	500	36.44	6.7%	13.5%	23.3%	35.2%	46.9%
RN50	1000	51.82	10.1%	22.9%	41.3%	59.9%	73.4%
RN50	5000	386.59	19.4%	48.6%	78.4%	90.6%	93.8%
RN100	500	13.77	17.0%	27.8%	40.9%	51.8%	58.7%
RN100	1000	14.78	24.2%	44.9%	66.3%	79.5%	86.9%
RN100	5000	75.64	46.1%	78.2%	95.2%†	99.8%†	99.9%†
RN500	500	1028.94	1.0%	1.8%	3.3%	5.6%	8.9%
RN500	1000	1072.09	1.4%	3.1%	6.8%	13.9%	26.5%
RN500	5000	3612.14	2.9%	11.6%	28.9%	56.1%	79.4%

Time Results

First, the two methods for discovering a region of a BN are compared in terms of execution time. The global results are calculated as the mean execution time for each network and sample size averaged over the five sampled data sets. Note, the global procedure uses the same time result

for each target node and depth of region because the time to extract the region from the global network is dominated by the time to learn the complete BN. The local results are calculated as the mean execution time for each network, sample size, and depth averaged over the 10 randomly chosen target variables and the 5 sampled data sets. The overall timing results for the evaluation networks are presented in Table IV-2. For each network and sample size the global execution time is given. In addition, the relative execution time (local / global) is presented for each depth, $d = 1$, 2, 3, 4, and 5.

The reduction in execution time ranges from *RegionMMHC* taking 1.0% of the time that *MMHC* does on RN500 with a sample size of 500 and depth of 1, to almost no reduction in time for RN100 with a sample size of 5000 and depth of 5. The ALARM10 network is shown to have the greatest time savings for all sample sizes. The local method is statistically significantly faster than the global method at a 0.05 level in all cases except for RN100 with 5000 samples at depths of 3, 4, and 5 (calculated using permutation testing) indicated with † in the Table IV-2. The experimental results confirm the natural intuition that as the depth of the region increases the time savings for using the local procedure versus the global approach is eroded, this is illustrated in Figure IV-4. Figure IV-4 plots the relative time averaged over the data set samplings, target nodes, and sample size (SS) for each of the networks.

The reduction of execution time varies widely between network and depth of the region. Much of the variation can be explained by the number of variables considered in learning the region (the number of variables found in the restrict step of *RegionMMHC*). Recall from the theoretical exploration of the two techniques, that under a few assumptions the relative time (local / global) of the techniques for learning the skeletons should be approximately the proportion of the entire network the region encompasses, $\approx \frac{|R|}{|\mathcal{V}|}$. Figure IV-5 explores this relationship by plotting the relative number of nodes that are considered in reconstructing the region, $\frac{|R|}{|\mathcal{V}|}$, versus the relative time spent building the skeleton of the graph (this does not include the time orienting the graph). The points plotted for increasing time and nodes considered represent increasing region depths. This figure

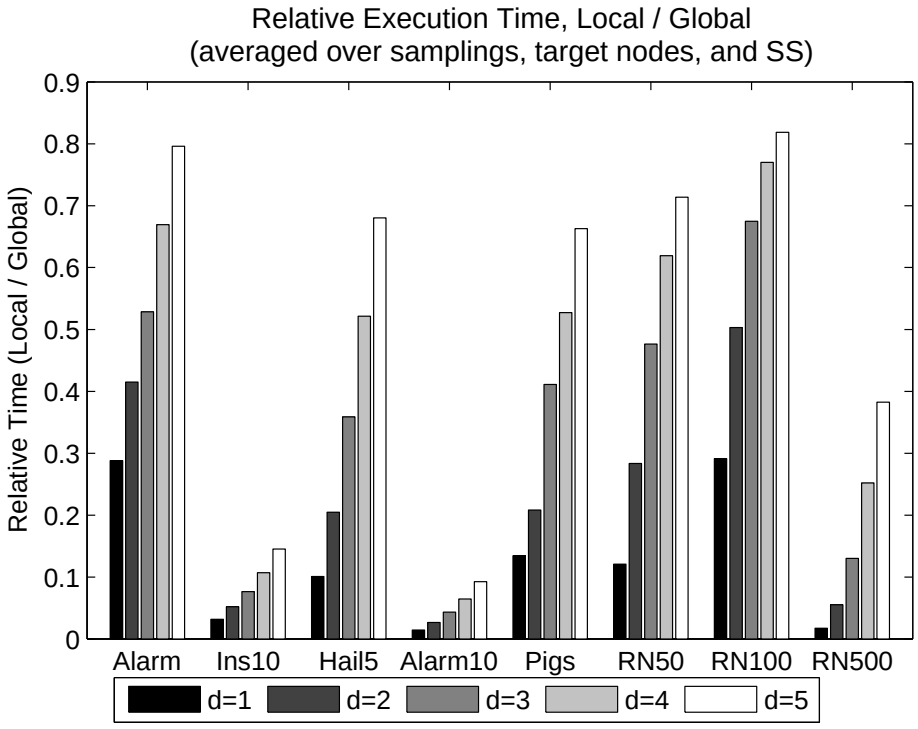


Figure IV-4: The relative execution time (local / global) averaged over data set samplings, target nodes, and sample size (SS) plotted for each network and region depth.

indicates an approximately linear relationship between the number of nodes and time in learning a region as suggested in the theoretical comparison.

Each network in the evaluation has inherent properties that might make one "harder" to learn than another. Figures IV-4 and IV-5 illustrate this point, where each network has different pattern between the local and global techniques.

Quality of Reconstructed Region

In addition to the timing results of the two methods, the quality of the discovered regions must be evaluated. For this evaluation, we use the metric of Structural Hamming Distance (*SHD*) as a measure of quality (Tsamardinos *et al.*, 2006b). The Structural Hamming Distance compares the structure of the corresponding equivalence classes of the learned and original network regions. Two networks in the same equivalence class capture the same independencies and dependencies; consequently are statistically indistinguishable. The equivalence class is represented as a partially

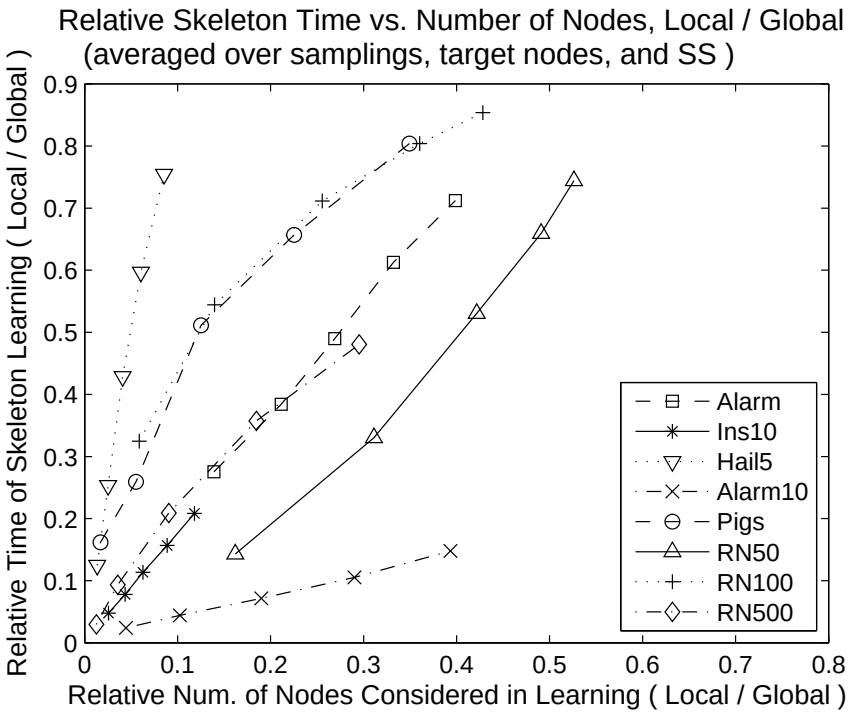


Figure IV-5: The relative number of nodes ($\frac{|R|}{|\mathcal{V}|}$) is plotted versus the relative time (local / global) in learning the skeleton of the region. The points are averaged over data set samplings, target nodes, and sample size and plotted for each network and increasing depth.

directed acyclic graph and referred to as a DAG pattern. The DAG pattern is made up with directed edges when all DAGs in the class agree on the orientation and undirected edges when the orientation differs between the DAGs. The distance between the two DAG patterns is the number of the following operators required to make the DAG patterns match: add or delete an undirected edge, and add, remove, or reverse the orientation of a directed edge (see Tsamardinos *et al.* 2006b for more details and use of this metric). Thus, an algorithm will be penalized by a score increase of one for learning a DAG pattern with an extra un-oriented edge and by one for not orienting an edge that should have been oriented. The reason for using a metric on DAG patterns is so we do not penalize an algorithm for not differentiating between statistically indistinguishable orientations.

For the global approach and the original network, the graphs were first converted to the associated DAG patterns then the region of interest around each of the 10 randomly chosen variables out to the given depth is extracted from the DAG pattern for comparison. For the local approach, the DAG

found by the search-and-score procedure before pruning (to depth $d+1$) is converted to the DAG pattern, then the region of depth d is extracted around each node of interest. The *SHD* is calculated from the region's DAG pattern. The results in terms of structural quality of the regions is shown in Table IV-3. The values used to evaluate the two methods in Table IV-3 are the relative difference between the local and global approach (calculated as local *SHD*- global *SHD*). The *SHD*'s relative difference for the extracted region for each network, sample size and depth is the difference between the mean *SHD* of the local and global methods. For each the mean *SHD* is averaged over the 10 random target variables and the 5 sampled data sets. Note, the relative comparison between the two methods is given as a difference, rather than a ratio as in the timing results presented above, because for several cases specifically the PIGS network with 5000 sample the global approach reconstructs the network perfectly. The exact reconstruction results in a *SHD* of 0.0 and the ratio diverges. When the reported relative difference between the methods is negative then the local method resulted in fewer errors and when positive the global approach had fewer errors.

Table IV-3: Structural Quality Results for the local and global approaches. The local and global structural quality is measured by Structural Hamming Distance (SHD). The global and local *SHD* is averaged over the 5 data samples and 10 random target nodes. The relative quality of the two approaches is given by the difference in *SHD* (local - global) and is reported for each network, sample size and depth of region. A negative relative *SHD* indicates fewer errors by the local approach; whereas, a positive relative *SHD* indicates fewer errors by the global approach. The local method is found to differ significantly from the global approach only for Pigs at sample size 5000 for depths 3, 4, and 5 (using permutation testing and alpha = 0.05) indicated with the ‡.

		Relative Quality (Local - Global)				
Data	SS	d=1	d=2	d=3	d=4	d=5
Alarm	500	−0.34	−0.84	−0.10	1.20	−1.40
Alarm	1000	−0.12	−0.50	−1.14	−1.04	0.42
Alarm	5000	−0.24	−1.20	−0.50	0.12	0.24
Ins10	500	−0.08	−0.82	−0.48	−0.38	−1.06
Ins10	1000	−0.46	−1.50	−1.92	−1.64	−1.34
Ins10	5000	−0.40	−2.14	−3.16	−3.86	−5.40
Hail5	500	−0.08	−0.32	−0.44	−0.94	−0.72
Hail5	1000	−0.22	−0.28	−0.86	−1.36	−1.66
Hail5	5000	0.08	0.42	−0.76	0.56	1.18
Alarm10	500	−0.30	−0.64	−0.18	−0.50	−0.94
Alarm10	1000	−0.38	−1.14	−1.66	−2.32	−2.48
Alarm10	5000	−0.52	−1.32	−2.26	−1.54	−0.78
Pigs	500	0.00	0.00	0.34	1.26	2.18
Pigs	1000	0.00	0.06	0.24	1.96	6.52
Pigs	5000	0.00	0.12	0.16‡	1.56‡	3.70‡
RN50	500	−0.08	0.00	−0.86	−1.46	−1.44
RN50	1000	−0.10	−0.50	−1.88	−3.38	−3.26
RN50	5000	0.04	0.22	−0.06	−0.20	0.06
RN100	500	−0.32	−0.58	−1.06	−0.64	−0.58
RN100	1000	−0.30	−0.42	−0.66	−0.72	0.30
RN100	5000	−0.26	−0.10	−0.32	−0.20	0.00
RN500	500	−0.26	−0.18	−1.48	−4.54	−3.48
RN500	1000	−0.16	−0.08	−0.70	−1.08	−5.12
RN500	5000	−0.24	−0.30	0.14	−1.78	−4.56

From Table IV-3, in all but 28 of the 120 cases the local method has fewer errors than the global methods (15 of those 28 cases are on the Pigs network where the global method performs exceptionally well). Overall, the mean difference between the local and global approaches averaged across all parameters is -0.46. The local methods is averaging fewer errors. Recall that the *SHD* measure is increased by one for each additional, missing, or mis-directed edge, therefore intuitively this difference represents less than one additional/missing edge on average. The greatest difference

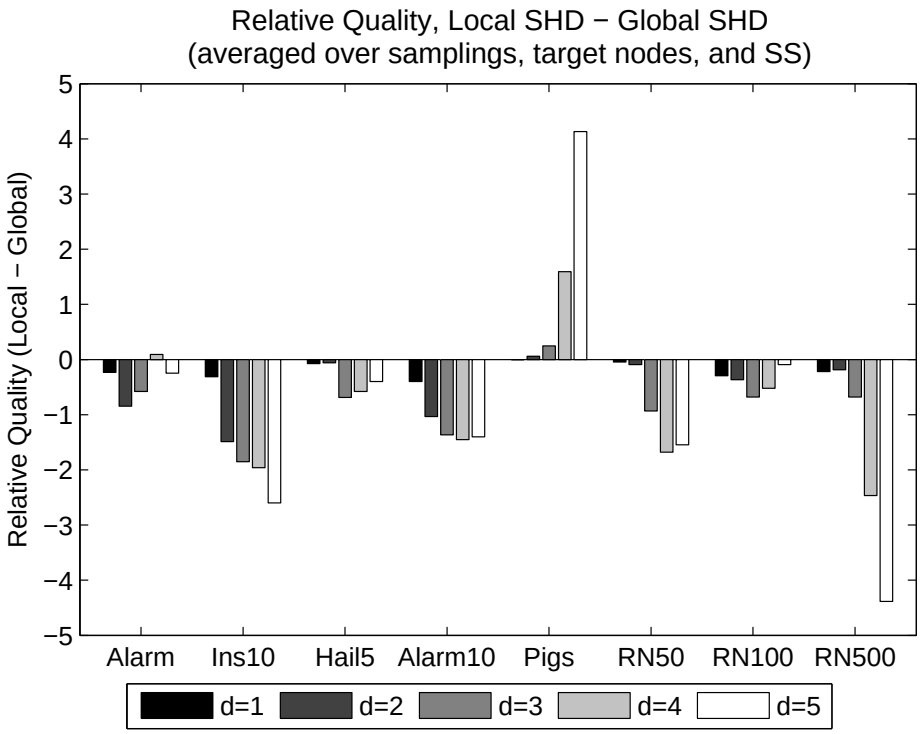


Figure IV-6: The relative structural quality of the learned regions (local - global) averaged over data set samplings, target nodes, and sample size (SS) plotted for each network and depth of region.

in terms of quality performance is for the PIGS network (see Figure IV-6. *MMHC* learns the PIGS network completely correctly for sample size of 5000, therefore all extracted regions are correctly found as well. Additionally, for 1000 samples on the PIGS network *MMHC* learns the $d = 1$ correctly for the 10 random target variables chosen. In terms of quality of the network region constructed the local method is found to differ significantly from the global approach only for PIGS at sample size 5000 for depths 3, 4, and 5 (using permutation testing and alpha = 0.05).

IV.5.2 Results versus another Local Learning Method

In this section, we now compare our local approach to that of Peña's *AlgorithmGPC* method (Peña *et al.*, 2005). This approach uses the same idea of recursive identification of the parent and children set out to some depth $d+1$ before pruning and edge orientation. It is in this final step, that

RegionMMHC and *AlgorithmGPC* differ. *RegionMMHC* uses PC-orient rules to identify collider edges and orient edges. *RegionMMHC* returns a PDAG or the equivalence class of the region.

We ran *RegionMMHC* on the same data sets measuring the elapsed time and the quality of the structure learned (because *RegionMMHC* return's an equivalence class the *SHD* was calculated directly from this structure). *RegionMMHC* used the standard 5% statistical threshold for tests of independence. *RegionMMHC* is a windows executable of C++ code, the computations were run on Pentium Xeon, 3.2GHz with 2GB RAM running Windows.

In general, *AlgorithmGPC* was more efficient when it completed the learning task. However, there were several learning tasks that *RegionMMHC* did not complete its computation: PIGS with 5000 sample, HAIL5 at all sample sizes, and RN500 at all sample sizes). Note, the comparison of computation time between the two methods comes with several caveats: the two methods were run on different machines with different hardware, with different operating systems, and different programs (Matlab for *RegionMMHC* and compiled C++ for *AlgorithmGPC*). A complete comparison of the relative execution time of *AlgorithmGPC* over *RegionMMHC* is presented in Appendix C.I Table C-1.

In terms of structural quality of the returned region, *AlgorithmGPC* had in general, more structural errors than *RegionMMHC*. The structural quality is variable depending on network, sample size, and depth of region to be learned. Out of the 85 different cases (averaged over target nodes and data set samples) where *AlgorithmGPC* completed, in only 9 cases did *AlgorithmGPC* have higher structural quality. A complete comparison of the relative structural quality of *RegionMMHC*'s SHD - *AlgorithmGPC*'s SHD is presented in Appendix C.I Table C-2.

IV.5.3 Learning Regions with 100,000 variables

A final feasibility study was included to show that *RegionMMHC* is able to scale to networks with hundreds of thousands of variables. A random network with 100,000 variables was created in the same manner as the other random networks. Five target nodes were selected randomly and a region of depth, $d = 3$ was learned on a data set of 1000 samples. The average region consisting

of 43 variables was found in 8426 seconds. Errors in finding the correct region were most often the result of missing an edge in building the skeleton. Running *MMHC* on this size of network would be extremely resource and computation expensive (as a comparison *MMHC* running on a random network of 10,000 variables takes over 5 days of computing time).

Additionally, problems were created with 10,000, 25,000, 50,000, and 75,000 variables to examine the behavior with increasing numbers of variables. Again, 5 random target nodes were selected and regions of depth $d = 3$ were learned. The time to learn the region increased, as expected, with the number of variables in approximately a linear relationship. The number of errors in the region also increased with the number of variables.

IV.6 Discussion of Results

The proposed local method was compared to a state-of-the-art global BN learning algorithm in terms of the time required to learn a region as well as the quality of the reconstruction. The local method was found to be faster on average on all but one case out of 120 combinations of network, sample size and depth. The average relative execution time for the local approach versus the global approach was 12% $\pm$ 10%, 22% $\pm$ 16%, 34% $\pm$ 21%, 44% $\pm$ 25%, and 54% $\pm$ 27% for regions of depth d = 1, 2, 3, 4, and 5 respectively. The computational gains are greatly affected by the difficulty of the learning task, the connectivity of the network, the amount of sample given to the learning procedure, and the depth of region to be learned (consequently the coverage in terms of percentage of the entire network) . In the worst case, the local method reduces to the global method (i.e., the region depth was large enough that approximately all of the variables in the network were considered for the region).

Of interest is that the learned region of the local approach has comparable quality to the global method. The quality of the region was better for the global approach only for the PIGS network with 5000 samples where the global approach learned the network perfectly. In summary, the local approach is able to achieve a speed-up in time while not (in general) reducing the quality of the

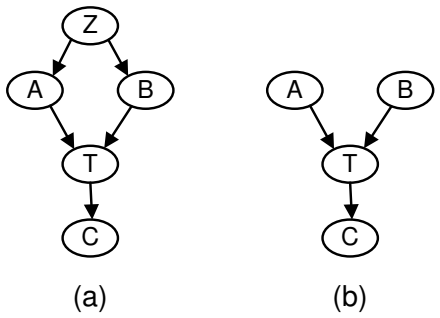


Figure IV-7: Interpreting d-separations from Global Network and Regions: (a) shows a global network graph and (b) shows the local region of depth $d = 1$ extracted from the global network. The local region structure matches that of the global network however, the d-separations read from the global network may not match those read from the region. For example, the global network has A and B d-connected conditioned on the empty set while the local region has A and B d-separated by the empty set.

region learned. These results suggest that the global viewpoint may not be needed for edge direction using the search-and-score procedure.

The comparison between local and global learning of regions was performed with algorithms both based on the same basic processes: the *MMPC* and search-and-score orientation methodologies. Consequently, the algorithms share many of the same assumptions and properties that aid in comparing the two methods. However, this common algorithmic base restricts the generalization of the results presented. The additional experimentation adds a comparison between the local approach (*RegionMMHC*) and another method aimed at learning local structures (*AlgorithmGPC*). When considering *RegionMMHC* and *AlgorithmGPC*, the *AlgorithmGPC* method learns a region in less time when the method completes however, for the larger networks and sample sizes the method may not run. In terms of quality, the *AlgorithmGPC* method in general learns region of lower quality then those of *RegionMMHC*.

IV.6.1 Future Work

The approach discussed aims to return the local subgraph of the Bayesian network. The information drawn from the region should be interpreted with care. Consider the two networks shown in Figure IV-7; Figure IV-7(a) shows the global network to consider and Figure IV-7(b) presents the

Algorithm 9 *RegionDsepMMHC* Algorithm

procedure REGIONDSEPMMHC($\mathcal{D}$, T, d)
Input: data $\mathcal{D}$, target node T, distance d
Output: $R_{\mathcal{G}}(T, d)$
% Restrict
  $nodes_0 = T$, $i = 1$
  while $i \leq d + 1$ **do**
    for every variable $X \in nodes_{i-1}$ **do**
      PC_X = MMPC$(X, \mathcal{D})$
      $nodes_i = nodes_i \cup PC_X$
    end for
    $i = i + 1$
  end while
  $\mathcal{D}_{\lceil +\infty}$ = data set only involving variables at $d + 1$ level.
  for every variable $X \in nodes_{d+1}$ **do**
    $PC_X = PC_X \cup$ MMPC$(X, \mathcal{D}_{d+1})$
  end for
% Search
  Starting from an empty graph perform Greedy Hill-Climbing with operators *add-edge*, *delete-edge*, *reverse-edge*. Only try operator *add-edge* $Y \rightarrow X$ if $Y \in PC_X$.
  DAG = the highest scoring DAG found, pruned around T to depth of $d + 1$.
  Return DAG
end procedure

local region of depth $d = 1$ extracted from the global network. While the network structure remains the same between the global structure and local region, the d-separations (or independences when considering a faithful network) read from the global network may not match those read from the local region in isolation. From the networks shown Figure IV-7(a) and (b), the global network has A and B d-connected when conditioning on the empty set while the local region shows A and B d-separated by the empty set.

This example illustrates that the d-separations read from the region graph are not guaranteed to correspond to those of the global graph. The definition of a region could be adjusted so that the d-separations read from the region graph correspond to those of the global graph. With this definition, an alternative version of the *RegionMMHC* method could be designed to return this graph (see method in Algorithm 9). This alternative method will return a region out to depth $d+1$ however, d-separations should be considered for those variables up to depth d and variables at depth $d + 1$ could only be considered in the conditioning set. Formally, the d-separation between any pair of variables X and Y such that $X, Y \in R_{\mathcal{G}}(T, d)$ of the region should be the same as the global network when the variables at the $d + 1$ depth, $\mathbf{Z} = \{z | z \in R_{\mathcal{G}}(T, d + 1) - R_{\mathcal{G}}(T, d)\}$, are included

in the conditional set and represent all global variables outside of the region $R_{\mathcal{G}}(T, d)$. In this representation, d-separations should not be read from variables at the $d + 1$ depth, these variables should only be used in a conditioning set.

Additional future work could focus on learning a different kind of region subgraph. The regions described here are one of an infinite kind of subgraphs that could be learned. In the *RegionMMHC* algorithm, variables are considered in a breadth-first, homogeneous expansion from the target, but the method is not dependent on this mode of exploration. Modifying *RegionMMHC* to change how the nodes are considered and expanded, can create many different algorithms that fit other desired subgraph arrangements. Additionally, an interesting new avenue of research is the possible use of techniques for combining local regions to form the complete global network.

IV.7 Conclusions

The emergence of extremely large data sets from a multitude of domains has exceeded the limits of most traditional BN learning algorithms. New techniques are necessary for handling the needs of these new data sets. In this Chapter, a new algorithm *RegionMMHC* for learning a region of a Bayesian network is presented. The learned region encapsulates a subgraph around a node of interest up to depth d. The results encourage future work in the area of learning local regions.

CHAPTER V

A STRATEGY FOR MAKING PREDICTIONS UNDER MANIPULATION

The first Causality Challenge competition (WCCI 2008 Causality Challenge, 2008) posted several causal discovery problems that require researchers to employ the full arsenal of state-of-the-art causal discovery methods, while prompting the development of new ones. The focus of this challenge is on predicting in the presence of manipulations performed by an external agent. The approach used the formalism of Causal Bayesian Networks to model and induce causal relations and to make predictions about the effects of the manipulation of the variables. Using state-of-the-art, under development, or newly invented methods specifically for the purposes of the competition, the approach addressed the following problems in turn in order to build and evaluate a model: (a) finding the Markov Blanket of the target even under some non-faithfulness conditions (e.g., parity functions), (b) reducing the problems to a size manageable by subsequent algorithms, (c) identifying and orienting the network edges, (d) identifying causal edges (i.e., not confounded), and (e) selecting the causal Markov Blanket of the target in the manipulated distribution. The results of the competition illustrate some of the strengths and weaknesses of the state-of-the-art of causal discovery methods and point to new directions in the field. An implementation of our approach is available at http://www.dsl-lab.org for use by other researchers.

V.1 Introduction

A principled submission to the first Causality Challenge tasks was developed (WCCI 2008 Causality Challenge, 2008). This publicly available challenge a part of the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008) had over 1400 submission by more than 30 teams. This challenge focused on making predictions in the presence of manipulations. When a variable is manipulated it is in a sense disconnected from its causes and consequently its predictive power may be affected. Methods should take into account the effect of the manipulations in developing predictive

models. However, many variable selection algorithms do not discover the cause-effect relationships between variables and the target. In an observational setting where the training and testing sets are obtained from the same "natural" distribution, the underlying mechanism is not required. In the challenge, the training and test sets are not necessarily identically distributed. The training set comes from the "natural" distribution, while different versions of the test set are used drawn from the "natural distribution" and other manipulated distributions.

In order to optimally predict the effects of manipulations on a system, one needs to induce a subset of the causal relations among the parts of the system. Three key characteristics of the challenge data sets led to the choice of Causal Bayesian Networks (CBN) as the formalism to model and induce causal relations and to make predictions about the effects of the manipulation of the variables: the data contain cross-sectional measurements, the generating causal models contain no feedback loops, and the definition of causality is stochastic. A CBN is a Bayesian Network where the edges have the additional semantics that they correspond to direct causal relations. Thus, a first major assumption in the analyses is that there exists a CBN that can represent the probability distribution of the data. This in turn implies assuming the Causal Markov Condition holds: every node X is probabilistically independent of its non-causal effects conditioned on its direct causes. An example of a graph of a CBN is shown in Figure V-1(a).

V.1.1 Theory for Making Predictions Under Manipulation

We will denote the variable to predict with the letter T (target). Let us denote the set of variables as $\mathcal{V}$ that is partitioned into observed variables included in the data $\mathcal{O}$, and unobserved variables $\mathcal{H}$. Single variables are denoted with capital letters or with V_i where i is an index and sets of variables with bold capital letters. Let $\mathbf{M}$ denote the set of manipulated variables. For the challenge it is assumed that $\mathbf{M} \subseteq \mathcal{O}$, i.e., there are no manipulated unobserved variables. We will denote with $P_{\mathbf{M}}(\mathcal{V})$ the joint probability distribution of variables $\mathcal{V}$ when the set of manipulated variables is $\mathbf{M}$. There were three different types of tasks in the competition, each requiring a different approach, that we now explain.

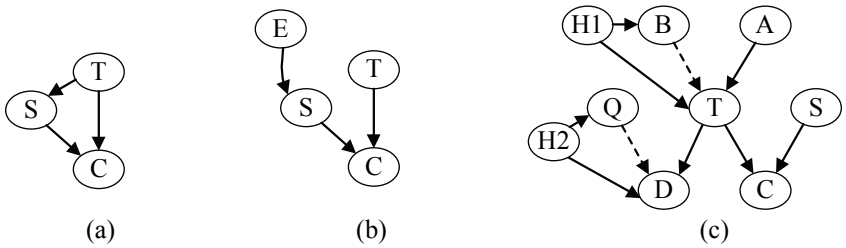


Figure V-1: Causal Bayesian Networks: The unmanipulated CBN graph, $G_\emptyset$, and CBN graph $G_{\{S\}}$ where S is manipulated, are depicted in (a) and (b). In (c), a network with a hidden variables $H1$ causing both B and T, $H2$ causing both D and Q, and dashed edges (when the marginal over the observed variables, $\mathcal{O}$, is considered) is shown.

Predictions under no manipulation

For this type of task, one could first estimate $P_\emptyset(T|\mathcal{V} \setminus \{T\})$. The estimation may be difficult and unreliable if the size of $\mathcal{V}$ is large. A Markov Blanket of T, $MB_\emptyset(T)$, for distribution $P_\emptyset$, is defined as a minimal set such that $P_\emptyset(T|\mathcal{V}\setminus\{T\}) = P_\emptyset(T|MB_\emptyset(T))$. In other words, a Markov Blanket contains the required information for optimal prediction of T, thus rendering the remaining variables superfluous and is the solution to the variable selection problem under some general conditions (Tsamardinos & Aliferis, 2003). Notice that in a CBN (by definition a minimal I-map, Pearl 1988), a $MB_\emptyset(T)$ corresponds to the parents, children, and spouses of T in the graph (Pearl, 1988, Sec. 3.3, Corollary 6). Based on the above, the approach for this task was to identify a Markov Blanket of T, $MB_\emptyset(T)$ then learn a predictive model using only these variables.

Predictions under known manipulations

In this case, we assume that there is a known subset of variables $\mathbf{M} \subseteq \mathcal{O}$ that are being effectively manipulated, i.e., their values are completely determined by the external agent, that we model with variable E. As in a typical supervised learning setting, one could attempt to learn a model for $P_\mathbf{M}(T|\mathcal{V} \setminus \{T\})$. According to Pearl (2000) and Spirtes *et al.* (2000), the joint distribution can be factorized as

$$P_\mathbf{M}(\mathcal{V}) = \prod_{V_i \in \mathcal{V}\setminus\mathbf{M}} P_\emptyset(V_i|Pa(V_i)) \cdot \prod_{V_i \in \mathbf{M}} P_\mathbf{M}(V_i|E)$$

where $Pa(V_i)$ are the parents (direct causes) of V_i and $P_{\mathbf{M}}(V_i|E)$ the manipulated distribution of a variable. From $P_{\mathbf{M}}(\mathcal{V})$ one could obtain $P_{\mathbf{M}}(T|\mathcal{V} \setminus \{T\})$ and solve the problem. However, this approach requires knowledge of the distributions of the manipulated variables $P_{\mathbf{M}}(V_i|E)$ that is not provided; in addition, it requires fitting the complete joint distribution of the variables that is computationally inefficient and prone to statistical errors.

Alternatively, we employ the concept of the Markov Blanket, to instead learn a model for $P_{\mathbf{M}}(T|MB_{\mathbf{M}}(T))$. If the causal graph is known, the $MB_{\mathbf{M}}(T)$ can be identified from it as follows. Let $G_{\emptyset}$ and $G_{\mathbf{M}}$ be the CBN graphs of the unmanipulated and manipulated distribution respectively. From Pearl (2000) and Spirtes *et al.* (2000), $G_{\mathbf{M}}$ results from $G_{\emptyset}$ by removing the direct causes of every variable $V_i \in \mathbf{M}$ and replacing them with an edge from an external agent performing the manipulations, E. An example is shown in Figures V-1(a-b) for $\mathbf{M} = \{S\}$. Intuitively, this is justified by the fact that the manipulated variables have no other causal dependence but with the external agent. Thus, $MB_{\mathbf{M}}(T)$ is a subset of $MB_{\emptyset}(T)$ with manipulated children and their corresponding spouses removed (if a node is a spouse via multiple children, it is removed only if all of them are manipulated). Even if $MB_{\mathbf{M}}(T)$ is known, $P_{\mathbf{M}}(T|MB_{\mathbf{M}}(T))$ should be *induced from observational data following* $P_{\emptyset}$. We now present the following theorem stemming again from the more general theory of probability invariance under manipulations by Spirtes *et al.* (2000) (proof in Appendix D.I):

Theorem V.1. *Let* $\langle G_{\emptyset}, P_{\emptyset} \rangle$ *be a CBN and* $\langle G_{\mathbf{M}}, P_{\mathbf{M}} \rangle$ *be the resulting CBN under manipulations of variables in* $\mathbf{M}$. *Suppose that* $T \notin \mathbf{M}$ *and also that there is no manipulated child* C *of* T *in* $G_{\emptyset}$ *with a descendant* D *in* $G_{\emptyset}$ *that is also in* $MB_{\mathbf{M}}(T)$. *Then,*

$$P_{\mathbf{M}}(T|MB_{\mathbf{M}}(T)) = P_{\emptyset}(T|MB_{\mathbf{M}}(T)).$$

In other words, when the theorem holds, we can learn an optimal model for predicting T in the manipulated distribution by learning $P_{\emptyset}(T|MB_{\mathbf{M}}(T))$ from data sampled from the unmanipulated distribution. The latter of course requires knowledge of $MB_{\mathbf{M}}(T)$ which is a subset of $MB_{\emptyset}(T)$. When the theorem does not hold, then predicting T using $P_{\emptyset}(T|MB_{\mathbf{M}}(T))$ is not theoretically guaranteed

to be optimal; however, the condition of the theorem is relatively strict and it is expected that it often holds in practice (of course, this claim requires further evaluation).

Notice the condition regarding the existence of a manipulated child of T and its descendant $D \in MB_{\mathbf{M}}(T)$ is important. Consider the network in Figure V-1(a), where the condition does not hold when S is manipulated, and the resulting network V-1(b). Then, we have:

$$P_{\emptyset}(T|MB_{\mathbf{M}}(T)) = \frac{P_{\emptyset}(T) \cdot P_{\emptyset}(S|T) \cdot P_{\emptyset}(C|S,T)}{\sum_t P_{\emptyset}(t) \cdot P_{\emptyset}(S|t) \cdot P_{\emptyset}(C|S,t)}$$

$$\begin{aligned} P_{\mathbf{M}}(T|MB_{\mathbf{M}}(T)) =& \frac{P_{\mathbf{M}}(T) \cdot P_{\mathbf{M}}(do(S)) \cdot P_{\mathbf{M}}(C|S,T)}{\sum_t P_{\mathbf{M}}(t) \cdot P_{\mathbf{M}}(do(S)) \cdot P_{\mathbf{M}}(C|S,t)} \\ =& \frac{P_{\emptyset}(T) \cdot P_{\mathbf{M}}(do(S)) \cdot P_{\emptyset}(C|S,T)}{\sum_t P_{\emptyset}(t) \cdot P_{\mathbf{M}}(do(S)) \cdot P_{\emptyset}(C|S,t)}, \end{aligned}$$

where $P(do(S))$ follows Pearl's nomenclature denoting the probability of S being manipulated to obtain a specific value and if V is not manipulated then $P_{\mathbf{M}}(V|Pa(V)) = P_{\emptyset}(V|Pa(V))$ (see Pearl 2000 for explanation and discussion). In general the top quantity takes different values from the bottom one; when the theorem does not hold, we could still fit a model from the observational data and use it in the manipulated distribution, if information about the distribution of the manipulations is provided.

From the above discussion, to identify $MB_{\mathbf{M}}(T)$ one needs to know both $MB_{\emptyset}(T)$ and the edge orientation in that graph neighborhood. So, we first attempt to learn the causal network from the training data and then derive $MB_{\mathbf{M}}(T)$ by deleting the appropriate edges. There are two potential problems with this approach, even if the network is induced perfectly. First, there may be several statistically indistinguishable networks that fit the data equally well. For example, the models $T \rightarrow X$ and $T \leftarrow X$ are indistinguishable with the $P_{\emptyset}$ distribution. We do not have a solution to this problem, which implies that some manipulated children of T may be falsely included in $MB_{\mathbf{M}}(T)$. The second problem with inducing $MB_{\mathbf{M}}(T)$ is the existence of hidden variables $\mathcal{H}$. The induced networks regard the marginal distribution over variables in $\mathcal{O}$. In Figure V-1(c) an example is shown,

where $\mathcal{H} = \{H1, H2\}$ and the dashed edges appear in the network capturing the marginal over $\mathcal{O}$. True causal parents and spouses (A and S) belong in $MB_{\mathbf{M}}(T)$ even when they are manipulated, but confounded parents and spouses (B and Q) should be removed when manipulated. In Section V.2.5 we present newly developed methods to address this issue.

For this type of task, our general strategy was to first learn $MB_{\emptyset}(T)$, then orient the edges in that neighborhood to identify a candidate $MB_{\mathbf{M}}(T)$; subsequently, evidence about possible confounding is obtained to further remove variables if necessary (details are described in Section V.2.5). Finally, a predictive model using only the variables in the estimated $MB_{\mathbf{M}}(T)$ was learned.

Predictions under unknown manipulations

For these tasks, the set $\mathbf{M}$ of manipulated variables is unknown. The only nodes that always belong in $MB_{\mathbf{M}}(T)$ for any $\mathbf{M} \subseteq \mathcal{O}$ are the parents of T. Thus, the safest bet for avoiding to include irrelevant or even misleading variables (depending on the sort of manipulations) in predicting T is to build a model $P_{\emptyset}(T|Pa(T))$, where $Pa(T)$ are the (non-confounded) parents (direct causes) of T.

V.2 General Steps of the Strategy

In order to identify the Markov Blankets to build the predictive models, several different algorithms were used in our procedure. Figure V-2 summarizes the general approach followed while the subsequent sections (noted in the figure) describe the process in more detail. The first step in our strategy is to identify the $MB_{\emptyset}(T)$. If there are no manipulations in the test set distribution, an SVM model is constructed using the variables in $MB_{\emptyset}(T)$ (Section V.1.1). If there are manipulations, a set of additional steps are taken to orient the edges in $MB_{\emptyset}(T)$ and identify non-confounded edges. Combining all this information, a set of variables is selected, either $MB_{\mathbf{M}}(T)$ or the non-confounded parents of T, depending on whether the manipulations are known or not, respectively (Section V.1.1 and Section V.1.1). The final set of variables is again used to construct an SVM model for predicting the cases in the manipulated test set.

Our method is publicly available online at http://www.dsl-lab.org. In order to fully automate the procedure, the released code has been modified from that used during the challenge. Wherever

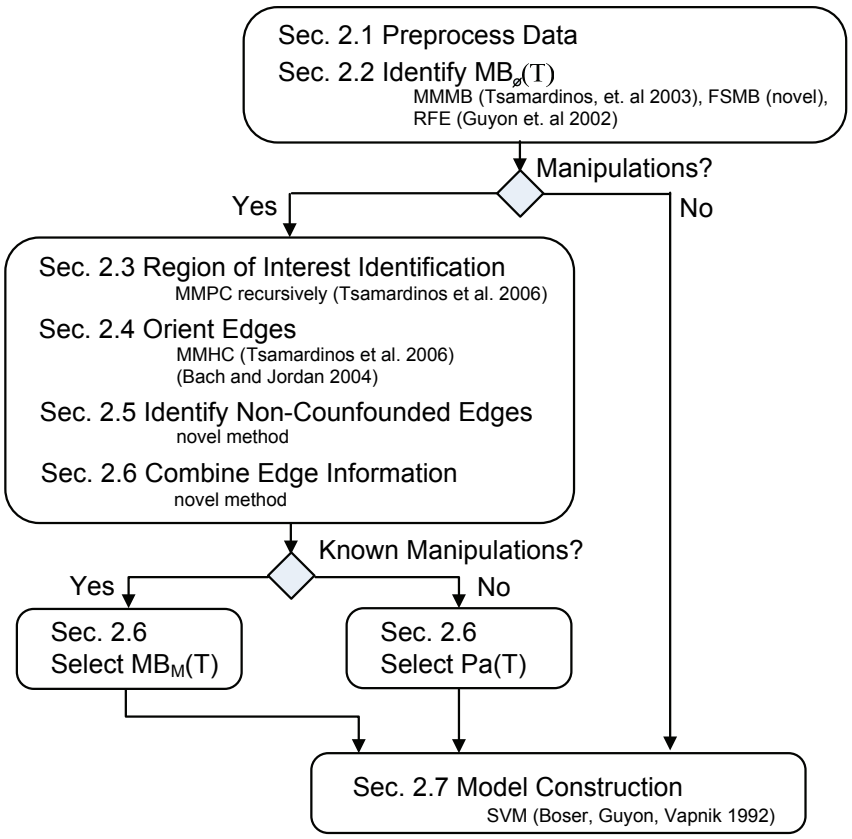


Figure V-2: Diagram illustrating the general steps of our method including the individual algorithms used.

a difference between the competition and the released code exists, we note it in the text. The code implementing the high-level strategy is released, although some of the employed algorithms are only available as executable Matlab p-files.

V.2.1 Preprocessing

The data sets used in the challenge represented real world problems that required preprocessing which was tailored for each data set. For the REGED data set each variable was normalized so its mean was zero and standard deviation was one. For the SIDO data set, the variables were binary and no preprocessing was performed. For the CINA data set, variables that were not binary were treated as continuous and normalized as above; binary variables were all set to values of zero and one. For the MARTI data set, the calibrant variables were used as an indication of the position-

dependent noise on the chip. For each training example, we fitted a 2D cubic spline to the values of the calibrants and then used the spline to obtain the correlated noise level at the chip location of each variable. The estimated noise was then subtracted from the value of each variable for that training sample.

V.2.2 Identifying $MB_{\emptyset}(T)$

Once the initial data sets have been preprocessed, the next step of our procedure was to identify the $MB_{\emptyset}(T)$. Algorithms such as HITON and MMMB (Aliferis *et al.*, 2003a; Tsamardinos *et al.*, 2003c; Aliferis *et al.*, 2009a,b) rely on statistical tests of conditional independence. A basic assumption of these and similar methods is that if a variable is a neighbor of the target, then it will have a detectable pairwise association with the target. The general case of this assumption is that the Faithfulness Condition (Spirtes *et al.*, 2000) holds in the causal network. However, there were no such guarantees in the problems of the competition. Thus, there could exist strong multivariate associations with the target (e.g., parity functions) whose participating variables have no detectable pairwise association with T. To address this problem we use our newly proposed algorithm called Feature Space Markov Blanket, FSMB (Brown & Tsamardinos, 2008).

Feature Space Markov Blanket (FSMB)

The FSMB algorithm is described in detail in Chapter III, but a short description is presented here. FSMB explicitly constructs a set of features, namely all the products among the variables up to a given degree d. For two variables and $d = 2$, these are V_1, V_2, V_1^2, V_2^2 and V_1V_2. It then runs HITON to find the Markov Blanket of T in this feature space. While straight-forward, this strategy does not scale up to data sets of practical sizes. A key idea in FSMB is to first learn an SVM model using a polynomial kernel that implicitly maps to this feature space consisting of all possible monomials up to a given degree d. We expect that if a feature is given a small absolute weight by the SVM, then it probably has a small association with T and there is no need to compute it and feed it to HITON. FSMB is enriched with a heuristic search to efficiently construct only the top-weighted features of the SVM model, before passing them to HITON.

This heuristic search procedure is now presented in more detail. The following standard SVM notation is used in this section; let v_k denote the predictor vector k in the data and $t_k \in \{-1, 1\}$ denote its class. Assume the use of a trained soft-margin, 1-norm SVM with full polynomial (heterogeneous) kernel $K(v_k, v_j) = \Phi(v_k) \cdot \Phi(v_j) = (v_k \cdot v_j + 1)^d$, where d is the degree of the kernel and the Lagrange multiplier vector is denoted a. The SVM model is stored as the Lagrange multipliers and support vectors, rather than explicitly constructing the feature and weight vectors of the decision function due to the large number of possible features.

In order to identify the top weighted-features without explicitly reconstructing the entire weight vector, bounds on the weights are found and updated through the search and feature construction process. Let $s_{i,j}$ be the sum of squares of the weights of all features (monomials in polynomial-kernel feature space) that involve variable i and are exactly of degree j. Then, similarly to the corresponding result for the Recursive Feature Elimination (Guyon *et al.*, 2002b) we can show that:

$$s_{i,j} = \binom{d}{j} \sum_{k=1,l=1}^{n} a_k a_l t_k t_l (H(v_k, v_l) - H(v_k^{\setminus i}, v_l^{\setminus i}))$$

where $v_k^{\setminus i}$ denotes vector v_k with the i component removed and $H(v_k, v_l) = (v_k \cdot v_l)^j$. Notice that $s_{i,j}$ is a bound on the square of the largest weight of any feature that can be constructed with variable i having degree exactly j.

Let us call this bound $b_{i,j}$ and initially set it to $s_{i,j}$. We use this bound to heuristically select some features Φ_q, for an indexing q of all features, to explicitly construct and calculate the corresponding weight w_q. We expect that the features with the largest weights probably increase the corresponding $b_{i,j}$'s to which they contribute. So, we select the degree l of monomials exhibiting the largest bound $l = \text{argmax}_j b_{i,j}$ and the variables V_i in that level with the largest bounds $b_{i,l}$. For example, let us assume that $l = 2$ and the variables V_1 and V_2 have the largest bounds $b_{1,2}$ and $b_{2,2}$. Then, we explicitly construct the features V_1^2, $V_1 V_2$ and V_2^2 and calculate their corresponding weights using the formula

$$w_q = \sum_{k=1}^{n} a_k t_k \Phi_q(v_k).$$

For example, if we denote with $v_{r,z}$ the value of the r-th training example for variable z, then the weight corresponding to constructed feature V_1V_2 equals $\sum_{k=1}^{n} \sqrt{2} a_k t_k v_{k,1} v_{k,2}$. The weight w_q of each explicitly constructed feature is then subtracted from the corresponding bounds: $b_{i,j} = b_{i,j} - w_q^2$. Thus, $b_{i,j}$ always maintains the sum of the squared weights of the remaining features, not yet constructed, involving variable i of degree exactly j. A stopping criterion can determine when the bound on the remaining weights is small enough to stop the explicit calculation of the weights. Preliminary experiments showing the time-efficiency and quality of the algorithm are presented in Brown & Tsamardinos (2008) and Chapter III.7.

Implementation of Identifying $MB_\emptyset(T)$

The MMMB algorithm (using the χ^2 test for conditional independence based on the G^2 statistic for discrete data and Fisher's z-test for continuous or mixed data) was employed to obtain a first approximation of the Markov Blanket (Tsamardinos *et al.*, 2003c).

To estimate how good of an approximation we obtained, we employed other variable selection algorithms and constructed models using all variable sets output (see Section V.2.7 for details on our procedure of building and evaluating the models). Specifically, we build models using as variable sets the output of MMMB, FSMB, RFE (Guyon *et al.* 2002b, run using the same kernel parameters as FSMB) and all variables. If all sets exhibited similar predictive cross-validated performance (judged manually), we accepted MMMB's output as a good approximation of $MB_\emptyset(T)$. Otherwise the better performance of RFE or FSMB, indicates important variables were missed and checked the output of FSMB for additional variables participating in strong multivariate associations. If that was the case, the interaction terms and constructed features were added as part of our Markov Blanket for all subsequent steps to use[1].

At this point, we considered that we have obtained a $MB_\emptyset(T)$ that could be used for optimal prediction under no manipulation, and is a superset of the Causal Markov Blanket in any manipulated distribution (plus false positives depending on the type of manipulations).

[1]In the released code, FSMB's constructed features are always included in the Markov Blanket, if they contain variables not participating in the output of MMMB.

V.2.3 Reducing the Size of the Problem to a Region of Interest

The previous step identifies the participants in the $MB_{\emptyset}(T)$. However, the methods employed do not indicate which variables are parents and which are children, i.e., the orientation of the edges in the $G_{\emptyset}$. This is necessary to be able to filter out the manipulated children and their parents and obtain $MB_{\mathbf{M}}(T)$. Unfortunately, many state-of-the-art methods for orientation are unable to run on problems of the size of the tasks in the competition.

To overcome the efficiency problem, we attempted to reduce the size of the problems by identifying the variables *at most three edges away from* T in $G_{\emptyset}$. Therefore, rather than learn the entire global network, we focus on a smaller region engulfing the target variable. This type of learning became possible with the invention of local causal structure-learning methods such as Grow-Shrink (Margaritis & Thrun, 1999) and MMPC, where MMPC returns the parents and children of T in a network $G_{\emptyset}$ (Tsamardinos *et al.*, 2003c). The idea of learning regions (subgraphs) of arbitrary size was first presented in Tsamardinos *et al.* (2003b). The variables in the region are identified through recursive application of a local neighborhood identification method (MMPC using the default parameter settings, Fisher's z test and χ^2 test on continuous and discrete data respectively) in a breadth-first search then, the graph is oriented as described in Section V.2.4.

Restricting our attention to a region may reduce the number of edges that can be oriented. That is, it is possible for remote parts of the network to lead to orientation of edges close to or involving T. Preliminary experiments (see Chapter IV) we have conducted however, indicate that in many typical networks this effect is not severe and the edges in the region can be oriented as well as when using the full network. The idea of reconstructing a region of interest of limited depth around T to help orient the Markov Blanket edges has also appeared in Bai *et al.* (2008).

The choice of a region of depth three is explained thusly; implicitly (in search-and-score methods) or explicitly (constraint-based methods) v-structures are crucial in orientation. A v-structure occurs when the subgraph $X \rightarrow T \leftarrow Z$ exist in the true unknown graph but the edge $X - Z$ is not. To determine from data that $X - Z$ is absent we need to make sure that we have conditioned on a subset of their parents. Thus, to identify a v-structure $X \rightarrow T \leftarrow Z$ we need the parents of X and Z that

are two edges away from T. The method we present in Section V.2.2 requires v-structures among the parents of T, thus forcing us to induce a region of depth three.

V.2.4 Identifying and Orienting Edges

In this step, we run standard Bayesian Network learning algorithms on the data projected on the variables of the restricted region found in the previous step. For the case of binary data, MMHC with the default parameter settings and a χ^2 test was employed to find a high scoring network; in extensive experimentation MMHC was deemed one of the best such learning algorithms (Tsamardinos *et al.*, 2006b). For the case of continuous or mixed data, the kernel generalized variance scoring metric of Bach & Jordan (2002), with $\kappa = 0.01$ and $\sigma = 1$, was used with a greedy hill-climbing search to learn the structure. In Bach & Jordan (2002), the variable distribution is assumed Gaussian in feature space, mapped implicitly by a kernel function. This method is able to work on combinations of discrete and continuous variables and performed well compared to other algorithms and approaches targeting continuous or mixed data as shown in Fu (2005). The final structures were converted to their corresponding PDAGs with the compelled edges identified. A compelled edge $X \rightarrow T$ provides evidence (under the Faithfulness Condition) that either X causes T, or (inclusive) X and T are confounded by a hidden variable.

V.2.5 Dealing with Confounded Variables

To deal with hidden variables and identify confounded parents of T, or confounded spouses of T we first tried the FCI algorithm (Spirtes *et al.*, 2000). Unfortunately, FCI could not scale up even to the reduced region found (FCI was run with version 4.3.9 of the Tetrad Project n.d.). It also failed to run even when we input several constraints to make it more efficient and specifically, to constrain the edges to the ones found by the previous step.

We then turned to the method of Mani *et al.* (2006) to identify a Y-structure involving a quadruple of the variables; see Figure V-3(a) for such a structure. If a Y-structure faithfully captures the marginal of the four variables, *then edge $C \rightarrow D$ has to be causal*, i.e., there can be no hidden

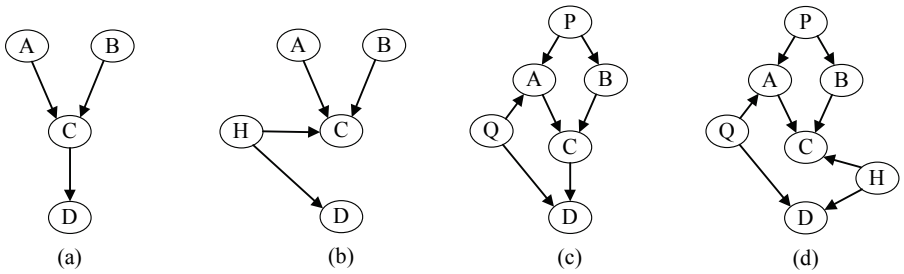


Figure V-3: Four example networks to explain the Y-structure analysis.

confounder of C and D, as shown in Figure V-3(b). If Figure V-3(b) was the case, A and D would be dependent given C and so their marginal would not faithful to Figure V-3(a). There is no causal claim for the other two edges in the graph.

We found this idea interesting but did not apply the algorithm as given by Mani *et al.* (2006) because the conditions to identify such a structure are restrictive (e.g., A and B need to be unconditionally independent). Instead, we extended the general idea to identify causal edges in more general settings, where the pairs A and B, or A and D may be conditionally independent instead of unconditionally, such as in Figure V-3(c) (this is mentioned as future work in Mani *et al.* 2006). We proved (proof omitted for scope) and implemented a test based on the following proposition:

Proposition V.1. *Let $\mathcal{V} = \mathcal{O} \cup \mathcal{H}$ be a set of variables, $\mathcal{O} \cap \mathcal{H} = \emptyset$; $P(\mathcal{V})$ is faithful to a CBN $\langle G, P \rangle$ and $I(X;Y|\mathbf{Z})$ denotes independence of X and Y given the conditioning set $\mathbf{Z}$ and $\neg I(X;Y|\mathbf{Z})$ denotes dependence. For the distinct variables $A, B, C, D \in \mathcal{O}$ when the following conditions hold:*

1. $\forall \mathbf{S} \subseteq \mathcal{O}, \neg I(A; C|\mathbf{S})$
2. $\forall \mathbf{S} \subseteq \mathcal{O}, \neg I(B; C|\mathbf{S})$
3. $\forall \mathbf{S} \subseteq \mathcal{O}, \neg I(D; C|\mathbf{S})$
4. $\exists \mathbf{Z}_1 \subseteq \mathcal{O}, I(A; B|\mathbf{Z}_1)$
5. $\neg I(A; B|\mathbf{Z}_1 \cup \{C\})$
6. $\exists \mathbf{Z}_2 \subseteq \mathcal{O}, I(A; D|\mathbf{Z}_2)$ *and* $C \in \mathbf{Z}_2$

then, there is a causal path $C \to \ldots \to D$ in G, where the intermediate variables belong in $\mathcal{H}$ (are hidden).

We call this set of conditions collectively the *Y-test for the variables A, B, C, and D*. In our implementation, we apply the Y-test for every quadruple of distinct variables A, B, C, D in the

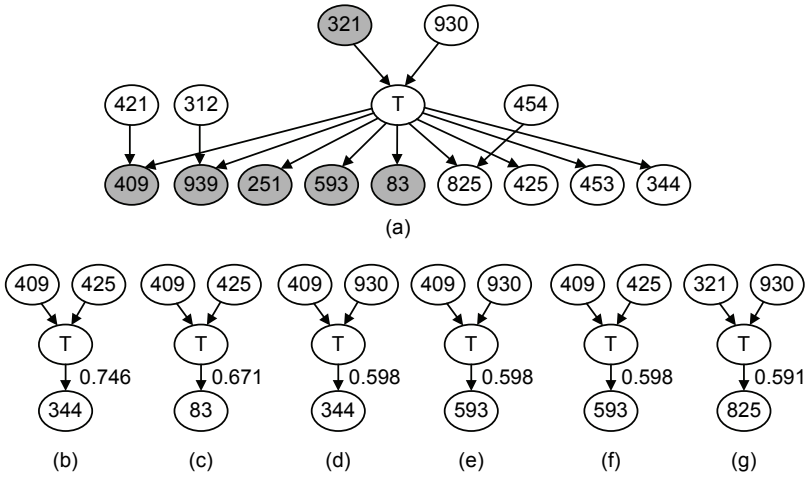
region of interest around T[2]. If all conditions (1) - (6) are satisfied then we considered the edge $C \rightarrow D$ as causal and without possible confounding. We applied the Y-test only once per quadruple of variables and reused cached results for improved efficiency as follows: If an edge $A-C$ (ignoring the direction) exists in the region of interest then $\forall \mathbf{S} \subseteq \mathcal{O}, \neg I(A; C|\mathbf{S})$, or MMPC would have discovered a d-separating set for A and C. Thus, condition (1) of the proposition holds. Similarly, if the edges $B - C$ and $C - D$ exist in the region of interest, the quadruple passes the first three conditions. If the edges $A - B$ and $A - D$ are not in the region of interest, it implies that MMPC has discovered subsets $\mathbf{Z}_1$ and $\mathbf{Z}_2$ that d-separate the two pairs of variables respectively: condition (4) and the first part of (6) also hold. Condition (5) is checked with an additional test of independence, using the specific $\mathbf{Z}_1$ found by MMPC when removing the edge $A - B$. Finally, it is checked whether $C \in \mathbf{Z}_2$, the subset found by MMPC when removing the edge $A - D$.

Multiple applications of the Y-test for different quadruple of variables may provide conflicting information for an edge $C \rightarrow D$. We devised two weighting schemes to rank the strength of evidence a single Y-test provides. First, a value was calculated as the minimum p-value returned by the independence tests of conditions (4) and (6). Let this value be referred to as the *p-score* of the Y-test. This value represents the closest the independence conditions (4) and (6) were to failing to pass the threshold for accepting dependence. Second, a ratio of the BDeu score of the Y-structure (including the nodes in the conditioning sets) to the BDeu score of an empty DAG was assessed. In preliminary tests on known networks, the BDeu score metric was not consistently informative; therefore, the p-score was used in further analysis.

V.2.6 Combining Information to Identify $MB_{\mathbf{M}}(T)$

We used the PDAG at the end of Section V.2.4 to obtain the orientation of some edges and the method of Section V.2.5 to obtain both orientation and causal evidence for some edges, i.e,. that they are non-confounded. The information from these two sources may be incomplete (some edges are not oriented or could appear due to possible confounding phenomena) and conflicting. This

[2]In our actual implementation the symmetrical test for B, $\exists \mathbf{Z}_3 \subseteq \mathcal{O}, I(B; D|\mathbf{Z}_3)$ and $C \in \mathbf{Z}_3$ is also checked, although theoretically not necessary.



Node	Num. Child	Max P-score
930	3	0.148
321	4	0.185
409	3	0.185
939	7	0.185
251	8	0.185
825	7	0.591
593	11	0.598
425	6	0.656
453	12	0.656
83	13	0.671
344	11	0.764

(h)

Figure V-4: Information available to determine $MB_{\mathbf{M}}(T)$ for REGED: (a) the DAG involving the MB variables determined by the search-and-score procedure (variables manipulated in REGED1 are shaded), (b)-(g) the top valid Y-tests ranked by p-score, and (h) a table of the variables from (a) considered to be either parents or children along with the number of valid Y-tests where the node appears as a child of T and the top p-score when this occurs.

information was combined manually and subjectively during the competition; however, for testing purposes during the post-challenge analysis and to be able to release a fully automated algorithm, we have replaced the manual step with an automated method. The latter attempts to follow as close as possible our thought process during the challenge.

We present the method following an example using the REGED1 data set. Figure V-4 illustrates and summarizes the different information sources. Figure V-4(a) shows the Markov Blanket variables extracted from the PDAG of Section V.2.4. The shaded nodes indicate the manipulated variables in REGED1. In addition, all possible Y-structures involving edges of the Markov Blanket were identified and scored. Figures V-4(b)-(g) show the top six Y-structures centered on the target node ranked by the maximum p-score. Finally, the table in V-4(h) lists for each variable the number of times it is determined to be a child of T and the maximum p-score among those instances. There were no Y-structures (A, B, X, T) that passed the Y-test with an edge $X \to T$ where $X \in MB_{\emptyset}(T)$; therefore, the Y-tests alone did not give any strong evidence for a variable to be a parent of the target.

We now describe how to identify the parents of T. We consider as possible parents all variables

returned by FSMB as neighbors of T. First, we identify the variables with strong evidence of being parents of T. These are the ones that appear as parents in the PDAG of the edge orientation phase of Section V.2.4. We sort them by the number of times they appear as non-confounded parents of T in Y-tests. In our example, these are variables with indexes $\{930, 321\}$ (Figure V-4(a)). Then, we filter out the variables with strong indication that they are indeed children of T; these are variables X for which the edge $T \rightarrow X$ gets a high p-score in some Y-test, i.e., they have maximum p-score above a threshold (arbitrarily set to 0.5). In our example, these are variables $\{825, 593, 425, 453, 83, 344\}$ (Figure V-4(h)). The remaining variables $\{409, 939, 251\}$ are those without strong evidence that they are either parents or children. These are sorted in decreasing order of the ratio of valid Y-tests as a parent to that as a child; ties are broken with preference to variables appearing less often as children of T in Y-tests. The final list to consider thus is $\{930, 321, 409, 939, 251\}$. During the competition, several subsets of this list were tried and a final decision was made among those submissions that ranked in the top 25% of all competitors. The automated procedure simply uses a threshold on the number of times the variables appear as children of T to remove the tail of the list.

If the complete $MB(T)$ is sought and not just the parents of T, we also need to identify the children and spouses of T. As children we consider the remaining non-manipulated variables adjacent to the target; in our example, these are variables with indexes $\{825, 425, 453, 344\}$. The spouses of the selected children are found from the PDAGs orientation: $\{454\}$ (alternatively, we could have used the same procedure for the identification of the parents of T as above, to identify the parents of the children of T).

In our effort to automate the above procedure after the challenge, we noticed that the procedure was not stable. Specifically, the lists of variables output and the corresponding models produced, varied significantly under different ordering of the variables in the data set. To alleviate the problem we augmented the procedure with a model-averaging-type step where we run the orientation procedure several times with different parameters (namely, we vary the equivalent sample size in the Bayesian Score and the kernel parameters for the scoring metric of Bach & Jordan 2002). Only the variables that appear consistently across parameter combinations remain in consideration. The

procedure has been validated in the post-challenge tests set by the organizers and was found stable and robust under permutations of the variables and subsampling of the data.

V.2.7 Building Predictive Models

Once the variable list was determined for each data set, a final classification SVM model was trained on only the variable list members (Boser *et al.*, 1992). An n-fold cross-validation design was used to select the optimal parameters: type of kernel (polynomial or Gaussian), kernel parameters (degree of kernel $\in \{1, 2, 3, 4\}$ or sigma $\in \{10^{-4}, 10^{-3}, \ldots, 10^{0}\}$), and C value $\in \{10^{-4}, 10^{-3}, \ldots, 10^{1}\}$. The value of n ranged from 5 to 10 based on the sample size available in the training sample. Once the best parameters were selected, a final SVM model was trained and used to predict the values for the test data sets.

V.3 Results

The classification performance (AUC reported as Tscore in the challenge results) is ultimately how the challenge submissions were rated. Table V-1 presents the Fnum, Fscore, Dscore, Tscore, and ranking of our final submission for each data set version. The number of entries before the final submission, the average Tscore (across the versions of a data set), and the overall ranking (generated from the average Tscore) are also shown in the table.

V.3.1 What Went Well

The specific implementation of our strategy performs well on the REGED data set achieving the top overall ranking. The strategy also exhibits decent performance on the unmanipulated data sets, version "0". This indicates that our implementation is approximating $MB_{\emptyset}(T)$ well. This is corroborated by the organizers' post-challenge analysis, shown in Figure V-5. In 3 of the 4 data sets (REGED, SIDO, and CINA) the method is performing well at identifying members of $MB_{\emptyset}(T)$. In fact, in those three data sets only $\sim$2 false positives are added to the Markov Blanket (the number of false negatives is undisclosed). Notice that our algorithms were able to accurately identify CINA's

Table V-1: Results on Challenge Data Sets: The Fnum, Fscore, Dscore, Tscore and Ranking is given for each version of the data sets. The results represent the final challenge submission. The number of entries, the overall ranking and average Tscore are given for each data problem. The cells are shaded in the colored quartile information: green - best 25%, yellow - best 50%, orange - worst 50%, and red worst 25%.

	Final Challenge Submission						
	Fnum	Fscore	Dscore	Tscore	Ranking		
CINA0	101	0.8496	0.9717	0.9721	9	Num. Entries / Total	7 / 277
CINA1	5	0.4716	0.9316	0.5113	23	Average Tscore	0.6015
CINA2	5	0.4716	0.9316	0.3210	25	Overall Ranking	23 / 25
MARTI0	24	0.5869	0.9952	0.9681	8	Num. Entries / Total	2 / 233
MARTI1	17	0.5643	0.9951	0.7837	9	Average Tscore	0.8083
MARTI2	3	0.4985	0.6973	0.6730	10	Overall Ranking	9 / 19
REGED0	15	0.8571	1.0000	0.9998	2	Num. Entries / Total	5 / 355
REGED1	9	0.7851	1.0000	0.9673	4	Average Tscore	0.9423
REGED2	3	1.0000	0.9728	0.8600	1	Overall Ranking	1 / 30
SIDO0	13	0.5115	0.9356	0.9230	12	Num. Entries / Total	2 / 242
SIDO1	4	0.5003	0.8587	0.6073	12	Average Tscore	0.6909
SIDO2	4	0.5003	0.8587	0.5426	14	Overall Ranking	12 / 28

$MB_\emptyset(T)$ numbering close to 100 variables. On MARTI it seems that $MB_\emptyset(T)$ was not accurately found, however we believe this is due to our inability to handle the noise correctly. Evidence to this is provided by the following experiment: the post-challenge analysis included other teams' preprocessed data for MARTI; re-running our method on the preprocessed data provided by Dr. Guyon we see a marked improvement in our performance (in particular on the MARTI0 data, where our method has proven to do well in all other cases). Specifically, the Tscore on MARTI0 improves from 0.9681 to a score of 0.9910 resulting in an improved ranking on that data set from eighth to fifth and corroborating that we approximate well the $MB_\emptyset(T)$ (the actual false positives and false negatives have not been released for post-challenge submissions).

V.3.2 What Went Wrong

While our methods performed well at identifying the unmanipulated Markov Blanket, the identification of the manipulated Markov Blanket was very poor on all but the REGED data set. This indicates that our methods for orienting the edges of $MB_\emptyset(T)$ performed poorly. We now provide some possible explanations.

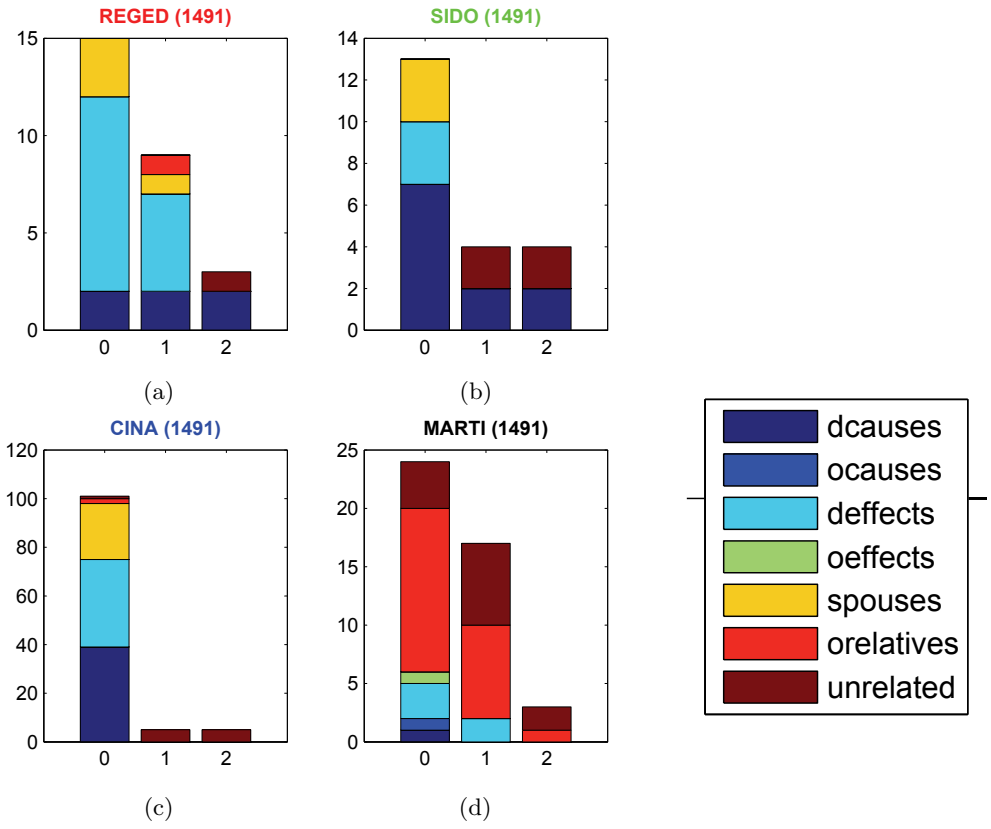


Figure V-5: The selected variables' relationship to the target variable, where dcauses = direct causes, deffects = direct effects, ocauses = other causes (indirect), oeffects = other effects (indirect), spouses = parent of direct effect, orelatives = other relatives, and unrelated = completely irrelevant.

Unfortunately, we spent most our time on the REGED data sets and the development of new methods, leaving little time for the rest of the data sets. Most importantly, we set out to solve a more difficult problem than what the organizers had set, namely inducing causality in the presence of hidden variables and violations of faithfulness. These are two important issues in real data sets, but did not occur in the challenge: FSMB identified between 0-4 features per data set that were added for consideration; these features were often considered spouses, or other relatives when selecting $MB_{\mathbf{M}}(T)$ and did not make much difference in performance. Also, there were actually no hidden variables in the challenge data sets. More specifically, all the variables participating in the models from which data were simulated, were also included in the released data sets. Because of the way data were simulated, the problematic confounding effect we described never occurred. We spent

Table V-2: Results on Challenge Data Sets: The Fnum, Fscore, Dscore, Tscore, and Ranking is given for each version of the data set. The results presented as if the $MB_{\emptyset}(T)$ is used for every variable list regardless of considering manipulations.The cells are shaded in the colored quartile information: green - best 25%, yellow - best 50%, orange - worst 50%, and red worst 25%.

	Unmanipulated MB used for all Data Sets				
	Fnum	Fscore	Dscore	Tscore	Ranking
CINA0	101	0.8496	0.9717	0.9721	9
CINA1	101	0.5795	0.9717	0.8581	4
CINA2	101	0.5795	0.9717	0.6917	8
MARTI0	24	0.5869	0.9948	0.9824	7
MARTI1	24	0.5985	0.9948	0.8477	9
MARTI2	24	0.7429	0.9948	0.6971	9
REGED0	15	0.8571	1.0000	0.9998	2
REGED1	15	0.7825	1.0000	0.9280	14
REGED2	15	1.0000	1.0000	0.7231	9
SIDO0	13	0.5015	0.9365	0.9237	12
SIDO1	13	0.5012	0.9365	0.6626	11
SIDO2	13	0.5012	0.9365	0.5713	11

a significant amount of time on this problem is because the FAQ of the competition specifically declared that there may be missing variables (a problem for many real-world analyses).

Also, our submissions were overly conservative in regards to including false positive variables, i.e, variables not in $MB_{\mathbf{M}}(T)$. However, it turns out that for this challenge, false negatives degrade performance significantly more than false positives (also see discussion in the organizers' post-challenge analysis online Appendix B, WCCI 2008 Causality Challenge 2008). This is exemplified by the following post-challenge experiment: we submitted a new set of entries where the variable list for each data set version was the $MB_{\emptyset}(T)$, a superset of $MB_{\mathbf{M}}(T)$. The results for these submissions are shown in Table V-2 and can be contrasted with the challenge results in V-1. On REGED, the performance is degraded since we were already ranking 1st on this task. On CINA, the challenge submission choice of $MB_{\mathbf{M}}(T)$ was both incorrect and very conservative, especially in light of the large size of the Markov Blanket and number of possible parents. The use of $MB_{\emptyset}(T)$ improved the performance and these results rank as high as fourth for CINA1. For MARTI and SIDO, the new submission returns a similar or slightly better ranking to that of the challenge submission. This analysis, while only over the limited data sets of this challenge, suggests that without an edge ori-

entation procedure to supply correct information to differentiate the parents and children, letting $MB_{\emptyset}(T)$ be the default manipulated Markov Blanket is a reasonable approach. In addition, we believe that a model averaging approach would also greatly improve the robustness of identifying the $MB_{\mathbf{M}}(T)$ and make it more resilient to edge-orientation errors.

Regarding the CINA data sets, we note that they consisted of a mixture of discrete and continuous variables. Many of the algorithms employed by our strategy heavily rely on tests of independence. Our implementations of these tests however, have been developed targeting only all discrete or all continuous variables and were not designed for mixed types of variables. Regarding the SIDO data sets, we were informed after the completion of the challenge that it contained variables created by the binarization of other variables. For example a variable V taking values $v_1, \ldots, v_k$ is converted to the binary variables $B_1, \ldots, B_k$ taking values $B_i = I(V = v_i)$, where I is the indicator function. The newly created variables B_i are all inter-dependent, since knowing $B_i = 1$ implies that $B_j = 0$, for $i \neq j$. Graphically, the new set of variables $\{B_i\}$ would consist of a *clique in the PDAG of a network*. If V is a parent of T in the original network, then all B_i's are connected to T and among each other. This reduces the identifiable Y-structures by our procedure and confuses all traditional search-and-score Bayesian Network learning algorithms. The problem stemming from binarization of variables points to an interesting future research direction.

Finally, due to the time pressure, several parts of our strategy were not fully optimized. We did not optimize the model construction procedure and just used standard SVMs with cross-validation. Most importantly, we did not have the time to fully test and optimize the novel algorithms and procedures for these tasks.

V.4 Lessons Learned and Conclusions

The most important outcome of participation in the challenge is the experience gained and realization of several theoretical and practical issues as well as ideas that emerged for future directions in the field.

Knowledge of the causal structure is theoretically necessary for making optimal predictions under

manipulations. This is exemplified in this challenge by the difference between the top non-causal submissions and the theoretical optimum performance; see the organizers' post-challenge analysis online (WCCI 2008 Causality Challenge, 2008, Figures 3-6). Regarding the state-of-the-art in causal discovery, there exists efficient, scalable, and publicly available code to learn the Markov Blanket. In fact, several other top participants also used the package Causal Explorer (Aliferis *et al.*, 2003b; Statnikov *et al.*, 2009) implementing such algorithms. These methods perform well on a range of high-dimensional data sets involving discrete, continuous, and mixed data. However, there is a shortage of reliable and efficient, publicly-available code or software packages that are meant to identify hidden variables or non-confounded variables. Of those available (e.g., Tetrad's FCI implementation), they are unable to scale to the size of the challenge problems (even when reduced to a region of depth 3). In addition, the state-of-the-art methods employed to learn the orientation did not perform well. Consequently, the manipulated Markov Blanket was unable to reliably identified.

Regarding important implementation issues, note that reducing the size of the problem to a region of depth 3 greatly improved the efficiency of the later applied methods; this reduction allowed the orientation procedures to complete in minutes rather than hours or days if the full variable set was considered. Several algorithms heavily depend on statistical tests that ought to be tailored for the problem at hand. Binarized variables pose a problem to causal-discovery methods at the moment.

In summary, a general strategy for predicting a quantity under manipulations of a system is presented. It relies on identifying $MB_{\mathbf{M}}(T)$ and fitting a model for $P_{\emptyset}(T|MB_{\mathbf{M}}(T))$ from the observational data. The steps of the strategy are shown in Figure V-2. They are implemented by existing algorithms and augmented with novel procedures for detecting certain kinds of violations of faithfulness and for detecting non-confounded causal edges. Overall, this challenge provided us with an opportunity to develop, apply, and compare methods for causal discovery on realistic, challenging problems and initiating new avenues of research.

CHAPTER VI

DISCUSSION AND CONCLUSIONS

This chapter presents a summary of the conclusions and future work; a more complete evaluation and description is presented in each of the Chapters II-V. The focus of this Ph.D. research is to develop and investigate several novel computational techniques for discovering informative patterns and complex relationship of biomedical data. The first method examines the composition of the decision function of Support Vector Machines (SVMs). Specifically, the method is designed to begin to understand what variables and/or combinations of variables are important to a classification task when using a polynomial Support Vector Machine (SVM) model. This new algorithm heuristically selects the most heavily weighted (and thus most important for classification) constructed features of a polynomial SVM. Sufficient conditions are provided for the heuristic algorithm to correctly return the top r weights. Even when the sufficient conditions fail, the research empirically shows that the returned weights closely approximate the true set of r top-weighted features when only examining a very small portion of the feature space. The method was able to successfully run on several simulated data sets where the true weight vector is known, comparing the norm of the r top-weighted features returned my the heuristic method to the known top r features. Additionally, the new method was applied to real world data sets. For these data sets, the true top weights are unknown however, the features returned were assessed by their classification performance (achieving similar performances as the full model and models using variable selection techniques). Also, for the case of the splice site data, the features selected seem consistent with biological knowledge in the domain.

This method is a first step in attempting to identify the top-weighted features of an SVM model; there are many future directions of this work, detailed in Chapter II and summarized here. First, the SVM models use the standard L2 norm where the L2 norm tends to spread the weighting across the features. In the future, it is proposed to investigate the use of L1 or L0 SVMs which may provide

more sparsity in the weight vector allowing for easier searching by the heuristic method. The search procedure itself can be extended and explored. Here, the search was guided by groupings of features involving a variable v at a level l; however, alternative collections of variables could be considered. In Chapter II, several general formulations are presented to consider sets of features and the norm of the weight vector over those features, which could be used to design new search methods. Another direction is to extend the process of selecting the top weighted features with Markov-blanket based variable selection algorithms, which was discussed in Chapter III.

Support Vector Machines (SVMs) models have been widely used to classify data. However, the reasoning behind the classification is complex, and previously unavailable to the user. The new, heuristic method is designed to explicitly determine the decision function used to classify data for polynomial SVMs. In particular, a heuristic method was designed to identify the highly weighted features of this decision function. These features may give insight into how the SVM classifies data and provide information on the features and variables relevant to the target class.

Next, in Chapter III a new feature selection method was developed. Markov Blanket-based and kernel-based methods illustrate two prominent paradigms in variable selection. The former follows a principled approach to variable selection and is able to guarantee some desirable theoretical properties such as optimality under certain broad conditions (e.g., data is i.i.d., Markov condition, faithfulness condition, etc.). Two examples of the conditions being violated are: (i) the optimal variable subset contains multivariate associations whose participant variables have no detectable univariate associations with T and (ii) the target variable is caused by variables in a redundant mechanism (see section III.3 for further details). The kernel-based approach is able to capture the multivariate and redundant relationships in such situations even in very high dimensional data sets. A new variable selection algorithm that combines the advantages of both approaches in a non-trivial way, called Feature Space Markov Blanket (FSMB), was presented.

The Feature Space Markov Blanket (FSMB) algorithm is the first attempt to construct an algorithm combining the theoretical properties of the two approaches. The main idea of FSMB is to identify the Markov Blanket of T in *feature space* instead of in the original variable space, where

multivariate associations become pairwise associations. FSMB employs a SVM to identify which features may have large pairwise association with T in feature space, so as to avoid considering all features.

The new method was compared with several prominent variable selection methods: HITON, Relief, and RFE on several simulated and real world data sets. The simulated data sets were designed to be problems where the Markov Blanket-based methods (HITON) would fail. FSMB was shown to perform as well or better than the other methods at returning the Markov Blanket with better sensitivity and specificity (note, RFE and Relief are not specifically designed to return the Markov Blanket). On the real world data, the true variables of interest are not known, therefore the methods are assessed by the number of variables/features returned and the classification performance (AUC). FSMB returns few features, which may be used to construct simple linear models that generally perform as well as the other techniques. For two cases, the Lung Cancer and Thrombin data sets, the method returns on 2 or 3 features. In this case, the model can be visualized to show the classification performance. On the HIVA data set, HITON does poorly (AUC = 0.527) while FSMB has better performance (AUC = 0.702). This difference in performance suggests that the HIVA domain might contain some multivariate relationships that HITON (and Markov Blanket-based methods) would fail to detect.

Future work on this method can be done in several areas. First, the method relies on the selection of several parameters. Better intuition into the optimal selection of parameters through domain knowledge or estimation procedures could be examined in future work. Additionally, the heuristic method offers a simple, efficient method to identify the top weights, but other search methods could be employed with tradeoffs in efficiency and/or quality of the results (this ties in with the future work discussed for Chapter II). Also, FSMB relies on the assumption that the features with the largest magnitude weights are most important (relevant) for the classification task; whether this assumption holds and why it may fail should be examined in future work. Other work may define and examine the prevalence of the "difficult" distributions where this new method is expected to out-perform other methods. Finally, an investigation on whether the ideas of this method may be

extended to other SVM formulations, namely regression and multi-categorical problems, should be explored.

Third, a new local causal discovery method was designed and studied, empirically and theoretically, for the focused learning of network regions from observational data. The new method as expected requires only a fraction of the time to learn a region compared to the global approach (where the full network is induced then pruned). Therefore, the new local method is able to scale up to even larger data sets than what is currently possible. The quality of learning by the local approach was of particular interest, that is whether learning a region in a myopic way, i.e., without simultaneously considering all parts of the network and how these interact, severely affects the quality of learning. The evaluation shows that in general the method for learning the region locally is more time-efficient and also produces structures of equal or higher-quality.

Future work in this area could focus on learning a different kind of region subgraph. The regions described here are one of an infinite kind of subgraphs that could be learned. In the new algorithm, variables are considered in a breadth-first, homogeneous expansion from the target, but the method is not dependent on this mode of exploration. Changing how the nodes are considered and expanded, can create many different algorithms that fit other desired subgraph arrangements. Additionally, an interesting new avenue of research is the possible use of techniques for combining local regions to form the complete global network.

Finally, using the above new methods and contemporary research, a principled submission (V) to the Causality Challenge tasks was developed (WCCI 2008 Causality Challenge, 2008). The overall strategy made use of the three other techniques described in this thesis as well as developing a theory to perform predictions under manipulation. The submission used the formalism of Causal Bayesian Networks to model and induce causal relations and to make predictions about the effects of the manipulation of the variables.

This approach on the challenge performed best on one of the four tasks. Across, the data sets, the method did well at identifying the unmanipulated Markov Blanket. However, when the directionality of the edges was needed to be determined to separate parents from children, the method did not do

as well. Also, submissions were overly conservative in regards to including false positive variables, i.e, variables not in $MB_{\mathbf{M}}(T)$. However, it turns out that for this challenge, false negatives degrade performance significantly more than false positives (also noted in the organizers' post-challenge analysis). The most important outcome of participation in the challenge is the experience gained and realization of several theoretical and practical issues as well as ideas that emerged for future directions in the field.

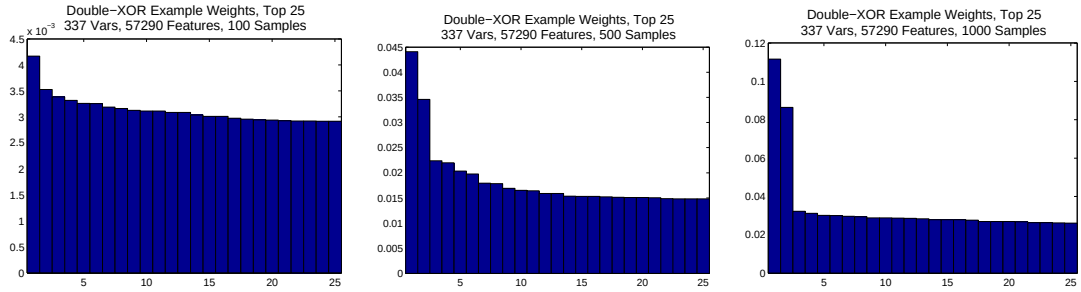
Finally, there exists efficient, scalable, and publicly available code to learn the Markov Blanket. However, there is a shortage of reliable and efficient, publicly-available code or software packages that are meant to identify hidden variables or non-confounded variables. In addition, it was observed that the state-of-the-art methods employed to learn the orientation did not perform well. Consequently, the manipulated Markov Blanket could not be reliably identified. Overall, this challenge provided an opportunity to develop, apply, and compare methods for causal discovery on realistic, challenging problems and initiating new avenues of research.

The thesis is constructed with chapters from the papers that have been submitted (or are in preparation) on this research. It contains several new algorithms that work to identify and discover informative patterns and complex relationship in biomedical data. The methods explored in the thesis are generally first steps in future research paths to explore understanding SVM models, variable selection in non-faithful problems, identifying causal relations in large domains, and learning with manipulations.

APPENDIX A

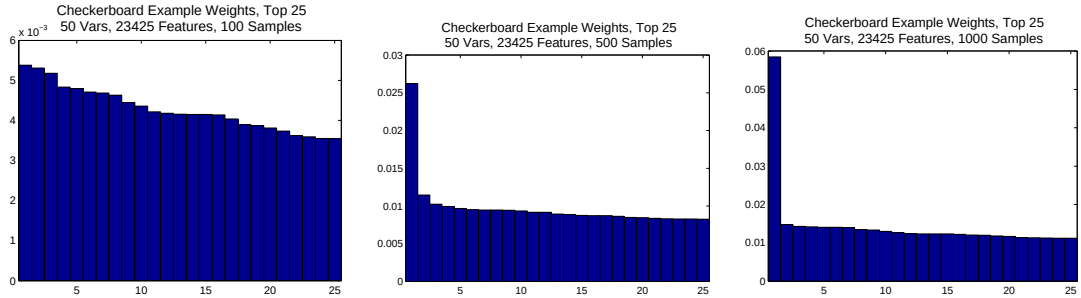
IDENTIFYING TOP-WEIGHTED FEATURES IN POLYNOMIAL SVM MODELS

A.I Weight Distributions and Contributions Matrix of Simulated Problems



(a) (b) (c)

Figure A-1: The distribution of weights of the top 25 features of the Double-XOR example with increasing sample across each row: (a) 100, (b) 500, and (c) 1000 samples.



(a) (b) (c)

Figure A-2: Distribution of Weights of Top 25 Features of the Checkerboard example with increasing sample across each row: (a) 100, (b) 500, and (c) 1000 samples.

A.II Brute Force vs. Heuristic Method

A.II.1 Timing Results

Each subplot of the figure graphs the brute force method's execution time along with the heuristic method's time for several values of t. Each plot shows the timing results for a specific size

problem. Figure A-3(a)-(c) are for the Circle problem with 250, 350, and 450 variables (31625, 61775, and 101925 features) respectively. Figure A-3(d)-(f) are for Double-XOR problems with 226, 337, and 448 variables (25877, 57290, and 101024 features) respectively. Figure A-3(g)-(i) are for the Checkerboard problems also with 50, 70, and 80 variables (23425, 62195, and 91880 features) respectively. Recall the brute force approach calculates all features and weights, while the heuristic method construct just p features and weights. The different colored bars within each plot illustrate the execution time of the brute force and heuristic method with $p = \{50, 100, 500, 1000, 5000\}$. The different groupings within each figure show the timing results for training data sets of increasing sample sizes (50, 100, 500, and 1000 samples).

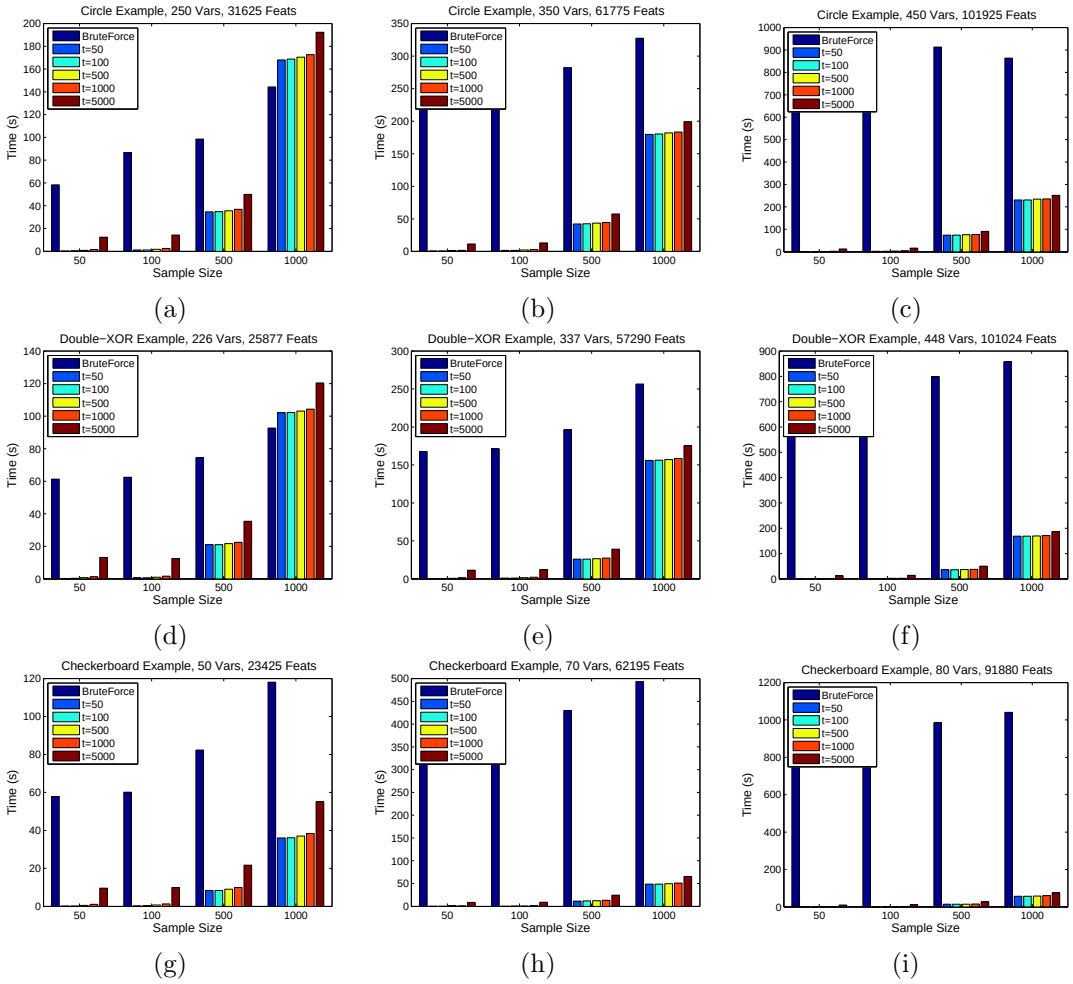


Figure A-3: Timing Results - Comparing Brute Force and Heuristic Methods: The timing results to compare the brute force and heuristic methods are presented over different problem sizes for the Circle (a)-(c), Double XOR (d)-(f), and Checkerboard (g)-(i) problems. The brute force method calculates all features and weights, and the heuristic method constructs p feature weights; the different colored bars within each plot represent the execution time of the brute force method and the heuristic method with $p = 50$, 100, 500, 1000, and 5000. The different groupings within each plot show execution time for training data sets of increasing sample size.

A.II.2 Quality Results

The quality results when comparing the top r weights are presented in Figure A-4. Each subplot of the figure graphs the quality metric over several parameters for a specific size problem. Figure A-4(a)-(c) are for the Circle problem with 250, 350, and 450 variables (31625, 61775, and 101925 features) respectively. Figure A-4(d)-(f) are for Double-XOR problems problems with 226, 337, and 448 variables (25877, 57290, and 101024 features) respectively. Figure A-4(g)-(i) are for the Checkerboard problems also with 50, 70, and 80 variables (23425, 62195, and 91880 features) respectively. For all figures the heuristic method was run with $p = 5000$ (construct 5000 features). The different colored bars within each plot illustrate the quality metric of the heuristic method with the number of features to return, $r = 50$, 100, 500, and 1000. The different groupings within each figure show the results for training data sets of increasing sample sizes (50, 100, 500, and 1000 samples).

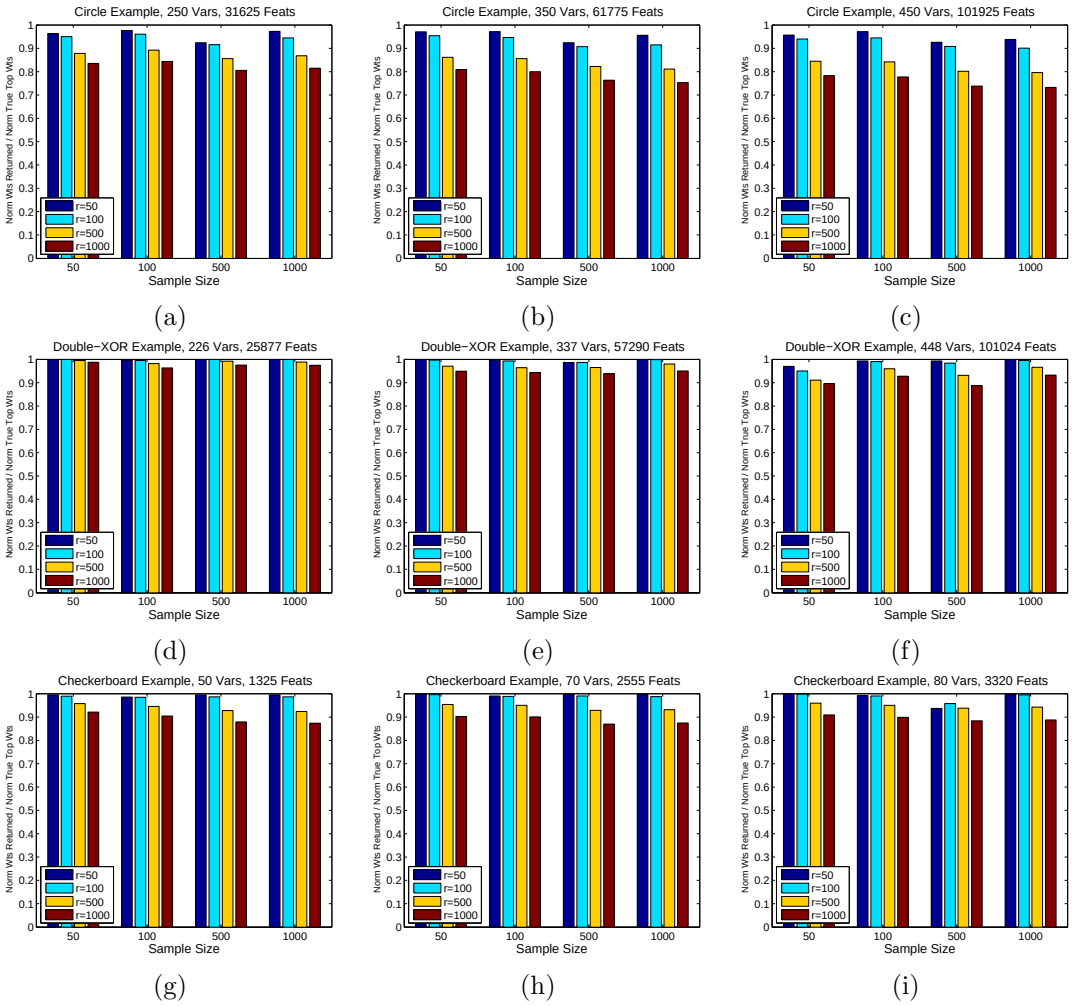


Figure A-4: Quality Results - Assessing the Heuristic Method on Top r Returned Features: The quality results assess whether the heuristic method is returning the top weighted features. The metric plotted is the L2 norm of the top r weights returned by the heuristic method divided by the L2 norm of the top r weights found by sorting the entire weight vector. The subplots presented the results over increasing problem sizes for the Circle (a)-(c), Double XOR (d)-(f), and Parity (g)-(i) problems. For each case, the heuristic method was run with $p = 5000$ (construct 5000 features). The different colored bars within each plot represent the quality metric for the heuristic method with $r = 50$, 100, 500, and 1000. The different groupings within each plot show the quality metric for training data sets of increasing sample size.

A.III RFE vs. Heuristic Method

A.III.1 Timing results

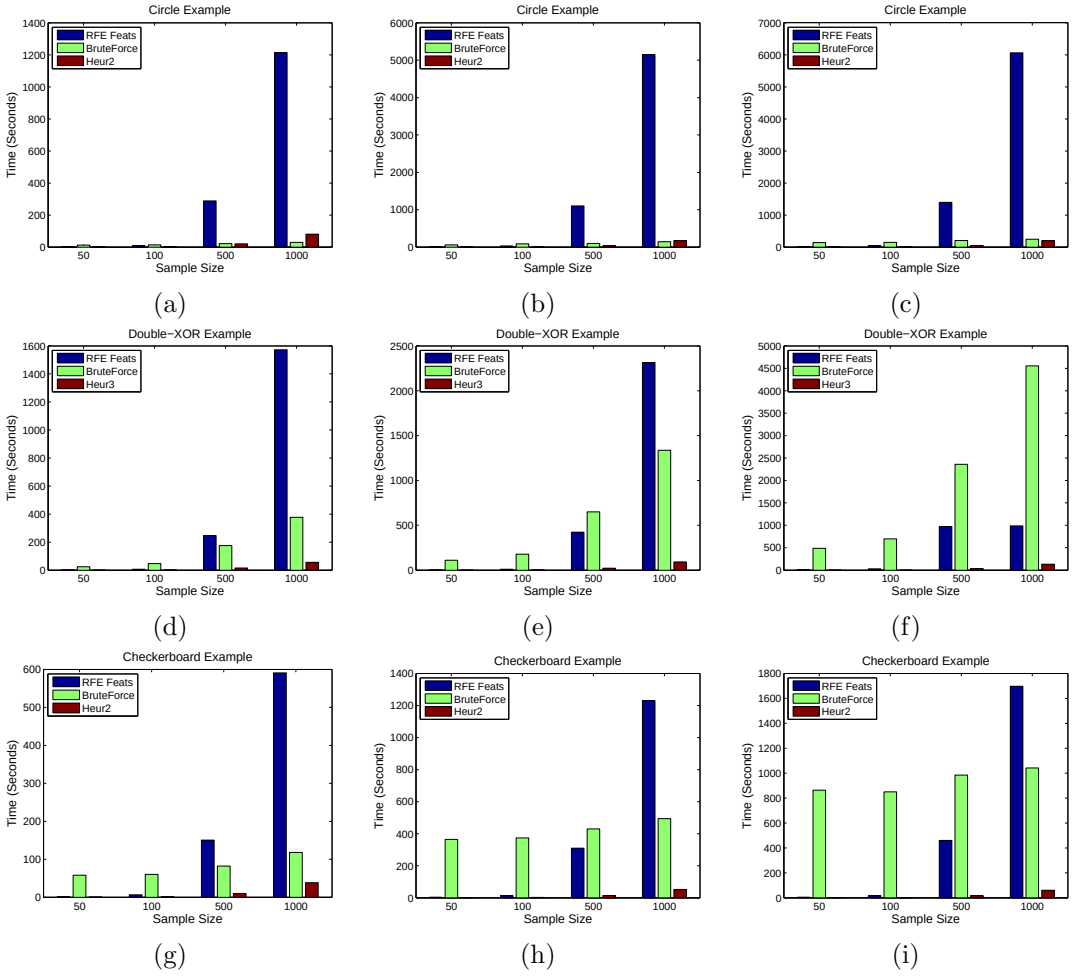


Figure A-5: Timing Results - RFE constructed Features: The time for running RFE to construct features is compared to the brute force and `Heur3` heuristic approaches. The subplots present the timing results over increasing problem sizes for the Circle (a-c), Double XOR (d-f), and Parity (g-i) problems. The different groupings within each plot show the times for training data sets of increasing sample size.

A.IV Lung Cancer Dataset

Table A-1: Lung Cancer Dataset: Variables selected by HITON(H), RFE(R), and those involved in the top features (F) are listed with their probe set ID and gene information.

Variable	Method	Symbol	probe set ID Gene Information
23	F	-	AFFX-hum_alu_at U14573 Human Alu-Sq subfamily consensus sequence.
51	F	-	AFFX-HSAC07 actin, beta
205	F	ANXA2P1	31444_s_at annexin A2 pseudogene 2
288	H	RPS2	31527_at ribosomal protein S2
604	F	RPL37A	31962_at ribosomal protein L37a
1060	F,R	VIM	34091_s_at vimentin
1376	F,R	RPS6	35125_at ribosomal protein S6
1668	R	COL1A2	Cluster Incl V00503:Human mRNA encoding Pro-alpha-2 chain of type I procollagen. (major part) /cds=(0,2223) 32307_s_at /gb=V00503 /gi=30123 /ug=Hs.179573 /len=2452
1679	R	ACTB	32318_s_at actin, beta
1906	F	IGHA2	Cluster Incl S71043:Ig alpha 2=immunoglobulin A heavy chain allotype 2 {constant region, germ line} [human, 33500_i_at peripheral blood neutrophils, Genomic, 1799 nt] /cds=(0,1022) /gb=S71043 /gi=546798 /ug=Hs.32225 /len=1047
1907	F	-	Cluster Incl S71043:Ig alpha 2=immunoglobulin A heavy chain allotype 2 {constant region, germ line} [human, 33501_r_at peripheral blood neutrophils, Genomic, 1799 nt] /cds=(0,1022) /gb=S71043 /gi=546798 /ug=Hs.32225 /len=1047
2093	H	*	34046_at hypothetical protein dJ37E16.5
2515	F	GAPDH	Cluster Incl U34995:Human normal keratinocyte substraction library mRNA, clone H22a, complete sequence 35905_s_at /cds=UNKNOWN /gb=U34995 /gi=1497857 /ug=Hs.195188 /len=1626
3119	H	UBE2D1	Cluster Incl AF020761:Homo sapiens stimulator of Fe transport mRNA, complete cds /cds=(85,1101) 37826_at /gb=AF020761 /gi=2738924 /ug=Hs.129683 /len=1404
3157	F	IGHV4-31	Cluster Incl Y14737:Homo sapiens mRNA for immunoglobulin lambda heavy chain /cds=(65,1498) /gb=Y14737 37864_s_at /gi=2765424 /ug=Hs.140 /len=1631
3255	H	FAT2	38202_at FAT tumor suppressor (Drosophila) homolog 2
3676	H	CSTA	Cluster Incl AA570193:nf38c11.s1 Homo sapiens cDNA /clone=IMAGE-916052 /gb=AA570193 /gi=2344173 39581_at /ug=Hs.2621 /len=450
4524	H	STARD13	Cluster Incl AL049801:Novel human gene mapping to chomosome 13, similar to rat RhoGAP /cds=(373,3360) 31790_at /gb=AL049801 /gi=4902677 /ug=Hs.13649 /len=5784
4586	H	ANKMY2	31852_at hypothetical protein DKFZp564O043
4786	R	DLK1	32648_at delta-like homolog (Drosophila)
4934	F	IGLV@	33273_f_at immunoglobulin lambda locus
4935	F	IGLV@	Cluster Incl M18645:Human Ig rearranged lambda-chain mRNA VJC-region subgroup lambda-IV from 33274_f_at heterohybridoma H6-3C4 /cds=(30,731) /gb=M18645 /gi=186103 /ug=Hs.181125 /len=872
4983	F	SFN	33322_i_at stratifin
6536	R	RAN	38708_at RAN, member RAS oncogene family
6686	H	H2AFZ	39337_at H2A histone family, member Z
6814	R	HMGA1	39704_s_at high-mobility group (nonhistone chromosomal) protein isoforms I and Y
6905	H	AP2M1	Cluster Incl D63475:Human mRNA for KIAA0109 gene, complete cds /cds=(86,1393) /gb=D63475 /gi=1665724 39795_at /ug=Hs.152936 /len=1868
6908	R	RPS28	Cluster Incl R87876:yo45h01.r1 Homo sapiens cDNA, 5 end /clone=IMAGE-180913 /clone_end=5" /gb=R87876 39798_at /gi=946689 /ug=Hs.153177 /len=483"
7366	R	RPS4Y1	41214_at ribosomal protein S4, Y-linked
7756	R	GSTP1	33396_at glutathione S-transferase pi
8201	R	RPS29	Cluster Incl AI541542:libtest16.A02.r Homo sapiens cDNA, 5 end /clone_end=5" /gb=AI541542 /gi=4458915 35278_at /ug=Hs.539 /len=639"
8429	R	CEACAM6	36105_at carcinoembryonic antigen-related cell adhesion molecule 6 (non-specific cross reacting antigen)
8727	R	CD63	37003_at CD63 antigen (melanoma 1 antigen)
8843	H	SPCS2	37359_at KIAA0102 gene product
8914	R	ATP1B1	37669_s_at ATPase, Na+/K+ transporting, beta 1 polypeptide
9071	H	HMGB2	38065_at high-mobility group (nonhistone chromosomal) protein 2
9977	F	EEF1A1	40887_g_at eukaryotic translation elongation factor 1 alpha 1-like 14
10139	H	NCAM1	Cluster Incl AA126505:zn86a09.s1 Homo sapiens cDNA, 3 end /clone=IMAGE-565048 /clone_end=3" 41289_at /gb=AA126505 /gi=1686153 /ug=Hs.237108 /len=713"
10525	H	SOX2	Cluster Incl L07335:Homo sapiens (clone 6AR33) HMG box mRNA, 3 end cds /cds=(0,983) /gb=L07335 33109_f_at /gi=184239 /ug=Hs.816 /len=1098"
10705	F	S100A2	2027_at M87068 /FEATURE= /DEFINITION=HUMCAN H.sapiens CaN19 mRNA sequence
10936	H	FNTA	1772_s_at farnesyltransferase, CAAX box, alpha
10997	R	IGFBP2	S37730 /FEATURE=cds /DEFINITION=S37712S4 insulin-like growth factor binding protein-2 [human, placenta, 1741_s_at Genomic, 1342 nt, segment 4 of 4]
11300	H	EIF4A2	1420_s_at eukaryotic translation initiation factor 4A, isoform 2
11355	R	TGFBI	1385_at transforming growth factor, beta-induced, 68kD
11359	H	PTPRZ1	1364_at protein tyrosine phosphatase, receptor-type, Z polypeptide 1
11436	F	EEF1A1	1288_s_at eukaryotic translation elongation factor 1 alpha 1
11743	R	USP9X	970_r_at ubiquitin specific protease 9, X chromosome (Drosophila fat facets related)
11942	F	ANXA2	769_s_at annexin A2
12097	F	KRT5	613_at keratin 5 (epidermolysis bullosa simplex, Dowling-Meara/Kobner/Weber-Cockayne types)
12150	R	MDK	577_at midkine (neurite growth-promoting factor 2)
12414	R	TACSTD2	291_s_at tumor-associated calcium signal transducer 2

APPENDIX B

MARKOV BLANKET-BASED VARIABLE SELECTION IN FEATURE SPACE

B.I Double XOR Problem Experimental Results

Table B-1: Results on Small Double-XOR Problem of 5 Variables. The sensitivity and specificity of identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3, x_4\}$) for each algorithm on the network of Fig. III-1(a). In this network, all parent-child relationships involving T are noisy-XOR. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Data		Sensitivity		
Size	HITON	Relief	RFE	FSMB
100	0.125 ± 0.24	0.875 ± 0.18	0.800 ± 0.23	0.900 ± 0.21
500	0.050 ± 0.11	1.000 ± 0.00	0.950 ± 0.16	1.000 ± 0.00
1000	0.050 ± 0.11	0.975 ± 0.08	0.975 ± 0.08	1.000 ± 0.00
Data		Specificity		
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
500	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
1000	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
Data		Distance		
Size	HITON	Relief	RFE	FSMB
100	0.875 ± 0.24	0.125 ± 0.18	0.200 ± 0.23	0.100 ± 0.21
500	0.950 ± 0.11	0.000 ± 0.00	0.050 ± 0.16	0.000 ± 0.00
1000	0.950 ± 0.11	0.025 ± 0.08	0.025 ± 0.08	0.000 ± 0.00

B.II Redundant Mechanism Experimental Results

Table B-2: Results on Redundant Mechanism Problem of 3 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2\}$) for each algorithm on the network of Fig. III-1(b). The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Data Size	Sensitivity HITON	Relief	RFE	FSMB
500	0.000 ± 0.00	0.700 ± 0.26	0.650 ± 0.24	0.700 ± 0.48
1000	0.200 ± 0.42	0.950 ± 0.16	0.950 ± 0.16	1.000 ± 0.00
1500	0.200 ± 0.42	0.750 ± 0.26	0.800 ± 0.26	1.000 ± 0.00
2000	0.400 ± 0.52	0.850 ± 0.24	0.850 ± 0.24	1.000 ± 0.00
Data Size	Specificity HITON	Relief	RFE	FSMB
500	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
1000	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
1500	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
2000	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
Data Size	Distance HITON	Relief	RFE	FSMB
500	1.000 ± 0.00	0.300 ± 0.26	0.350 ± 0.24	0.300 ± 0.48
1000	0.800 ± 0.42	0.050 ± 0.16	0.050 ± 0.16	0.000 ± 0.00
1500	0.800 ± 0.42	0.250 ± 0.26	0.200 ± 0.26	0.000 ± 0.00
2000	0.600 ± 0.52	0.150 ± 0.24	0.150 ± 0.24	0.000 ± 0.00

B.III Noisy 3-Parity Supplemental Experimental Results

The data for this classification problem is determined by a noisy 3-input parity function. The same experimental design is used as in section III.7.1 except the noise injected in the parity function is 20% (in the chapter 30% noise was used).

Table B-3: The Results on Noisy 3-Parity Problem of 60 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 60 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 60

Data Size	HITON	Relief	RFE	FSMB
	Sensitivity			
100	0.000 ± 0.00	0.500 ± 0.45	0.367 ± 0.37	0.667 ± 0.44
500	0.033 ± 0.11	0.867 ± 0.32	0.933 ± 0.21	1.000 ± 0.00
1000	0.033 ± 0.11	0.900 ± 0.32	1.000 ± 0.00	1.000 ± 0.00
	Specificity			
100	0.979 ± 0.01	0.572 ± 0.43	0.628 ± 0.36	0.768 ± 0.03
500	0.974 ± 0.01	0.939 ± 0.15	0.968 ± 0.04	0.840 ± 0.04
1000	0.968 ± 0.01	0.946 ± 0.03	0.944 ± 0.03	0.837 ± 0.02
	Distance			
100	1.000 ± 0.00	0.854 ± 0.25	0.869 ± 0.15	0.482 ± 0.35
500	0.967 ± 0.11	0.167 ± 0.35	0.089 ± 0.21	0.160 ± 0.04
1000	0.967 ± 0.11	0.149 ± 0.30	0.056 ± 0.03	0.163 ± 0.02

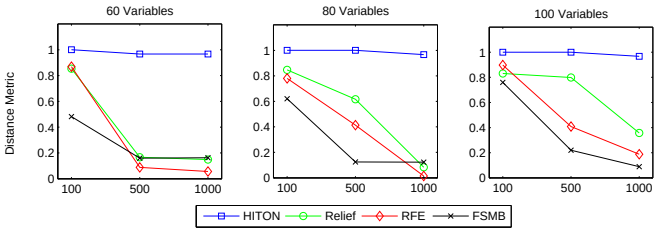


Figure B-1: Noisy 3-Parity Results Summary. This figure is plotting the distance metrics for each algorithm versus increasing sample size. The three plots are for the three problems sizes increasing from left to right: 60, 80, and 100 variables. The colored lines indicate the method: blue - HITON, green - Relief, red - RFE, and black - FSMB.

Table B-4: The Results on Noisy 3-Parity Problem of 80 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 60 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 80				
Data	Sensitivity			
Size	HITON	Relief	RFE	FSMB
100	0.000 ± 0.00	0.633 ± 0.43	0.400 ± 0.38	0.433 ± 0.32
500	0.000 ± 0.00	0.500 ± 0.53	0.600 ± 0.52	1.000 ± 0.00
1000	0.033 ± 0.11	0.967 ± 0.11	1.000 ± 0.00	1.000 ± 0.00
Data	Specificity			
Size	HITON	Relief	RFE	FSMB
100	0.981 ± 0.01	0.447 ± 0.41	0.736 ± 0.31	0.809 ± 0.04
500	0.981 ± 0.01	0.818 ± 0.30	0.932 ± 0.10	0.875 ± 0.04
1000	0.975 ± 0.01	0.927 ± 0.15	0.986 ± 0.01	0.877 ± 0.03
Data	Distance			
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	0.847 ± 0.21	0.780 ± 0.21	0.620 ± 0.27
500	1.000 ± 0.00	0.616 ± 0.51	0.413 ± 0.52	0.125 ± 0.04
1000	0.967 ± 0.11	0.083 ± 0.18	0.014 ± 0.01	0.123 ± 0.03

Table B-5: The Results on Noisy 3-Parity Problem of 100 Variables. The sensitivity, specificity, and distance measure for identifying the true Markov Blanket ($MB(T) = \{x_1, x_2, x_3\}$) of the noisy 3-parity problem with 60 variables. The results are presented as mean values and their standard deviation over 10 different samplings from the distribution.

Number of Variables = 100				
Data	Sensitivity			
Size	HITON	Relief	RFE	FSMB
100	0.000 ± 0.00	0.733 ± 0.41	0.533 ± 0.45	0.267 ± 0.34
500	0.000 ± 0.00	0.533 ± 0.45	0.600 ± 0.52	0.867 ± 0.28
1000	0.033 ± 0.11	0.767 ± 0.39	0.833 ± 0.36	1.000 ± 0.00
Data	Specificity			
Size	HITON	Relief	RFE	FSMB
100	0.978 ± 0.01	0.392 ± 0.43	0.501 ± 0.45	0.841 ± 0.02
500	0.984 ± 0.01	0.596 ± 0.43	0.969 ± 0.04	0.888 ± 0.03
1000	0.978 ± 0.01	0.842 ± 0.30	0.956 ± 0.07	0.911 ± 0.02
Data	Distance			
Size	HITON	Relief	RFE	FSMB
100	1.000 ± 0.00	0.831 ± 0.27	0.898 ± 0.17	0.760 ± 0.32
500	1.000 ± 0.00	0.799 ± 0.33	0.408 ± 0.51	0.220 ± 0.24
1000	0.967 ± 0.11	0.357 ± 0.43	0.188 ± 0.36	0.089 ± 0.02

B.IV Number of Variables Selected in Real World Data Sets

The number of variables selected by each method is listed in the table below. The methods and data sets are a subset of those reported in Aliferis *et al.* (2009a) and Aliferis *et al.* (2009b) along with the new analysis of FSMB.

Table B-6: Number of Variables Selected in Real World Data Sets.

	Data Set					
Variable Selection Method	Infant Mortality	Sylva	Hiva	Gisette	Ohsumed	Thrombin
No Variable Selectior	86	216	1,617	5,000	14,373	139,351
RFE, 50%, best subset selected	5	27	51	625	1,797	34,838
RFE, 20%, best subset selected	9	36	111	344	1,929	9,576
UAF - KruskalWallis - SVM, 50%	3	54	51	1,250	7,187	69,675
UAF - KruskalWallis - SVM, 20%	23	57	111	1,638	9,199	89,185
UAF - Signal2Noise - SVM, 50%	21	27	808	1,250	7,187	34,838
UAF - Signal2Noise - SVM, 20%	2	29	1,294	2,048	9,199	45,663
Random Forest Var. Selection	86	36	217	–	–	–
LARS - Elastic Net	9	181	168	176	155	168
RELIEF, neighbors = 1, 50%	43	54	808	1,250	14,373	17,419
RELIEF, neighbors = 1, 20%	12	71	271	2,048	9,199	14,963
RELIEF, neighbors = 5, 50%	11	27	808	1,250	14,373	17,419
RELIEF, neighbors = 5, 20%	28	36	71	2,048	9,199	14,963
L0-norm	47	111	191	158	215	63
Koller-Sahami, k=0	22	64	–	–	–	–
Koller-Sahami, k=1	7	79	–	–	–	–
Koller-Sahami, k=2	12	89	–	–	–	–
IAMB, G^2 test, alpha = 0.05	3	9	7	8	7	6
K2MB	2	9	5	6	83	–
BLCD-MB	2	9	5	6	83	–
HITON-PC, G^2 test, maxk = 3, alpha = 0.05	5	29	7	53	35	13
HITON-PC, G^2 test, maxk = 2, alpha = 0.05	5	42	10	151	44	26
MMPC, G^2 test, maxk=3, alpha = 0.05	5	29	8	10	33	1
MMPC, G^2 test, maxk=2, alpha = 0.05	5	42	12	14	43	1
HITON-MB, maxk = 3, alpha = 0.05	7	50	8	226	91	36
MMMB, maxk = 3, alpha = 0.05	8	50	10	79	80	2
FSMB	13	29	14	48	40	5

B.V Percentage of Variables Selected in Real World Data Sets

The percentage of variables selected by each method is listed in the table below. The methods and data sets are a subset of those reported in Aliferis *et al.* (2009a) and Aliferis *et al.* (2009b) along with the new analysis of FSMB.

Table B-7: Percentage of Variables Selected in Real World Data Sets

	Data Set					
Variable Selection Method	Infant Mortality	Ohsumed	Gisette	Sylva	Hiva	Thrombin
No Variable Selection	100.00%	100.00%	100.00%	100.00%	100.00%	100.000%
RFE, 50%, best subset selected	5.81%	12.50%	12.50%	12.50%	3.15%	25.000%
RFE, 20%, best subset selected	10.47%	13.42%	6.88%	16.67%	6.86%	6.872%
UAF - KruskalWallis - SVM, 50%	3.49%	50.00%	25.00%	25.00%	3.15%	50.000%
UAF - KruskalWallis - SVM, 20%	26.74%	64.00%	32.76%	26.39%	6.86%	64.000%
UAF - Signal2Noise - SVM, 50%	24.42%	50.00%	25.00%	12.50%	49.97%	25.000%
UAF - Signal2Noise - SVM, 20%	2.33%	64.00%	40.96%	13.43%	80.02%	32.768%
Random Forest Var. Selection	100.00%	–	–	16.67%	13.42%	–
LARS - Elastic Net	10.47%	1.08%	3.52%	83.80%	10.39%	0.121%
RELIEF, neighbors = 1, 50%	50.00%	100.00%	25.00%	25.00%	49.97%	12.500%
RELIEF, neighbors = 1, 20%	13.95%	64.00%	40.96%	32.87%	16.76%	10.738%
RELIEF, neighbors = 5, 50%	12.79%	100.00%	25.00%	12.50%	49.97%	12.500%
RELIEF, neighbors = 5, 20%	32.56%	64.00%	40.96%	16.67%	4.39%	10.738%
L0-norm	54.65%	1.50%	3.16%	51.39%	11.81%	0.045%
Koller-Sahami, k=0	25.58%	–	–	29.63%	–	–
Koller-Sahami, k=1	8.14%	–	–	36.57%	–	–
Koller-Sahami, k=2	13.95%	–	–	41.20%	–	–
IAMB, G^2 test, alpha = 0.05	3.49%	0.05%	0.16%	4.17%	0.43%	0.004%
K2MB	2.33%	0.58%	0.12%	4.17%	0.31%	–
BLCD-MB	2.33%	0.58%	0.12%	4.17%	0.31%	–
HITON-PC, G^2 test, maxk = 3, alpha = 0.05	5.81%	0.24%	1.06%	13.43%	0.43%	0.009%
HITON-PC, G^2 test, maxk = 2, alpha = 0.05	5.81%	0.31%	3.02%	19.44%	0.62%	0.019%
MMPC, G^2 test, maxk=3, alpha = 0.05	5.81%	0.23%	0.20%	13.43%	0.49%	0.001%
MMPC, G^2 test, maxk=2, alpha = 0.05	5.81%	0.30%	0.28%	19.44%	0.74%	0.001%
HITON-MB, maxk = 3, alpha = 0.05	8.14%	0.63%	4.52%	23.15%	0.49%	0.026%
MMMB, maxk = 3, alpha = 0.05	9.30%	0.56%	1.58%	23.15%	0.62%	0.001%
FSMB	15.12%	0.28%	0.00%	13.43%	0.87%	0.004%

B.VI Classification Performance (AUC) in Real World Data Sets

The classification performance of each method is listed in the table below. The methods and data sets are a subset of those reported in Aliferis *et al.* (2009a) and Aliferis *et al.* (2009b) along with the new analysis of FSMB.

Table B-8: Classification Performance (AUC) in Real World Data Sets

	Data Set					
Variable Selection Method	Infant Mortality	Sylva	Hiva	Gisette	Ohsumed	Thrombin
No Variable Selectior	0.820	0.998	0.717	0.997	0.857	0.925
RFE, 50%, best subset selected	0.748	0.998	0.640	0.998	0.852	0.917
RFE, 20%, best subset selected	0.747	0.998	0.747	0.997	0.859	0.908
UAF - KruskalWallis - SVM, 50%	0.839	0.999	0.668	0.999	0.879	0.940
UAF - KruskalWallis - SVM, 20%	0.874	0.999	0.714	0.999	0.872	0.930
UAF - Signal2Noise - SVM, 50%	0.855	0.999	0.693	0.999	0.864	0.932
UAF - Signal2Noise - SVM, 20%	0.837	0.999	0.723	0.999	0.865	0.939
Random Forest Var. Selection	0.820	0.999	0.696	–	–	–
LARS - Elastic Net	0.882	0.999	0.729	0.995	0.800	0.887
RELIEF, neighbors = 1, 50%	0.824	0.999	0.706	0.999	0.857	0.921
RELIEF, neighbors = 1, 20%	0.771	0.999	0.639	0.999	0.866	0.924
RELIEF, neighbors = 5, 50%	0.771	0.999	0.744	0.999	0.857	0.894
RELIEF, neighbors = 5, 20%	0.841	0.999	0.606	0.998	0.859	0.893
L0-norm	0.817	0.998	0.682	0.994	0.718	0.814
Koller-Sahami, k=0	0.845	0.999	–	–	–	–
Koller-Sahami, k=1	0.858	0.999	–	–	–	–
Koller-Sahami, k=2	0.800	0.998	–	–	–	–
IAMB, G^2 test, alpha = 0.05	0.811	0.992	0.488	0.972	0.665	0.769
K2MB	0.780	0.992	0.662	0.947	0.718	–
BLCD-MB	0.780	0.992	0.662	0.947	0.718	–
HITON-PC, G^2 test, maxk = 3, alpha = 0.05	0.860	0.997	0.706	0.990	0.773	0.825
HITON-PC, G^2 test, maxk = 2, alpha = 0.05	0.860	0.997	0.708	0.994	0.826	0.863
MMPC, G^2 test, maxk=3, alpha = 0.05	0.860	0.997	0.699	0.980	0.773	0.753
MMPC, G^2 test, maxk=2, alpha = 0.05	0.860	0.997	0.701	0.980	0.822	0.753
HITON-MB, maxk = 3, alpha = 0.05	0.865	0.997	0.527	0.997	0.778	0.798
MMMB, maxk = 3, alpha = 0.05	0.863	0.997	0.674	0.990	0.741	0.753
FSMB	0.803	0.997	0.702	0.993	0.811	0.939

APPENDIX C

LEARNING BAYESIAN NETWORK REGIONS

C.I Learning Local Regions: *RegionMMHC* vs. *AlgorithmGPC*

First, a comparison of the execution time of the two methods is presented. Note, the comparison of computation time between the two methods comes with several caveats: the two methods were run on different machines with different hardware, with different operating systems, and different programs (Matlab for *RegionMMHC* and compiled C++ for *AlgorithmGPC*).

Table C-1: Execution Time Results for the *RegionMMHC* and *AlgorithmGPC* approaches. The execution time is averaged over the 5 data samples and the 10 random target nodes for each network and sample size. The relative execution time (*AlgorithmGPC*/ *RegionMMHC*) is then reported for each network, sample size, and depth of region.

		Relative Time (AlgorithmsGPC / RegionMMHC)				
Data	SS	d=1	d=2	d=3	d=4	d=5
Alarm	500	17.4%	7.9%	7.9%	7.5%	8.2%
Alarm	1000	7.9%	8.4%	9.5%	8.8%	9.3%
Alarm	5000	14.9%	18.3%	20.4%	20.4%	22.2%
Ins10	500	5.5%	6.5%	7.4%	8.0%	8.2%
Ins10	1000	6.8%	8.6%	10.1%	10.8%	11.0%
Ins10	5000	17.1%	26.1%	29.9%	30.1%	32.4%
Alarm10	500	5.0%	5.0%	4.8%	4.9%	4.9%
Alarm10	1000	6.7%	7.1%	7.4%	7.8%	8.0%
Alarm10	5000	19.8%	22.4%	24.0%	25.1%	26.2%
Hail5	500	NA	NA	NA	NA	NA
Hail5	1000	NA	NA	NA	NA	NA
Hail5	5000	NA	NA	NA	NA	NA
Pigs	500	3.9%	7.4%	12.1%	13.3%	19.4%
Pigs	1000	37.5%	94.6%	238.2%	227.9%	354.8%
Pigs	5000	NA	NA	NA	NA	NA
RN50	500	7.8%	7.9%	7.9%	9.5%	12.6%
RN50	1000	8.0%	6.7%	5.9%	5.6%	5.2%
RN50	5000	22.4%	11.8%	9.2%	9.6%	9.6%
RN100	500	9.3%	9.8%	11.1%	14.3%	17.0%
RN100	1000	16.1%	20.1%	24.8%	32.6%	38.7%
RN100	5000	383.6%	243.3%	298.6%	405.7%	447.1%
RN500	500	NA	NA	NA	NA	NA
RN500	1000	NA	NA	NA	NA	NA
RN500	5000	NA	NA	NA	NA	NA

The methods are also compared in terms of structural quality.

Table C-2: Structural Quality Results for the *RegionMMHC* and *AlgorithmGPC* approaches. The *RegionMMHC* and *AlgorithmGPC* structural quality is measured by Structural Hamming Distance (SHD). The global and local *SHD* is averaged over the 5 data samples and 10 random target nodes. The relative quality of the two approaches is given by the difference in *SHD* (*RegionMMHC- AlgorithmGPC*) and is reported for each network, sample size and depth of region. A negative relative *SHD* indicates fewer errors by the *RegionMMHC*; whereas, a positive relative *SHD* indicates fewer errors by *AlgorithmGPC*.

		Relative Quality (RegionMMHC - AlgorithmsGPC)				
Data	SS	d=1	d=2	d=3	d=4	d=5
Alarm	500	−2.34	−3.04	−2.48	−1.34	−3.80
Alarm	1000	−1.42	−2.38	−4.28	−4.84	−4.26
Alarm	5000	−1.60	−3.70	−3.76	−4.12	−4.94
Ins10	500	−0.88	−1.36	−1.84	−3.48	−8.62
Ins10	1000	−1.36	−1.02	−0.40	−1.72	−3.22
Ins10	5000	−1.38	−0.70	0.34	2.34	10.16
Alarm10	500	−1.26	−2.18	−3.60	−6.28	−8.14
Alarm10	1000	−1.78	−3.20	−5.14	−7.64	−9.04
Alarm10	5000	−2.04	−3.00	−5.10	−5.52	−3.38
Hail5	500	NA	NA	NA	NA	NA
Hail5	1000	NA	NA	NA	NA	NA
Hail5	5000	NA	NA	NA	NA	NA
Pigs	500	−1.84	−8.88	−29.56	−58.86	−116.76
Pigs	1000	−1.54	−7.40	−24.60	−47.48	−87.94
Pigs	5000	NA	NA	NA	NA	NA
RN50	500	−12.08	−104.40	−120.72	14.14	79.52
RN50	1000	−12.56	−105.02	−124.52	0.16	45.54
RN50	5000	−11.60	−93.60	−117.54	−43.94	−41.52
RN100	500	1.38	−21.94	−211.62	−277.14	−270.02
RN100	1000	1.28	−21.34	−207.18	−262.90	−240.14
RN100	5000	−1.26	−29.04	−192.02	−199.82	−160.92
RN500	500	NA	NA	NA	NA	NA
RN500	1000	NA	NA	NA	NA	NA
RN500	5000	NA	NA	NA	NA	NA

APPENDIX D

A STRATEGY FOR MAKING PREDICTIONS UNDER MANIPULATION

D.I Proof

Theorem V.1 *Let $\langle G_\emptyset, P_\emptyset \rangle$ be a CBN and $\langle G_{\mathbf{M}}, P_{\mathbf{M}} \rangle$ be the resulting CBN under manipulations of variables in $\mathbf{M}$. Suppose that $T \notin \mathbf{M}$ and also that there is no manipulated child C of T in $G_\emptyset$ with a descendant D in $G_\emptyset$ that is also in $MB_{\mathbf{M}}(T)$. Then,*

$$P_{\mathbf{M}}(T|MB_{\mathbf{M}}(T)) = P_\emptyset(T|MB_{\mathbf{M}}(T)).$$

Proof. We base the proof of the theorem on the more general theory of probability invariance under manipulations found in Spirtes *et al.* (2000). Let G be the original graph $G_\emptyset$ with the additional exogenous variable E representing the manipulating agent and edges from E to any manipulated variable in $\mathbf{M}$. All graph operations that follow in the proof are on G (in the terminology of Spirtes *et al.* (2000) G is the combined graph G_{comb}). Then $P_\emptyset(\mathbf{Y}|\mathbf{Z}) = P_{\mathbf{M}}(\mathbf{Y}|\mathbf{Z})$, if $Dsep(E, \mathbf{Y}|\mathbf{Z})$, where $\mathbf{Y}$, $\mathbf{Z}$ are two disjoint sets and $Dsep(E, \mathbf{Y}|\mathbf{Z})$ denotes the d-separation of E from $\mathbf{Y}$ given $\mathbf{Z}$ in G. Thus, we just need to show that $Dsep(T; E|MB_{\mathbf{M}}(T))$ under the conditions $\mathcal{C}$:

There is no pair of variables C, D such that:

1. $E \rightarrow C \leftarrow T$
2. $C \rightsquigarrow D$
3. $D \in MB_{\mathbf{M}}(T)$

where $C \rightsquigarrow D$ denotes a directed path from C to D. Let us assume that the d-separation does not hold when conditions $\mathcal{C}$ do, and reach a contradiction. Recall that there are no incoming edges to E since it is an exogenous variable and no edge from E to T.

Since the d-separation does not hold, there must be an open path from E to T that is not blocked by $MB_{\mathbf{M}}(T)$. Take a path of the form $E \to \cdots P \to T$. $P \in MB_{\mathbf{M}}(T)$ under any manipulation and so we condition on it and it blocks the path. Thus, since there is an open path, it must be of the form $E \to \cdots C \leftarrow T$. For the path to be open, for each collider on it, we must be conditioning on either the collider or a descendant of the collider. Let us now consider the last collider on the path, which can be (1) C itself, or (2) some other node G.

Case (1): The open path is of the form $E \to \cdots C \leftarrow T$ and C is the last collider on it. We also distinguish two subcases, either (1a) the path is of the form $E \to C \leftarrow T$, or (1b) of the form $E \to \cdots S \to C \leftarrow T$. If (1a) is true, since C is a collider on the open path of case (1) we must be conditioning on either itself or a descendant of it $D \in MB_{\mathbf{M}}(T)$. Since, in (1a) C is manipulated, $C \notin MB_{\mathbf{M}}(T)$ and we cannot be conditioning on C itself. Thus, there is a $D \in MB_{\mathbf{M}}(T)$, descendant of C and conditions $\mathcal{C}$ all hold reaching a contradiction.

If (1b)is true, then S cannot belong in $MB_{\mathbf{M}}(T)$ or it would block the path by conditioning on it. Thus, $S \notin MB_{\mathbf{M}}(T)$ and the only way for this to be possible is if C is manipulated and so $E \to C \leftarrow T$ holds. Similarly to case (1a) we then conclude that conditions $\mathcal{C}$ should hold, reaching a contradiction.

Case (2): The open path is of the form $E \to \cdots G \leftarrow \cdots \leftarrow C \leftarrow T$ and G is the last collider on the path. If $C \in MB_{\mathbf{M}}(T)$ then we condition on it and it blocks the path. Thus, $C \notin MB_{\mathbf{M}}(T)$ which means C is manipulated and so $E \to C \leftarrow T$ holds. For the path to be open, given that G is a collider we must be conditioning on a node $D \in MB_{\mathbf{M}}(T)$ that is either G itself or a descendant of it. In either case, D must be a descendant of C too since there is a directed path $G \leftarrow \cdots \leftarrow C$ (notice this path cannot be of the form $G \leftarrow Q \to C$ or C and not G would be the last collider on the path $E \to \cdots G \leftarrow \cdots \leftarrow C \leftarrow T$). Thus, case (2) implies conditions $\mathcal{C}$ hold, again contrary to what we assumed. □

BIBLIOGRAPHY

Abramson, B., Brown, J., Edwards, W., Murphy, A., & Winkler, R.L. 1996. Hailfinder: A Bayesian System for forecasting severe weather. *International Journal of Forecasting*, **12**(1), 57–71.

Aliferis, C. F., Tsamardinos, I., & Statnikov, A. 2003a. HITON, A Novel Markov Blanket Algorithm for Optimal Variable Selection. *Pages 21–25 of: American Medical Informatics Association (AMIA).*

Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X.D. 2009a. Local Causal and Markov Blanket Induction Algorithms for Causal Discovery and Feature Selection for Classification. Part I: Algorithms and Empirical Evaluation. *to appear in Journal of Machine Learning Research.*

Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X.D. 2009b. Local Causal and Markov Blanket Induction Algorithms for Causal Discovery and Feature Selection for Classification. Part II: Analysis and Extensions. *to appear in Journal of Machine Learning Research.*

Aliferis, Constantin F., Tsamardinos, Ioannis, Statnikov, Alexander, & Brown, Laura E. 2003b. Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery. *Pages 371–376 of: International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS '03).*

Allmuallim, H, & Dietterich, TG. 1994. Learning Boolean concepts in the presences of many irrelevant features. *Artificial Intelligence*, **69**(1-2), 279–305.

Andreassen, S., Jensen, F.V., Andersen, S.K., Falck, B., Kjarulff, U., Woldbye, M., & et. al. 1989. Munin - An Expert EMG Assistant. *In:* Desmedt, JE (ed), *Computer-Aided Electromyography and Expert Systems.*

Andreassen, S, Riekehr, C, Kristensen, B, Schonheyder, HC, & Leibovici, L. 1999. Using probabilistic and decision-theoretic methods in treatment and prognosis modeling. *Artificial Intelligence in Medicine*, **15**(2), 121–134.

Bach, F.R., & Jordan, M.I. 2002. Learning Graphical Models with Mercer Kernels. *Pages 1009–1016 of: Proceedings of the Conference on Neural Information Processing Systems (NIPS-02).*

Bai, X., Padman, R., Ramsey, J., & Spirtes, P. 2008. Tabu Search-Enhanced Graphical Models for Classification in High Dimensions. *INFORMS JOURNAL ON COMPUTING*, **20**(3), 423–437.

Bay, SD, Shrager, J, Pohorille, A, & Langley, P. 2002. Revising regulatory networks: from expression data to linear causal models. *Journal of Biomedical Informatics*, **35**(5-6), 289–297.

Beinlich, I., Suermondt, G., Chavez, R., & Cooper, G. 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Pages 247–256 of: 2nd European Conference in Artificial Intelligence in Medicine.*

Bhattacharjee, A., Richards, W.G., Staunton, J., Li, C., Monti, S., Vasa, P., & et al. 2001. Classification of human lung carcinomas by mRNA expression profliing reveals distinct adenocarcinoma

subclasses. *Proceedings of the National Academy of Sciences*, **98**(24), 13790–5.

Bi, J., Bennett, K., Embrechts, M., Breneman, C., & Song, M. 2003. Dimensionality Reduction via Sparse Support Vector Machines. *Journal of Machine Learning Research*, **3**, 1229–1243.

Binder, J., Koller, D., Russell, S., & Kanazawa, K. 1997. Adaptive probabilistic networks with hidden variables. *Machine Learning*, **29**(2-3), 213–244.

Boser, B., Guyon, I., & Vapnik, V. 1992. An training algorithm for optimal margin classifiers. *Pages 144–152 of: Fifth Annual Workshop on Computational Learning Theory*.

Breiman, L. 1995. Better Subset Regression Using the Nonnegative Garrote. *Technometrics*, **37**(4), 373–384.

Brown, L.E., & Tsamardinos, I. 2008. *Markov Blanket-Based Variable Selection in Feature Space*. Tech. rept. TR-08-XX. Vanderbilt Univeristy.

Chang, Chih-Chung, & Lin, Chih-Jen. 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Cheng, J., Hatzis, C., Hayashi, H., Krogel, M., Morishita, S., Page, D., & Sese, J. 2002. KDD Cup 2001 Report. *ACM SIGKDD Explorations*, **3**(2), 1–18.

Chickering, DM. 2002. Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, **3**(1), 507–554.

Cooper, GF. 1997. A Simple Constraint-Based Algorithm for Efficiently Mining Observational Databases for Causal Relationships. *Data Mining and Knowledge Discovery*, **1**(2), 203–224.

Cowell, RG, Dawid, AP, Lauritzen, SL, & Spiegelhalter, DJ. 1999. *Probabilistic Networks and Expert Systems*. Springer.

Dash, Denver. 2005. Restructing dynamic causal systems in equilibrium. *In: Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AIStats)*.

Degroeve, S., De Baets, B., Van de Peer, Y., & Rouze, P. 2002. Feature subset selection for splice site prediction. *Bioinformatics*, **18**(suppl. 2), S75–S83.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. 2004. Least angle regression. *Annals of Statistics*, **32**, 407–499.

Fan, J., & Lv, J. 2008. Sure Independence Screening for Ultrahigh Dimensional Feature Space. *Journal of Royal Statistical Soceity, Ser. B*, **70**, 849–911.

Florez-Lopez, R. 2002. Reviewing RELIEF and its extensions: a new approach for estimating attributes considering high-correlated features. *Pages 605–608 of: Proceedings of the IEEE International Conference on Data Mining (ICDM)*.

Frank, I.E., & Friedman, J.H. 1993. A Statistical View of Some Chemometrics Regression Tools. *Technometrics*, **35**(2), 109–135.

Friedman, N. 2004. Inferring Cellular Networks Using Probabilistic Graphical Models. *Science*, **303**(5659), 799–805.

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. 2000. Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology*, **7**(3/4), 601–620.

Fu, LD. 2005. *A Comparison of State-of-the-Art Algorithms for Learning Bayesian Network Structure from Continuous Data*. M.Phil. thesis, Vanderbilt University.

Fu, W. 1998. Penalized Regression: The Bridge versus the Lasso. *Journal of Computational and Graphical Statistics*, **7**(3), 397–416.

Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., & Haussler, D. 2000. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**(10), 906–914.

Gilad-Bachrachy, R., Navotz, A., & Tishby, N. 2004. Margin-Based Feature Selection - Theory and Algorithms. *In: Proceedings of the 21st International Conference on Machine Learning*.

Glymour, C, & Cooper, GF. 1999. *Computation, Causation, and Discovery*. Menlo Park, CA: AAAI Press / MIT Press.

Guyon, I, & Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, **3**(March), 1157–1182.

Guyon, I, Weston, J, Barnhill, S, & Vapnik, V. 2002a. Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**(1-3), 389–422.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. 2002b. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, **46**(1-3), 389–422.

Guyon, I., Bitter, H-M., Ahmed, Z., Brown, M., & Heller, J. 2003 (December). Multivariate Non-Linear Feature Selection with Kernel Multiplicative Updates and Gram-Schmidt Relief. *In: Proceedings of the BISC FLINT - CBBI Workshop*.

Guyon, I, Azar, ARS, Weston, J, Elisseeff, A, Bakir, G, Sinz, F, Chang, C-C, & Jun-Cheng, C-J(Developers). 2009. *Challenge Learning Object Package (CLOP)*. http://clopinet.com/CLOP/.

Hall, M.A. 2000. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. *Pages 359–366 of: Proceedings of the International Conference on Machine Learning*.

Hardin, D., Tsamardinos, I., & Aliferis, C.F. 2004. A Theoretical Characterization of Linear SVM-Based Feature Selection. *In: Twenty-First International Conference on Machine Learning*.

Hartemink, AJ, Gifford, DK, Jaakkola, TS, & Young, RA. 2002. Combining Location and Expression Data for Principled Discovery of Genetic Regulatory Network Models. *Pages 437–449 of: In Proceedings of the Pacific Symposium on Biocomputing*, vol. 7. PSB.

Hashimoto, RF, Kim, S, Shmulevich, I, Zhang, W, Bittner, ML, & Dougherty, ER. 2004. Growing genetic regulatory networks from seed genes. *Bioinformatics*, **20**(8), 1241–1247.

Hastie, T., Tibshirani, R., & Friedman, J. 2001. *Elements of Statistical Learning: data mining, inference, and prediction*. New York, NY: Springer-Verlag.

Haussler, D. 1997. A Brief Look at Some Machine Learning Problems in Genomics. *Pages 109–113 of: Proccedings of COLT*.

Heckerman, DE, Horvitz, EJ, & Nathwani, BN. 1992. Towards normative expert systems. I. The Pathfinder project. *Methods of Information in Medicine*, **31**, 90–105.

Hersh, W.R., Buckley, C., Leone, T.J., & D.H., Hickam. 1994. OHSUMED: An interactive retrieval evaluation and new large test collection for research. *Pages 192–201 of: 17th Annual ACM SIGIR Conference*.

Hoerl, A.E., & Kennard, R.W. 1970. Ridge Regression: Applications to Nonorthogonal Problems. *Technometrics*, **12**(1), 69–82.

Inza, I., Larranaga, P., Blanco, R., & Cerrolaza, A.J. 2004. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, **31**(2), 91–103.

Jensen, Claus Skaaning. 1997 (May). *Blocking Gibbs Sampling for Inference in Large and Complex Bayesian Networks*. Ph.D. thesis, Aalborg University, Denmark.

Joachims, T. 2002. *Learning to classify text using support vector machines*. Kluwer.

John, G.H., Kohavi, R., & Pfleger, K. 1994. Irrelevent feature and the subset selection problem. *In: Proceedings of the International Conference on Machine Learning*.

KDD Cup 2001. 2001. *DuPont Pharmaceuticals Research Laboratories, http://www.cs.wisc.edu/d̃page/kddcup2001/*.

Kira, K, & Rendell, LA. 1992. A Practical Approach to Feature Selection. *Pages 249–256 of: Proceedings of the Ninth International Workshop on Machine Learning (ICML 1992)*. Morgan Kaufmann.

Kohavi, R., & John, G. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273–324.

Koller, D., & Sahami, M. 1996 (July). Toward Optimal Feature Selection. *Pages 284–292 of: Proceedings of the Thirteenth International Conference in Machine Learning (ICML-96)*. ICML, Bari, Italy.

Kononenko, I. 1994. Estimating Attributes: Analysis and Extensions of RELIEF. *Pages 171–182 of: European Conference on Machine Learning (ECML)*.

Larranaga, Pedro, Calvo, Borja, Santana, Roberto, Bielza, Concha, Galdiano, Josu, Inza, Inaki, Lozano, Jose A., Armananzas, Ruben, Santafe, Guzman, Perez, Aritz, & Robles, Victor. 2006. Machine learning in bioinformatics. *Brief Bioinform*, **7**(1), 86–112.

Li, J., Wong, L., & Yang, Q. 2005. Special issue on data mining for bioinformatics. *IEEE Intelligent Systems*, **20**(6).

Lim, L.P., & Burge, C.B. 2001. A computational analysis of sequence features involved in recognition of short introns. *PNAS*, **98**(20), 11193–11198.

Ling, C.X., Noble, W.S., & Yang, Q. 2005. Special issue: Machine Learning for Bioinformatics-part 1. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **2**(2), 81–2.

Lucas, PJF, Boot, H, & Taal, BG. 1998. Computer-based decision-support in the management of primary gastric non-Hodgkin lymphona. *Methods of Information in Medicine*, **37**, 206–219.

Mani, S., & Cooper, G.F. 1999. A Study in Causal Discovery from Population-Based Infant Birth and Death Records. *In: Proceedings of the AMIA Annual Fall Symposium*.

Mani, S., Spirtes, P., & Cooper, G.F. 2006. A theoretical study of Y structures for causal discovery. *Pages 314–323 of: Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Margaritis, D, & Thrun, S. 1999. Bayesian Network Induction via Local Neighborhoods. *In: Proceedings of the Conference on Neural Information Processing Systems (NIPS-99)*MIT Press, for NIPS.

Mladenic, D., & Grobelnik, M. 1999. Feature Selection for Unbalanced Class Distribution and Naive Bayes. *Pages 258–267 of: Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*.

Moore, A., & Wong, W. 2003. Optimal Reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. *In: Proceedings of the Twentieth International Conference on Machine Learning (ICML)*.

Neapolitan, RE. 2003. *Learning Bayesian Networks*. Prentice Hall.

NIPS 2003 Challenge. 2003. *Feature Selection Challenge*. http://www.nipsfsc.ecs.soton.ac.uk/datasets/.

Peña, JM, Björkegren, J, & Tegnér, J. 2005. Growing Bayesian Network Models of Gene Networks from Seed Genes. *Bioinformatics*, **21, suppl 2**(September), ii224–ii229.

Peña, J.M., Nilsson, R., Björkegren, J., & Tegnèr, J. 2007. Towards Scalable and Data Efficient Learning of Markov Boundaries. *International Journal of Approximate Reasoning*, **45**(2), 211–232.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmannn Publishers.

Pearl, J. 2000. *Causality, Models, Reasoning, and Inference*. Cambridge University Press.

Rakotomamonjy, A. 2003. Variable Selection Using SVM-based Criteria. *Journal of Machine Learning Research*, **3**, 1357–1370.

Robnik-Sikonja, M., & Kononenko, I. 2003. Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, **53**(1-2), 23–69.

Sachs, K, Perez, O, Pe'er, D, Lauffenburger, DA, & Nolan, GP. 2005. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, **308**(5721), 523–529.

Saeys, Y., Degroeve, S., Aeyels, D., Van de Peer, Y., & Rouze, P. 2003. Fast feature selection using a simple Estimation of Distribution Algorithm: A case study on splice site prediction. *Bioinformatics*, **1**(1), 1–10.

Scheines, R. 2009. *Causal Structure Search: Philosophical Foundations and Problems*. Presentation to Causality Reading Group: http://www.afia-france.org/tiki-index.php?page=Groupe+de+lecture.

Schölkopf, B., Burges, C.J.C., & Smola, A.J. (eds). 1999. *Advances in Kernel Methods: Support Vector Learning*. MIT Press.

Silverstein, C., Brin, S., Motwani, R., & Ullman, J.D. 2000. Scalable Techniques for Mining Causal Structures. *Data Mining and Knowledge Discovery*, **4**(2/3), 163–192.

Spirtes, P, Glymour, C, & Scheines, R. 1990. *Evolving Knowledge in the Natural and Behavioral Sciences*. Pittman, London. Chap. Causality from Probability, pages 181–199.

Spirtes, P., Glymour, C., & Scheines, R. 1993. *Causation, Prediction, and Search*. 1st edn. Springer/Verlag.

Spirtes, P, Glymour, C, & Scheines, R. 2000. *Causation, Prediction, and Search*. 2nd edn. Cambridge, MA: MIT Press.

Statnikov, A. 2009. *Algorithms for Discovery of Multiple Markov Boundaries: Application to the Molecular Signature Multiplicity Problem*. Ph.D. thesis, Vanderbilt University.

Statnikov, A, Hardin, D, & Aliferis, CF. 2006. Using SVM Weight-Based Methods to Identify Causally Relevant and Non-Causally Relevant Variables. *In: NIPS 2006 Workshop on Causality and Feature Selection*.

Statnikov, A, Tsamardinos, I, Brown, LE, & Aliferis, CF. 2009. *Challenges in Causality Volumn I: Causation and Prediction Challenge*. Microtome Publishing. Chap. Causal Explorer: A Matlab Library of Algorithms for Causal Discovery and Variable Selection for Classification.

Tetrad Project. *Tetrad Project*. http://www.phil.cmu.edu/projects/tetrad/.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of Royal Statistical Soceity, Ser. B*, **58**(1), 267–288.

Tong, S, & Koller, D. 2001. Active Learning for Structure in Bayesian Networks. *Pages 863–869 of: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. IJCAI.

Tsamardinos, I., & Aliferis, C. F. 2003. Towards Principled Feature Selection: Relevancy, Filters and Wrappers. *In: Ninth International Workshop on Artificial Intelligence and Statistics (AI and Stats 2003)*. AISTATS.

Tsamardinos, I., Aliferis, C. F., & Statnikov, A. 2003a. Algorithms for Large Scale Markov Blanket Discovery. *Pages 376–381 of: The Sixteenth International FLAIRS Conference*. FLAIRS.

Tsamardinos, I, Aliferis, CF, Statnikov, A, & Brown, LE. 2003b (March). *Scaling-Up Bayesian Network Learning to Thousands of Variables Using Local Learning Techniques*. Tech. rept. TR-03-02. Vanderbilt University.

Tsamardinos, I, Aliferis, CF, & Statnikov, A. 2003c. Time and Sample Efficeint Discovery of Markov Blankets and Direct Causal Relations. *Pages 673–678 of: Proceedings of Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC: ACM Press, for ACM SIGKDD.

Tsamardinos, I, Aliferis, CF, & Statnikov, A. 2003d. *Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations*. Tech. rept. DSL-03-02. Dept. of Biomedical Informatics, Nashville, TN USA.

Tsamardinos, I, Statnikov, A, Brown, LE, & Aliferis, CF. 2006a. Generating Realistic Large Bayesian Networks by Tiling. *In: The 19th International FLAIRS Conference*.

Tsamardinos, I, Brown, LE, & Aliferis, CF. 2006b. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, **65**(1), 31–78.

Tusher, V.G., Tibshirani, R., & Chu, G. 2001. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, **98**, 5116–5121.

Tuv, E., Borisov, A., Runger, G., & Torkkola, K. 2009. Feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination. *Journal of Machine Learning Research*, **10**(July), 1341–1366.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag.

Vapnik, V. 1998. *Statistical Learning Theory*. New York, NY: Wiley.

Verma, T, & Pearl, J. 1988. Causal networks: Semantics and expressiveness. *Pages 69–78 of: Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence*Elsevier Science Publishing Co, for UAI.

Verma, T, & Pearl, J. 1990. Equivalence and synthesis of causal models. *In: Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI-90)*. UAI.

Verma, T, & Pearl, J. 1991. A Theory of Inferred Causation. *Pages 441–452 of: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*Morgan Kaufmann Publishers, for KR.

Wang, Y., Tetko, I.V., Hall, M.A., Frank, E., Facius, A., Mayer, K.F., & Mewes, H.W. 2005. Gene selection from microarray data for cancer classification–a machine learning approach.

Computational Biology and Chemistry, **29**(1), 37–46.

WCCI 2006 Challenge. 2006. *Performance Prediction Challenge*. http://www.modelselect.inf.ethz.ch/index.php.

WCCI 2008 Causality Challenge. 2008. *Causation and Prediction Challenge*. http://clopinet.com/isabelle/Projects/WCCI2008/Analysis.html.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. 2000. Feature Selection for SVMs. *Pages 668–674 of: Neural Information Processing Systems (NIPS)*.

Weston, J., Elisseeff, A., Schölkopf, B., & Tipping, M. 2003. Use of the Zero-Norm with Linear Models and Kernel Methods. *Journal of Machine Learning Research*, **3**, 1439–1461.

Woolf, PJ, Prudhomme, W, Daheron, L, Daley, GQ, & Lauffenburger, DA. 2005. Bayesian analysis of signalling networks governing embryonic stem cell fate decisions. *Bioinformatics*, **21**(6), 741–753.

Yaramakala, Sandeep, & Margaritis, Dimitris. 2005. Speculative Markov Blanket Discovery for Optimal Feature Selection. *icdm*, **0**, 809–812.

Yoo, C, & Cooper, GF. 2004. An evaluation of a system that recommends microarray experiments to perform to discover gene-regulation pathways. *Artificial Intelligence in Medicine*, **31**(2), 169–182.

Yoo, C, Thorsson, V, & Cooper, GF. 2002. Discovery of Causal Relationships in a Gene-Regulation Pathway from a Mixture of Experimental and Observational DNA Microarray Data. *Pages 498–509 of: Pacific Symposium on Biocomputing*, vol. 7.

Yu, L., & Liu, H. 2003. Feature Selection for High-Dimensional Data. *Pages 856–863 of: Proceedings of the International Conference on Machine Learning*.

Zhao, Z., & Liu, H. 2007. Searching for Interacting Features. *Pages 1156–1161 of: Proceedings of the Joint Conferences on Artificial Intelligence*.

Zou, H., & Hastie, T. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of Royal Statistical Soceity, Ser. B*, **67**(2), 301–320.