Causation and Prediction Challenge

Challenges in Machine Learning, Volume 2

Causation and Prediction Challenge

Challenges in Machine Learning, Volume 2

Isabelle Guyon, Constantin Aliferis,
Greg Cooper, André Elisseeff,
Jean-Philippe Pellet, Peter Spirtes, and
Alexander Statnikov, editors

Nicola Talbot, production editor

Microtome Publishing
Brookline, Massachusetts
www.mtome.com

Causation and Prediction Challenge
Challenges in Machine Learning, Volume 2

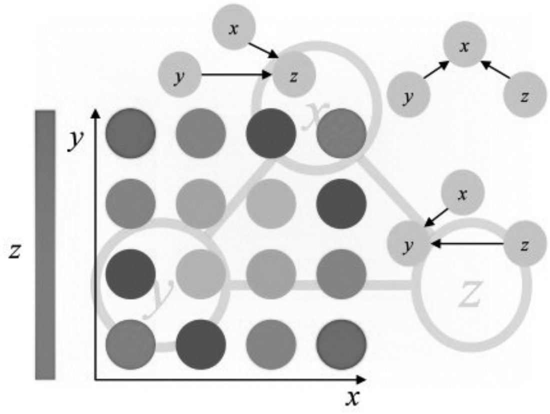
Isabelle Guyon, Constantin Aliferis,
Greg Cooper, André Elisseeff,
Jean-Philippe Pellet, Peter Spirtes, and
Alexander Statnikov, editors

Nicola Talbot, production editor

ISBN-13: 978-0-9719777-2-3

Causality Workbench

⟨http://clopinet.com/causality⟩



Foreword

For someone like me, who 20 years ago was laboring to get anyone in any discipline to take causal Bayes networks seriously, the work described in the book signals a major victory, a battle that will be looked back on as a turning point, even if the war is still undecided.

Much of the mathematical representation of causal systems that underlies the work described in this book was worked out in the late 1980s and early 1990s. For fully a decade, however, only a handful of serious scientists worked on developing, testing, and applying algorithms for causal discovery based on this representation. As a result, the techniques and even the representational ideas are still unknown in large swaths of the academy, even those areas devoted primarily to causal science. Most statisticians don't know or teach it, only a few economists work in the area, and in epidemiology, which is almost single mindedly devoted to using observational studies to learn about causal hypotheses, Sir Bradford Hill's 1964 "criteria for causality" are still the standard and if you mention ideas like d-separation and the collider problem you risk getting thrown out of the conference dinner with no dessert. Thank goodness for computer science.

What the causality workbench team has done here is singular—the team has managed to engage the massive talent of the machine learning and computer science community—and then set them to work on the daunting task of causal discovery in realistic scientific settings. To do this, the team constructed a web-based "Causality Workbench" and populated it with simulated and real data sets, discovery challenges (with prizes!) for these datasets, and a Matlab library of causal discovery software (the *Causal Explorer Software Library*). The data sets include gene expression data, drug discovery data, and census data, and the challenges involve using observational data to predict the effect of specific interventions, using mixed data (observational and interventional) to predict the effect of interventions not yet performed, and other prediction tasks that require inference not only from a sample to the population from which it was drawn, but to other populations from which we have no sample (because intervention changes the population). Unlike toy problems that we use to teach causal discovery, these datasets involve complicated dependencies, interactions, non-linear relationships, and sets of variables that include discrete, continuous, ordinal, and categorical variables. The work on these problems, which is described more than ably by the editors and the authors, combines techniques from machine learning, e.g., support vector machines, classical statistics, e.g., ridge regression and Bernoulli mixture models, and causal discovery (collider discovery and Markov blankets), to produce real advances in causal discovery.

Besides the obvious quality and ingenuity of the work, I was struck by how wide a community of scholarship the Causality Challenge has created. The authors of featured articles or of competition winning algorithms hail from all over the world. Australia, China, Crete, England, France, Germany, Mexico, Pakistan, and Taiwan are all represented. The list of "registered users" of the *Causal Explorer Software Users* includes researchers from 58 of the top research universities in the US, including Carnegie Mellon, Cornell, Duke, Harvard, Johns Hopkins, MIT, Northwestern, Princeton, Stanford, UC Berkeley, UCLA, the University of Pennsylvania, and Yale.

Clearly the work is not done. What is done, however, is the creation of a serious scientific community devoted to developing and applying reliable and fast computational methods for finding causal, not just predictive, models of the world. The Causality Workbench Team deserves high praise, not just for inspiring all the excellent work that follows in this book, but for doing so much to bring together and focus so much talent, and to lay the foundation for a bright future. The science of causal discovery is coming of age.

Richard Scheines
Professor of Philosophy, Machine Learning, and Human-Computer Interaction
Carnegie Mellon University

Preface

The Causality Workbench Team was founded in January 2007 with the objective of evaluating methods for solving causal problems. The problem of attributing causes to effects is pervasive in science, medicine, economy and almost every aspects of our everyday life involving human reasoning and decision making. Advancing the methodology for reliably determining causal relationships would therefore have an immediate and important impact, both economical and fundamental. The goal of determining causal relationships is to predict the consequences of given actions or manipulations. For instance, the effect of taking a drug on health status, or the effect of reducing taxes on the economy. This is fundamentally different from making predictions from observations. Observations imply no experimentation, no interventions on the system under study, whereas actions introduce a disruption in the natural functioning of the system. The canonical way of determining whether events are causally related is to conduct controlled experiments in which the system of interest is "manipulated" to verify hypothetical causal relationships. However, experimentation is often costly, infeasible or unethical. This has prompted a lot of recent research on learning causal relationships from available observational data. These methods can unravel causal relationships to a certain extent, but must generally be complemented by experimentation.

The need for assisting policy making and the availability of massive amounts of "observational" data triggered a proliferation of proposed causal discovery techniques. Each scientific discipline has its favorite approach (e.g. Bayesian networks in biology and structural equation modeling in social sciences, not necessarily reflecting better match of techniques to domains, but rather historical tradition. Standard benchmarks are needed to foster scientific progress, but the design of a good causal discovery benchmark platform, which is not biased in favor a particular model or approach, is not trivial. To stimulate research in causal discovery, the Causality Workbench Team created a platform in the form of a web service, which will allow researchers to share problems and test methods. See http://clopinet.com/causality. This volume gathers the material of the first causality challenge organized by the Causality Workbench Team for the World Congress in Artificial Intelligence (WCCI), June 3, 2008 in Hong-Kong. Most feature selection algorithms emanating from machine learning do not seek to model mechanisms: they do not attempt to uncover cause-effect relationships between feature and target. This is justified because uncovering mechanisms is unnecessary for making good predictions in a purely observational setting. Usually the samples in both the training and tests sets are assumed to have been obtained by identically and independently sampling from the same "natural" distribution. In contrast, in this challenge, we investigate a setting in which the training and test data are not necessarily identically distributed. For each task (e.g. REGED, SIDO, etc.), we have a single training set, but several test sets (associated with the dataset name, e.g. REGED0, REGED1, and REGED2). The training data come from a so-called "natural distribution", and the test data in version zero of the task (e.g. REGED0) are also drawn from the same distribution. We call this test set "unmanipulated test set". The test data from the two other versions of the task (REGED1 and REGED2) are "manipulated test sets" resulting from interventions of an external agent, which has "manipulated" some or all the variables in a certain way. The effect of such manipulations is to disconnect the manipulated variables from their natural causes. This may affect the predictive power of a number of variables in the system, including the manipulated variables. Hence, to obtain optimum predictions of the target variable, feature selection strategies should take into account such manipulations.

The book contains a collection of papers first published in JMLR W&CP, including a paper summarizing the results of the challenge and contributions of the top ranking entrants. We added in appendix fact sheets describing the methods used by participants and a technical report with details on the datasets. The book is complemented by a web site from which the datasets can be downloaded and post-challenge submissions can be made to benchmark new algorithms, see http://www.causality.inf.ethz.ch/challenge.php.

November 2009

The Causality Workbench Team:

Isabelle Guyon
Clopinet, California
isabelle@clopinet.com

Constantin Aliferis
New-York University, New-York
constantin.aliferis@nyumc.org

Greg Cooper
University of Pittsburgh, Pennsylvania
gfc@pitt.edu

André Elisseeff
IBM Research, Zürich
ael@zurich.ibm.com

Jean-Philippe Pellet
IBM Research and ETH, Zürich
jep@zurich.ibm.com

Peter Spirtes
Carnegie Mellon University, Pennsylvania
ps7z@andrew.cmu.edu

Alexander Statnikov
New York University
alexander.statnikov@med.nyu.edu

Table of Contents

Papers published in JMLR W&CP

Design and Analysis of the Causation and Prediction Challenge 1
Isabelle Guyon, Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov; JMLR W&CP 3:1–33, 2008.

A Strategy for Making Predictions Under Manipulation 31
Laura E. Brown and Ioannis Tsamardinos; JMLR W&CP 3:35–52, 2008.

Feature Ranking Using Linear SVM 47
Yin-Wen Chang and Chih-Jen Lin; JMLR W&CP 3:53–64, 2008.

Random Sets Approach and its Applications 59
Vladimir Nikulin; JMLR W&CP 3:65–76, 2008.

Bernoulli Mixture Models for Markov Blanket Filtering and Classificiation 71
Mehreen Saeed; JMLR W&CP 3:77–91, 2008.

Partial orientation and local structural learning of causal networks for prediction 85
Jianxin Yin, You Zhou, Changzhang Wang, Ping He, Cheng Zheng and Zhi Geng; JMLR W&CP 3:93–105, 2008.

Causal & Non-Causal Feature Selection for Ridge Regression 97
Gavin Cawley; JMLR W&CP 3:107–128, 2008.

Appendix I Causation and Prediction Challenge Fact Sheets

Feature selection, redundancy elimination, and gradient boosted trees 119
Alexander Borisov

Regularized and Averaged Selective Naïve Bayes Classifier 122
Marc Boullé

A Strategy for Making Predictions Under Manipulation 126
Laura Brown and Ioannis Tsamardinos

Causation, Prediction, Feature Selection and Regularization 130
Gavin Cawley

SVM-Based Feature Selection for Causation and Prediction Challenge 135
Yin-Wen Chang

Boosting Probabilistic Network for causality prediction 138
Louis Duclos-Gosselin

Dimensionality reduction through unsupervised learning 140
Nistor Grozavu

Markov blanket of the target and Norm1 linear SVM 142
Cristian Grozea

An Energy-based Model for Feature Selection 144
H. Jair Escalante, Luis Enrique

Translate Binary Variable to Continuous Variable 149
Jinzhu Jia

Univariate feature ranking and SVM classifier 151
Jianming Jin

Collider scores 153
Ernest Mwebaze and John Quinn

Random Sets Approach and its Applications 155
Vladimir Nikulin

Optimally Compressive Regression 158
Florin Popescu

Markov blanket and kernel ridge regression 161
Marius Popescu

Markov Blanket Filtering using Mixture Models 166
Mehreen Saeed

Ensemble Machine Learning Method 169
Ching-Wei Wang

Partial Orientation and Local Structural Learning of DAGs for Prediction 172
Jianxin Yin and Prof. Zhi Geng's Group

Causative Feature Selection by PC Algorithm and SVMs 175
Wu Zhili

Appendix II Technical Report Describing the Datasets of the Challenge

Introduction 179

Dataset A: REGED 184

Dataset B: SIDO 188

Dataset C: CINA 196

Dataset D: MARTI 205

Appendix A: Generation of random probes 210

Appendix B: Probe method for scoring causes & consequences 240

Appendix C: ChemTK QSAR descriptors used for SIDO 256

Appendix D: Chemical Computing Group (CCG) QSAR descriptors 262

Appendix E: Matlab code to filter MARTI data 262

Appendix III Causal Explorer Software Library

Causal Explorer*: A Matlab Library of Algorithms for Causal Discovery and Variable Selection for Classification* 267
Alexander Statnikov, Ioannis Tsamardinos, Laura E. Brown and Constantin F. Aliferis

Design and Analysis of the Causation and Prediction Challenge

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet, California

Constantin Aliferis CONSTANTIN.ALIFERIS@NYUMC.ORG
New York University, New York

Greg Cooper GFC@PITT.EDU
University of Pittsburgh, Pennsylvania

André Elisseeff AEL@ZURICH.IBM.COM
IBM Research, Zürich

Jean-Philippe Pellet JEP@ZURICH.IBM.COM
IBM Research and ETH, Zürich

Peter Spirtes PS7Z@ANDREW.CMU.EDU
Carnegie Mellon University, Pennsylvania

Alexander Statnikov ALEXANDER.STATNIKOV@MED.NYU.EDU
New York University

Editor: Neil Lawrence

Abstract

We organized for WCCI 2008 a challenge to evaluate causal modeling techniques, focusing on predicting the effect of "interventions" performed by an external agent. Examples of that problem are found in the medical domain to predict the effect of a drug prior to administering it, or in econometrics to predict the effect of a new policy prior to issuing it. We concentrate on a given target variable to be predicted (*e.g.,* health status of a patient) from a number of candidate predictive variables or "features" (*e.g.,* risk factors in the medical domain). Under interventions, variable predictive power and causality are tied together. For instance, both smoking and coughing may be predictive of lung cancer (the target) in the absence of external intervention; however, prohibiting smoking (a possible cause) may prevent lung cancer, but administering a cough medicine to stop coughing (a possible consequence) would not. We propose four tasks from various application domains, each dataset including a training set drawn from a "natural" distribution in which no variable are externally manipulated and three test sets: one from the same distribution as the training set and two corresponding to data drawn when an external agent is manipulating certain variables. The goal is to predict a binary target variable, whose values on test data are withheld. The participants were asked to provide predictions of the target variable on test data and the list of variables (features) used to make predictions. The challenge platform remains open for post-challenge submissions and the organization of other events is under way (see http://clopinet.com/causality).

Keywords: challenge, competition, causality, causal discovery, feature selection, intervention, manipulation.

1. Introduction

The problem of attributing causes to effects is pervasive in science, medicine, economics and almost every aspect of our everyday life involving human reasoning and decision making. One important goal of causal modeling is to unravel enough of the data generating process to be able to make predictions under manipulations of the system of interest by an external agent (*e.g.,* experiments). Being able to predict the results of actual or potential experiments (consequences or effects)[1] is very useful because experiments are often costly and sometimes impossible or unethical to perform. For instance, in policy-making, one may want to predict "the effect on a population's health status" of "forbidding individuals to smoke in public places" before passing a law. This example illustrates the case of an experiment which is possible, but expensive. On the other hand, forcing people to smoke would constitute an unethical experiment.

The need for assisting policy making and the availability of massive amounts of "observational" data has prompted the proliferation of proposed causal discovery techniques. These techniques estimate the structure of the data generating process from which the effect of intervention can be estimated. Each scientific discipline has its favorite approach (*e.g.,* Bayesian networks in biology and structural equation modeling in the social sciences), not necessarily reflecting a better match of techniques to domains, but rather the historical tradition. Standard benchmarks are needed to foster scientific progress. In organizing a challenge for WCCI on the theme of causality, our goals included:

- Stimulating the causal discovery community to make progress by exposing it to large datasets, whose size is more typical of data mining and machine learning tasks than causal learning.
- Drawing the attention of the computational intelligence community to the importance of causal modeling and discovery problems and the opportunities to explore machine learning and data mining techniques.
- Pointing out possible limitations of current methods on some particularly difficult problems.

The last item is especially relevant for feature selection algorithms emanating from machine learning as most current machine learning methods do not attempt to uncover cause-effect relationships between features and target. This is justified for a prediction task where training and tests sets are obtained by drawing samples identically and independently from the same "natural" distribution. We call this a purely "observational" setting. In that setting, statistical predictive models do not need to model data generative mechanisms and both causal and consequential features may be predictive of a certain target variable. For instance both smoking and coughing are predictive of respiratory disease; one is a cause and the other a symptom (consequence). In contrast, in this challenge, we investigated a setting in which *the training and test data are not necessarily identically distributed.* Test data may be drawn from a post-manipulation distribution that is distinct from the unmanipulated "natural" distribution from which training data are drawn. This problem is related to the more general problem of "distribution shift" or "covariate shift", which has recently gained the attention of the machine learning community and was the object of a challenge (Quiñonero Candela et al., 2007). In the particular case we are interested in, the post-manipulation distribution results from **actions** or **interventions** of an external agent who is forcing some variables to assume particular values rather than letting the data generative system produce values according to its own dynamics. Acting on a cause of en event can change the event, but acting on a consequence cannot. For instance, acting on a cause of disease like smoking can change the disease state, but acting on the symptom (coughing) cannot. Thus it

1. In this paper, we will use interchangeably "manipulation" or "intervention" and "consequence" or "effect".

is extremely important to distinguish between causes and effects to predict the consequences of actions on a given target variable.

The main objective of the challenge was to predict a binary target variable (classification problem) from a set of candidate predictive variables, which may be binary or continuous. For each task of the challenge (*e.g.,* REGED, SIDO, etc.), we have a single training set, but several test sets (associated with the dataset name, *e.g.,* REGED0, REGED1, and REGED2). The training data come from a so-called "natural distribution", and the test data in version zero of the task (*e.g.,* REGED0) are also drawn from the same distribution. We call this test set a "natural" or "unmanipulated" test set. The test data from the two other versions of the task (*e.g.,* REGED1 and REGED2) are "manipulated" test sets resulting from **interventions** of an external agent, which has "manipulated" some or all the variables in some way (excluding the "target" or "response variable"). The effect of such manipulations is to *disconnect the manipulated variables from their natural causes*. This may affect the predictive power of a number of variables in the system, including the manipulated variables. Hence, to obtain optimum predictions of the target variable, feature selection strategies should take into account such manipulations.

In this challenge, we are focusing on causal relationships between random variables, as opposed to causal relationships between events or objects. We consider only stationary systems in equilibrium, hence eliminating the need for an explicit reference to time in our samples. This setup is typical of so-called "cross-sectional" studies in medicine (as opposed to "longitudinal" studies). In practice, this means that the samples for each version of the test set, *e.g.,* REGED0, REGED1, and REGED2, are drawn independently, according to a given distribution, which changes only between test set version. Having no explicit reference to time may be surprising to researchers new to causal modeling, since causes must always precede their effects. Causal models in this context enforce an order of evaluation of the variables, without reference to an exact timing.[2]

The type of causal relationships under consideration have often been modeled as Bayesian causal networks or structural equation models (SEM) (Pearl, 2000; Spirtes et al., 2000; Neapolitan, 2003). In the graphical representation of such models, an arrow between two variables $A \rightarrow B$ indicates the direction of a causal relationship: A causes B. A node in of the graph, labeled with a particular variable X, represents a mechanism to evaluate the value of X given the parent node variable values. For Bayesian networks, such evaluation is carried out by a conditional probability distribution $P(X|Parents(X))$ while for structural equation models it is carried out by a function of the parent variables, plus some noise. Learning a causal graph can be thought of as a model selection problem: Alternative graph architectures are considered and a selection is performed, either by ranking the architectures with a global score (*e.g.,* a marginal likelihood, or a penalty-based cost function), or by retaining only graphs that fulfill a number of constraints such as dependencies or independencies between subsets of variables.

Bayesian networks and SEMs provide a convenient language to talk about the type of problem we are interested in, but our setting does not preclude of any particular model. Some of the data used in the challenge were generated by real unknown processes, which probably violate some commonly made causal modeling assumptions, such as "causal sufficiency"[3], linearity, Gaussian noise, absence of cycles, etc. By adopting a predictive modeling perspective, we purposely took some distance with the interpretation of causal models as data generative models. The goal of the challenge was not to reverse engineer the data generative process, it is to make accurate predictions of a target variable. To sharpen this distinction, we made available only a limited amount of training data, such that the learner may not necessarily be able to reliably de-

2. When manipulations are performed, we must specify whether we sample from the distribution before or after the effects of the manipulation have propagated. Here we assume that we sample after the effects have propagated.
3. "Causal sufficiency" roughly means that there are no unobserved common causes of the observed variables.

termine all conditional dependencies and independencies. Hence, modeling strategies making radical simplifying assumptions might do better than strategies trying to be faithful to the data generative process, because of the well-known fit *vs.* robustness (or bias *vs.* variance) tradeoff.

2. General setting

We created a web site from which data and instructions on how to participate were outlined: `http://clopinet.com/causality`. This first causality challenge is part of a larger program, which we initiated, called the "causality workbench"; the web site hosts repositories of code, data, models, publications and other events, including challenges and teleconference seminars. Our first challenge started on December 15, 2007 and ended on April 30, 2008. Four datasets were proposed and progressively introduced (the last one being released 2 months prior the end of the challenge). More details on the datasets are found in Section 3.

Our challenge is formatted in a similar way to most machine learning problems: pairs of training examples $\{\mathbf{x}, y\}$ are provided. The goal is to predict the target variable y for new test instances of $\mathbf{x}$. The elements of vector $\mathbf{x}$ are interchangeably called "variables" or "features" in this paper. Unlike most machine learning problems, the training and test sets are not always distributed similarly. We provide large test sets to obtain statistically significant results. Both the training and the unlabeled test sets were provided from the beginning of the competition. We required that the participants would not use the unlabeled test data to train their models, and this rule was enforced by verifying the code of the best ranking entrants after the end of the challenge (see Appendix B). This rule was motivated by several considerations: (1) We are investigating problems in which only "observational" training data are available for model building. Test data are not supposed to be available at model building time; we use them only to test the ability of our model to make predictions about the effect of hypothetical actions performed on the system in the future. (2) In a challenge, we need very large test sets to obtain small error bars on the participant performances, otherwise most differences between algorithms would not be statistically significant. However, such large amount of "manipulated" test data would not be available all at once in many real world situations.

Prediction results and features sets could be submitted on-line to get immediate feed-back, as a means of stimulating participation. To limit the amount of knowledge that could be gained from viewing test set results, the participants were only informed about the quartile of their method's performances. In previous challenges we organized (Guyon et al., 2006a,,), we provided feed-back on a small validation set, whose target values were released shortly before the end of the challenge, and we used a separate larger test to perform the final evaluation. In this challenge, we developed this new way of providing feed-back (using performance quartiles) because information about the post-manipulation distribution (distinct from the training data "natural" distribution) could be induced from a more detailed form of performance feed-back on a validation set. The quartile method achieves essentially the same goal of stimulating the participants while simplifying the challenge protocol.

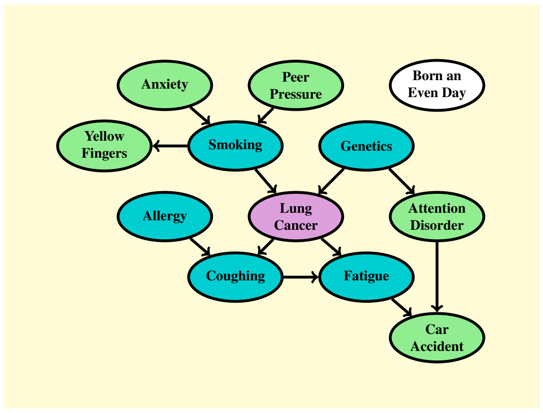
Another difference compared to our previous challenges is that we did not request that the participants return results on all tasks of the challenge. For each task, they were only required to return predictions on all three versions of any given test set (manipulated or not). In this way, we intended to lower the level of effort of participation because we knew many algorithms lend themselves only to certain kinds of data. To encourage participants to submit results on more than one task, we set up an exponential reward system: a prize of \$100 was promised for winning on any of the 4 tasks, but the progression of the rewards for winning on 2, 3, or 4 datasets was \$400, \$900, and \$1600. This successfully encouraged entrants to submit on all datasets. Another final difference from previous challenges is that we authorized only one final entry (as opposed to 5 in previous challenges) to compensate for the fact that participants had 4 chances of winning (one for each dataset). In this way, we limited the statistical risk that the winning entry be better only "by chance". However, we did allow submissions of multiple prediction results for *nested subsets of variables*, with the purpose of obtaining performance curves as a function of number of features. In Section 5, our initial analysis is based on the best result in the performance curve for each participant. We complemented it by an analysis making pairwise comparisons of entries at the same number of features, to account for a possible bias detrimental to the participants who provided single predictions.

To introduce the participants to the problem of making predictions under interventions, we provided a tutorial (Guyon et al., 2007), and we created a toy example, which was not part of the challenge, but which was interfaced to the challenge platform in the same way as the other datasets. The participants could use it for practice purposes, and we provided guidance on how to solve the problem on the web site. We briefly describe this example, illustrated in Figure 1, to clarify the challenge. More details are found on the website of the challenge.

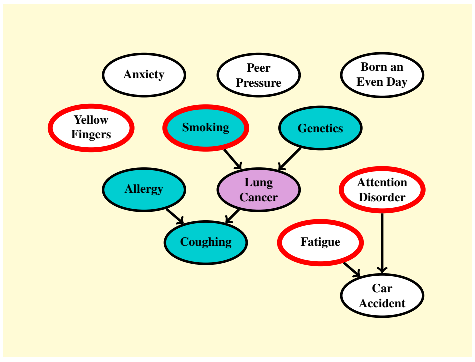
LUCAS0: The toy example of Figure 1-a models the problem of predicting lung cancer as a causal network. Each node represents a variable/feature and the arcs represent causal relationships, *i.e.,* $A \rightarrow B$ represents that A is a cause of B. The target variable is "Lung Cancer". Each node in the graph is associated with a table of conditional probabilities $P(X = x|Parent_1(X) = p_1, Parent_2(X) = p_2, ...)$ defining the "natural" distribution. The generative model is a Markov process (a so-called "Bayesian network"), so the state of the children is stochastically determined by the states of the parents. The values must be drawn in a certain order, so that the children are evaluated after their parents. Both the training and test sets of LUCAS0 are drawn according the natural distribution. In the figure, we outline in dark green the Markov blanket of the target, which includes all targets' parents (node immediate antecedents), children (node immediate descendants), and spouses (immediate antecedents of an immediate descendant). The Markov blanket (MB) is the set of variables such that the target is independent of all other variables given MB.[4] It is widely believed that, if the MB were perfectly known, adding more variables to the feature set would be unnecessary to make optimal predictions of the target variable. However, this statement depends on the criterion of optimality and is true only in the sample limit and if the predictor is asymptotically unbiased (Tsamardinos and Aliferis, 2003). For example, a linear classifier may benefit from the inclusion of non-MB features, even in the sample limit and with perfect knowledge of the MB, if the functional relation of the target and the MB is non-linear. In this challenge, the goal is not to discover the MB, it is to make best predictions of the target variable on test data.

LUCAS1: In the example of Figure 1-b, the training data are the same as in LUCAS0. We model a scenario in which an external agent manipulates some of the variables of the system, circled in red in the figure (Yellow Fingers, Smoking, Fatigue, and Attention Disorder). The intention of such manipulations may include disease prevention or cure. The external agent sets the manipulated variables to desired values, hence "disconnecting" those variables from their parents. The other variables are obtained by letting the system evolve according to its own dynamics. As a result of manipulations, many variables may become disconnected from the target and the Markov blanket (MB) may change. If the identity of the manipulated variables is revealed (as in the case of REGED1 and MARTI1), one can deduce from the graph of the natural distribution inferred from training data which variables to exclude from the set of predictive variables. In particular, the MB of the post-manipulation distribution is a restriction of the

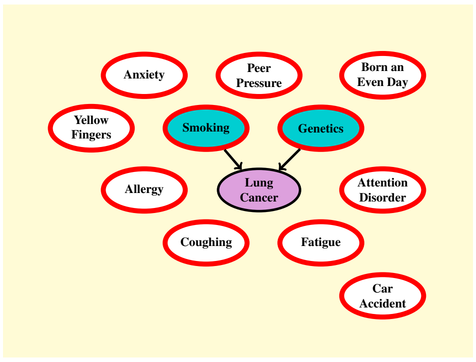
4. Other definitions of the Markov blanket are possible. Our definition coincides with what other authors call Markov boundary or "minimal" Markov blanket. Although we refer to "the" Markov blanket, for some distributions it is not unique and it does not always coincide with the sets of parents, children and spouses. But we limit ourselves to this case in the example, for simplicity.



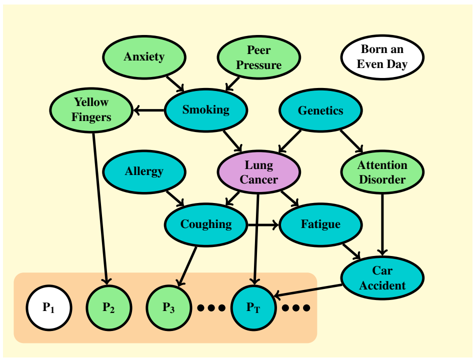
(a) LUCAS0



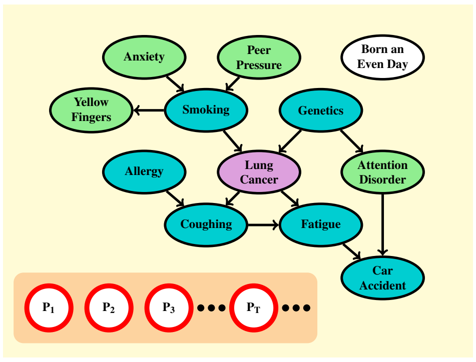
(b) LUCAS1



(c) LUCAS2



(d) LUCAP0



(e) LUCAP1

Figure 1: **Lung cancer toy example.** The dark green nodes represents the minimal Markov blanket or "Markov boundary" (MB) of the target variable "Lung Cancer". The white nodes are independent of the target. Given the MB, both white and light green nodes are (conditionally) independent of the target. The manipulated nodes are emphasized in red. As a result of being manipulated, they are disconnected from their original causes and the MB is restricted to the remaining dark green nodes. See text.

MB of the natural distribution resulting from the removal of manipulated children and spouses whose children are all manipulated (unless it is also a parent of the target).

LUCAS2: In Figure 1-c we manipulated all the variables except the target. As a result, only the direct causes of the target are predictive, and they coincide with the Markov blanket (MB) of the post-manipulation distribution.

LUCAP0: In Figure 1-d, we are modeling the following situation: Imagine that we have REAL data generated from some UNKNOWN process (we do not know the causal relationships among variables). Further, for various reasons, which may include practical reasons, ethical reasons, or cost, we are unable to carry out any kind of manipulation on the real variables, so we must resort to performing causal discovery and evaluating the effectiveness of our causal discovery using *unmanipulated* data (data drawn from the natural distribution). To that end, we add a large number of artificial variables called "probes", which are generated from some functions (plus some noise) of subsets of the real variables. We shuffle the order of all the variables and probes not to make it too easy to identify the probes. For the probes we (the organizers) have perfect knowledge of the causal relationships. For the other variables, we only know that some of them (light green nodes) might be predictive while not belonging to the MB, and some of them (dark green nodes) might belong to the MB. The members of the MB include some real variables and some probes. To assess feature selection methods, we use the probes by computing statistics such as the fraction of non-MB probes in the feature subset selected.

LUCAP1 and LUCAP2: While we cannot manipulate the real variables in our model setup, we can manipulate the probes. The probe method allows us to conservatively evaluate causal feature selection algorithms, because we know that the output of an algorithm should not include any probe for a distribution where all probes are manipulated. The test sets of LUCAP1 and LUCAP2 (Figure 1-e) are obtained by manipulating all probes (in every sample) in two different ways. The training data are the same as in LUCAP0. Knowing that we manipulated all probes, and that probes can only be non-causes of the target, a possible strategy is to select only features that are causes of the target.[5] If this strategy is followed, the fraction of probes in the feature set selected allows us to compute an estimate the fraction of non-causes wrongly selected.[6]

3. Description of the datasets

We use two types of data:

- **Re-simulated data:** We train a "causal" model (a causal Bayesian network or a structural equation model) with real data. The model is then used to generate artificial training and test data for the challenge. Truth values of causal relationships are known for the data generating model and used for scoring causal discovery results. REGED is an example of re-simulated dataset.
- **Real data with probe variables:** We use a dataset of real samples. Some of the variables may be causally related to the target and some may be predictive but non-causal. The nature of the causal relationships of the variables to the target is unknown (although domain knowledge may allow us to validate the discoveries to some extent). We have added to the set of real variables a number of distractor variables called "probes", which are generated by an artificial stochastic process, including explicit functions of some of

5. Note however that some of the real variables that are non-causes may be predictive, so eliminating all non-causes of the target is a sure way to eliminate all probes but not necessarily an optimum strategy.
6. The validity of the estimation depends on many factors, including the number of probes and the distributional assumptions of non-causes made in the probe data generative process.

the real variables, other artificial variables, and/or the target. All probes are non-causes of the target, some are completely unrelated to the target. The identity of the probes in concealed. The fact that truth values of causal relationships are known only for the probes affects the evaluation of causal discovery, which is less reliable than for artificial data.

The **training data** and test sets labeled 0 are generated from a so-called "natural" pre-manipulation distribution. The variable values are sampled from the system when it is allowed to evolve according to its own dynamics, after it has settled in a steady state. For the probe method, the system includes the artificial probe generating mechanism. **Test sets** labeled 1 and 2 are generated from a so-called **post-manipulation distribution**. An external agent performs an "intervention" on the system. Depending on the problem at hand, interventions can be of several kinds, *e.g.,* clamping one or several variables to given values or drawing them from an alternative distribution, then sampling the other variables according to the original conditional probabilities. In our design, **the target variable is never manipulated**. For the probe method, since we do not have the possibility of manipulating the real variables, we only manipulate the probes. The effect of manipulations is to disconnect the variables from their natural causes. Manipulations allow us to eventually influence the target, if we manipulate causes of the target. Manipulating non-causes should have no effect on the target. Hence, without inferring causal relationships, it should be more difficult to make predictions for post-manipulation distributions.

Table 1: **Datasets.** All target variables are binary. Each dataset has three test sets of the same size numbered 0, 1, and 2.

Dataset	Domain	Type	Features	Feat. #	Train #	Test #
REGED	Genomics	Re-simulated	Numeric	999	500	20000
SIDO	Pharmacology	Real + probes	Binary	4932	12678	10000
CINA	Econometrics	Real + probes	Mixed	132	16033	10000
MARTI	Genomics	Re-simulated	Numeric	999	500	20000

We proposed four tasks (Table 1):

REGED (REsimulated Gene Expression Dataset): Find genes which could be responsible for lung cancer. The data are "re-simulated", *i.e.,* generated by a model derived from real human lung-cancer microarray gene expression data. From the causal discovery point of view, it is important to separate genes whose activity causes lung cancer from those whose activity is a consequence of the disease. All three datasets (REGED0, REGED1, and REGED2) include 999 features (no hidden variables or missing data), the same 500 training examples, and different test sets of 20000 examples. The target variable is binary; it separates malignant samples (adenocarcinoma) from control samples (squamous cells). The three test sets differ in their distribution. REGED0: No manipulation (distribution identical to the training data). REGED1: Variables in a given set are manipulated and their identity is disclosed. REGED2: Many variables are manipulated, including all the consequences of the target, but the identity of the manipulated variables was not disclosed. When variables are manipulated, the model is allowed to evolve according to its own mechanism until the effect of the manipulations propagate.

SIDO (SImple Drug Operation mechanisms) contains descriptors of molecules which have been tested against the AIDS HIV virus. The target values indicate the molecular activity (+1 active, −1 inactive). The causal discovery task is to uncover causes of molecular activity among the molecule descriptors. This would help chemists in the design of new compounds, retaining activity, but having perhaps other desirable properties (less toxic, easier to administer). The molecular descriptors were generated programmatically from the three dimensional description of the molecule, with several programs used by pharmaceutical companies for QSAR studies

(Quantitative Structure-Activity Relationship). For example, a descriptor may be the number of carbon molecules, the presence of an aliphatic cycle, the length of the longest saturated chain, etc. The dataset includes 4932 variables (other than the target), which are either molecular descriptors (all potential causes of the target) or "probes" (artificially generated variables that are not causes of the target). The training set and the unmanipulated test set SIDO0 are similarly distributed. They are constructed such that some of the "probes" are effects (consequences) of the target and/or of other real variables, and some are unrelated to the target or other real variables. Hence, both in the training set and the unmanipulated test set, all the probes are non-causes of the target, yet some of them may be "observationally" predictive of the target. In the manipulated test sets SIDO1 and SIDO2, all the "probes" are manipulated in every sample by an external agent (*i.e.,* set to given values, not affected by the dynamics of the system) and can therefore not be relied upon to predict the target. The identity of the probes is concealed. They are used to assess the effectiveness of the algorithms to dismiss non-causes of the target for making predictions in manipulated test data. In SIDO1, the manipulation consists in a simple randomization of the variable values, whereas in SIDO2 the values are chosen to bias prediction results unfavorably, if the manipulated variables are chosen as predictors (adversarial design).

CINA (Census Is Not Adult) is derived from census data (the UCI machine-learning repository Adult database). The data consists of census records for a number of individuals. The causal discovery task is to uncover the socio-economic factors affecting higher income (the target value indicates whether the income exceeds 50K). The 14 original attributes (features) including age, workclass, education, marital status, occupation, native country, etc. are continuous, binary, or categorical. Categorical variables were converted to multiple binary variables (as we shall see, this preprocessing, which facilitates the tasks of some classifiers, complicates causal discovery). Distracter features or "probes" (artificially generated variables, which are not causes of the target) were added. In training data, some of the probes are effects (consequences) of the target and/or of other real variables. Some are unrelated to the target or other real variables. Hence, some of the probes may be correlated to the target in training data, although they do not cause it. The unmanipulated test data in CINA0 are distributed like the training data. Hence, both causes and consequences of the target might be predictive in the unmanipulated test data. In contrast, in the manipulated test data of CINA1 and CINA2, all the probes are manipulated by an external agent (*i.e.,* set to given values, not affected by the dynamics of the system) and therefore they cannot be relied upon to predict the target. In a similar way to SIDO, the difference between versions 1 and 2 is that in version 1 the probe values are simply randomized whereas in version 2 they are chosen in an adversarial way.

MARTI (Measurement ARTIfact) is obtained from the same data generative process as REGED, a source of simulated genomic data. Similarly to REGED the data do not have hidden variables or missing data, but a noise model was added to simulate the imperfections of the measurement device. The goal is still to find genes, which could be responsible of lung cancer. The target variable is binary; it indicates malignant samples *vs.* control samples. The feature values representing measurements of gene expression levels are assumed to have been recorded from a two-dimensional microarray 32×32. The training set was perturbed by a zero-mean correlated noise model. The test sets have no added noise. This situation simulates a case where we would be using different instruments at "training time" and "test time", *e.g.,* we would use DNA microarrays to collect training data and PCR for testing. We avoided adding noise to the test set because it would be too difficult to filter it without visualizing the test data or computing statistics on the test data, which we forbid. So the scenario is that the second instrument (used at test time) is more accurate. In practice, the measurements would also probably be more expensive, so part of the goals of training would be to reduce the size of the feature set (we did not make this a focus in this first challenge).

The problems proposed are challenging in several respects:

- Several assumptions commonly made in causal discovery are violated, including "causal sufficiency"[7], "faithfulness"[8], "linearity", and "Gaussianity".
- Relatively small training sets are provided, making it difficult to infer conditional independencies and learning distributions.
- Large numbers of variables are provided, a particular hurdle for some causal discovery algorithms that do not scale up.

More details on the datasets, including the origin of the raw data, their preparation, past usage, and baseline results can be found in a Technical Report (Guyon et al., 2008).

4. Evaluation

The participants were asked to return *prediction scores* or **discriminant values** v for the target variable on test examples, and a **list of features** used for computing the prediction scores, sorted in order of decreasing predictive power, or unsorted. The classification decision is made by setting a threshold θ on the discriminant value v: predict the positive class if $v > \theta$ and the negative class otherwise. The participants could optionally provide results for nested subsets of features, varying the subset size by powers of 2 (1, 2, 4, 8, etc.).

Tscore: The participants were ranked according to the area under the ROC curve (AUC) computed for test examples (referred to as Tscore), that is the area under the curve plotting sensitivity *vs.* (1− specificity) when the threshold θ is varied (or equivalently the area under the curve plotting sensitivity *vs.* specificity). We call "sensitivity" the error rate of the positive class and "specificity" the error rate of the negative class. The AUC is a standard metric in classification. If results were provided for nested subsets of features, the best Tscore was retained. There are several ways of estimating error bars for the AUC. We use a simple heuristic, which gives us approximate error bars, and is fast and easy to implement: we find on the AUC curve the point corresponding to the largest balanced accuracy BAC = 0.5 (sensitivity + specificity). We then estimate the standard deviation of the BAC as:

$$\sigma = \frac{1}{2}\sqrt{\frac{p_+(1-p_+)}{m_+} + \frac{p_-(1-p_-)}{m_-}}\,, \tag{1}$$

where m_+ is the number of examples of the positive class, m_- is the number of examples of the negative class, and p_+ and p_- are the probabilities of error on examples of the positive and negative class, approximated by their empirical estimates, the sensitivity and the specificity (Guyon et al., 2006b).

Fscore: We also computed other statistics, which were not used to rank participants, but used in the analysis of the results. Those included the number of features used by the participants called "Fnum", and a statistic assessing the quality of causal discovery in the feature set selected called "Fscore". As with the Tscore, we provided quartile feed-back on Fnum and Fscore during the competition. For the Fscore, we used the AUC for the problem of separating features belonging to the Markov blanket of the test set distribution *vs.* other features. Details are provided on the web site of the challenge. As it turns out, for reasons explained in Section 5, this statistic correlates poorly with the Tscore and, after experimenting with various scores, we found better alternatives.

7. "Causal sufficiency" roughly means that there are no unobserved common causes of the observed variables.
8. "Faithfulness" roughly means that every conditional independence relation that holds in the population is entailed to hold for all values of the free parameters.

Table 2: **Best scores of ranked entrants.** The table shows the results of the best entries of the ranked entrants and their corresponding scores: Top Tscore = area under the ROC curve on test data for the top ranked entries; Top Fscore = a measure of "causal relevance" of the features used during the challenge (see text). For comparison, we also include the largest reachable score, which was obtained by including reference entries made by the organizers using knowledge about the true causal relationships (Max Ts and Max Fs).

Dataset	Top Tscore		Max Ts	Top Fscore		Max Fs
REGED0	Yin-Wen Chang	1.000±0.001	1.000	Gavin Cawley	0.941±0.036	1.000
REGED1	Marius Popescu	0.989±0.003	0.998	Yin-Wen Chang	0.857±0.062	1.000
REGED2	Yin-Wen Chang	0.839±0.005	0.953	CaMML Team	1.000±0.153	1.000
SIDO0	J. Yin & Z. Geng Gr.	0.944±0.008	0.947	H. Jair Escalante	0.844±0.007	1.000
SIDO1	Gavin Cawley	0.753±0.014	0.789	Mehreen Saeed	0.724±0.007	1.000
SIDO2	Gavin Cawley	0.668±0.013	0.767	Mehreen Saeed	0.724±0.007	1.000
CINA0	Vladimir Nikulin	0.976±0.003	0.979	H. Jair Escalante	0.955±0.032	1.000
CINA1	Gavin Cawley	0.869±0.005	0.898	Mehreen Saeed	0.786±0.039	1.000
CINA2	Yin-Wen Chang	0.816±0.005	0.891	Mehreen Saeed	0.786±0.039	1.000
MARTI0	Gavin Cawley	1.000±0.001	1.000	Gavin Cawley	0.870±0.048	1.000
MARTI1	Gavin Cawley	0.947±0.004	0.954	Gavin Cawley	0.806±0.063	1.000
MARTI2	Gavin Cawley	0.798±0.006	0.827	Gavin Cawley	0.996±0.153	1.000

5. Result Analysis

5.1 Best challenge results

We declared three winners of the challenge:

- **Gavin Cawley** (University of East Anglia, UK): Best prediction accuracy on SIDO and MARTI, using Causal explorer and linear ridge regression ensembles. Prize: $400.
- **Yin Wen Chang** (National Taiwan University): Best prediction accuracy on REGED and CINA, using SVM. Prize: $400.
- **Jianxin Yin and Zhi Geng's group** (Peking University, Beijing, China): Best overall contribution, using Partial Orientation and Local Structural Learning (new original causal discovery algorithm and best on Pareto front causation/prediction, *i.e.,* with smallest Euclidian distance to the extreme point with zero error and zero features). Prize: free WCCI 2008 registration.

The top-ranking results are summarized in Table 2. These results are taken from the last entries of the ranked entrants.[9]

Following the rules of the challenge, the participants were allowed to turn in *multiple prediction results* corresponding to *nested subsets of features*. The best Tscore over all feature set sizes was then retained and the performances were averaged over all three test sets for each task REGED, SIDO, CINA, and MARTI. In this way, we encouraged the participants to rank features rather than select a single feature subset, since feature ranking is of interest for visualization, data understanding, monitoring the tradeoff "number of features"/"prediction performance", and prioritizing potential targets of action. The entries of Gavin Cawley (Cawley, 2008) and Yin Wen Chang and Chih-Jen Lin (Chang and Lin, 2008) made use of this possibility of turning

9. This table reports the results published on the web site of the challenge, using the original definition of the Fscore, whose effectiveness to assess causal discovery is questioned in Section 5.2.

in multiple results. They each won on two datasets and ranked second and third on the two others. Their average Tscore over all tasks is almost identical and better than that of other entrants (see Figure 2-b).

The participants who used nested subsets had an advantage over other participants, not only because they could make multiple submissions and be scored on the basis of the best results, but also because the selection of the best point was made with test data drawn from the post-manipulation distribution, therefore implicitly giving access to information on the post-manipulation distribution. By examining the results and the "Fact Sheets", we noticed that most participants having performed causal discovery opted to return a *single feature subset* while those using non-causal feature selection performed feature ranking and opted to return multiple predictions for *nested subsets of features*, therefore introducing a bias in the results. To compensate for that bias, we made pairwise comparisons between classifiers, at equal number of features (see details in Section 5.2). According to this new method of comparison, Jianxin Yin and Zhi Geng's group obtain the best position of the Pareto front of the Fscore *vs.* Tscore graph (see Figure 2-b). Because of this achievement and the originality of the method that they developed (Yin et al., 2008), they were awarded a prize for "best overall contribution".[10] Also noteworthy was the performances of Vladimir Nikulin (Suncorp, Australia), who ranked second on CINA and fourth on REGED and MARTI in average Tscore, based on predictions made with a single feature subset obtained with the "random subset method" (Nikulin, 2008). His average performances were as good as Jianxin Yin and Zhi Geng's group on average in the pairwise comparison of classifiers (Figure 2-b), even though his average Fscore is significantly lower.

Also worthy of attention are the entries of Marc Boullé and Laura E. Brown & Ioannis Tsamardinos, who did not compete towards the prizes (and therefore were not ranked), identified as M.B. and L.E.B. & Y.T. in the figures.[11] Marc Boullé reached Tscore=0.998 for REGED1 and Laura E. Brown & Ioannis Tsamardinos reached Tscore=0.86 on REGED2. Marc Boullé also reached Tscore=0.979 on CINA0 and Tscore=0.898 on CINA1. The entries of Marc Boullé, using a univariate feature selection method and a naïve Bayes classifier (Boullé, 2007a,) were best on REGED0 and REGED1 and on CINA0 and CINA1. The entry of Laura E. Brown & Ioannis Tsamardinos on REGED2 is significantly better than anyone else's and they are best on average on REGED. They use a novel structure-based causal discovery method (Brown and Tsamardinos, 2008). Finally, Mehreen Saeed ranked fourth and sixth on SIDO and CINA, using a novel fast method for computing Markov blankets (Saeed, 2008). She achieved the best Fscores on SIDO1&2 and CINA1&2.

All the top-ranking entrants we just mentioned supplied their code to the organizers, who could verify that they complied with all the rules of the challenge and that their results are reproducible (see Appendix B).

5.2 Causation and Prediction

One of the goals of the challenge was to test the efficacy of using causal models to make good predictions under manipulations. In an attempt to quantify the validity of causal models, we defined an Fscore (see Section 2). Our first analysis of the challenge results revealed that this score correlates poorly with the Tscore, measuring prediction accuracy. In particular, many entrants obtained a high Fscore on REGED2 and yet a poor Tscore. In retrospect, this is easily understood. We provide a simple explanation for the case of unsorted feature sets in which, for

10. As explained in Section 5.2, the original Fscore had some limitations. In Figure 2 we plot the new Fscore described in that section.
11. As per his own request, the entries of Marc Boullé (M.B.) were marked as "Reference" entries like those of the organizers and did not count towards wining the prizes; Laura E. Brown and Ioannis Tsamardinos (L.E.B. & Y.T.) could not compete because they are close collaborators of some of the organizers.

REGED, the Fscore is $0.5(tp/(tp+fn)+tn/(tn+fp))$, where tp is the number of true positive (correctly selected features), fn false negative, tn true negative, and fp false positive. REGED2 has only 2 causally relevant feature (direct causes) in the *manipulated Markov blanket*; *i.e.,* the Markov blanket of the test set distribution, which is manipulated. Most people included these two features in their feature set and obtained $tp/(tp+fn) = 1$. Since the number of irrelevant features is by comparison very large (of the order of 1000), even if the number of wrongly selected features fp is of the order of 10, $tn/(tn+fp)$ is still of the order of 1. The resulting Fscore is therefore close to 1. However, from the point of view of the predictive power of the feature set, including 10 false positive rather than 2 makes a lot of difference. We clearly see that the first Fscore we selected was a bad choice.

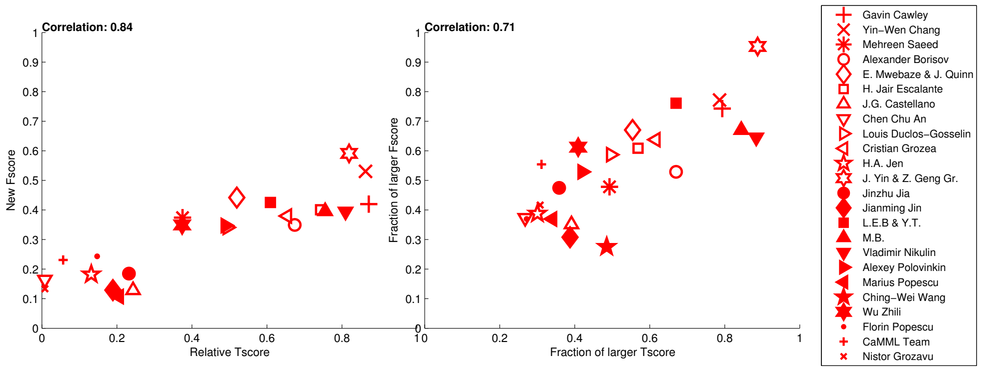
Definition of a new Fscore. We ended up using as the new Fscore the **Fmeasure** for REGED and MARTI and the **precision** for SIDO and CINA, after experimenting with various alternative measures inspired by information retrieval, see our justification below. We use the following definitions: precision = $tp/(tp+fp)$, recall = $tp/(tp+fn)$ (also called sensitivity), and Fmeasure = 2 precision recall / (precision + recall). Our explorations indicate that precision, recall, and Fmeasure correlate well with Tscore for artificially generated datasets (REGED and MARTI). The *Fmeasure*, which captures the tradeoff between precision and recall, is a good measure of feature set quality for these datasets. However, recall correlates poorly with Tscore for SIDO and CINA, which are datasets of *real variables* with added *artificial probe variables*. This is because, in such cases, we must resort in approximating the recall by the fraction of *real variables* present in the selected feature set, which can be very different from the true recall (the fraction of truly relevant variables). Hence, if many real variables are irrelevant, a good causal discovery algorithm that eliminates them would get a poor estimated recall. Hence, we can only use *precision* as of feature set quality for those datasets. A plot of the new Fscore *vs.* Tscore (Figure 2-a) reveals that a significant correlation of 0.84 is attained (pvalue 2.10^{-19}),[12] when the scores are averaged over all datasets and test set versions.

To factor out the variability due to the choice of the classifier, we asked several participants to train their learning machine on all the feature sets submitted by the participants and we redrew the same graphs. The performances improved or degraded for some participants and some datasets, but on average, the correlation between Tscore and the new Fscore did not change significantly.[13] See the on-line results for details.

Pairwise comparisons. In an effort to remove the bias introduced by selecting the best Tscore for participants who returned multiple prediction results for nested subsets of features, we made pairwise comparisons of entries, using the same number of features. Specifically, if one entry used a single feature subset of size n and the other provided results for nested subsets, we selected for the second entry the Tscore corresponding to n by interpolating between Tscore values for the nested subsets. If both entries used nested feature subsets, we compared

12. This is the pvalue for the hypothesis of no correlation. We use confidence bounds that are based on an asymptotic normal distribution of $0.5\cdot\log((1+R)/(1-R))$, where R is the Pearson correlation coefficient, as provided by the Matlab statistics toolbox.

13. Computing these scores requires defining truth values for the set of "relevant features" and "irrelevant features". In our original Fscore, we used the Markov blanket of the test set distribution as set of "relevant features". For SIDO and CINA, there is only partial knowledge of the causal graph. The set of "relevant variables" is approximated by all true variables and the probes belonging to the Markov blanket (of the test set distribution). As an additional refinement, we experimented with three possible definitions of "relevant features": (1) the Markov blanket (MB), (2) MB + all causes and effects, and (3) all variables connected to the target through any directed or undirected path. If the test data are manipulated, those sets of variables are restricted to the variables not disconnected from the target as a result of manipulations. We ended up computing the new Fscore for each definition of "relevant features" and performing a weighed average with weights 3, 2, 1. We did not experiment with these weights, but the resulting score correlates better with Tscore than when the Markov blanket of the test distribution alone is used as reference "relevant" feature set. This is an indication that features, which are outside of the Markov blanket may be useful to make predictions (see Section 6 for a discussion).



(a) Regular ranking (b) Pairwise comparisons

Figure 2: **Correlation between feature selection score and prediction.** The new feature selection score Fscore (see text) evaluating the accuracy of causal discovery is plot as a function of the prediction accuracy score Tscore (the area under the ROC curve for test examples). The relative Tscore is defined as (Tscore − 0.5)/(Max Tscore − 0.5). Both Tscore and Fscore are averaged over all tasks and test set versions. (a) Ranking according to the rules of the challenge, selecting the best Tscore for nested feature subset results. (b) Ranking obtained with pairwise comparisons of classifiers, using the same number of features in each comparison.

them at the median feature subset size used by other entrants. If both entries used a single subset of features, we directly compared their Tscores. For each participant, we counted the fraction of times his Tscore was larger than that of others. We proceeded similarly with the Fscore. Figure 2-b shows the resulting plot. One notices that the performances of the winners by Tscore, Gavin Cawley and Yin-Wen Chang, regress in the pairwise comparison and that Jianxin Yin and Zhi Geng's group, Vladimir Nikulin, and Marc Boullé (M.B.), appear now to have better predictive accuracy. Jianxin Yin and Zhi Geng's group stand out on the Pareto front by achieving also best Fscore.

5.3 Methods employed

The methods employed by the top-ranking entrants can be categorized in three families:

- **Causal:** Methods employing causal discovery techniques to unravel cause-effect relationships in the neighborhood of the target.
- **Markov blanket:** Methods for extracting the Markov blanket, without attempting to unravel cause-effect relationships.
- **Feature selection:** Methods for selecting predictive features making no explicit attempt to uncover the Markov blanket or perform causal discovery.

In this section, we briefly describe prototypical examples of such methods taken among those employed by top-ranking participants.

Causal discovery: The top-ranking entrants who used causal modeling proceeded in the following way: they used a "constraint-based method" to establish a local causal graph in the neighborhood of the target, using conditional independence tests. They then extracted a feature subset from this neighborhood and used it to build a predictive model. The predictive models used belong to the family of regularized discriminant classifiers and include L1-penalized logistic regression, ridge regression, and Support Vector Machines (SVM). Descriptions of these methods are found *e.g.,* in (Hastie et al., 2000). The feature subsets extracted from the local causal graph differ according to the test set distribution. For unmanipulated test sets (sets numbered 0), the Markov blanket of the target is chosen, including only direct causes (parents), direct effects (children), and spouses. For test sets drawn from post-manipulation distributions (numbered 1 and 2), two cases arise: if the identity of the manipulated features is known to the participants, the feature subset selected is a restriction of the Markov blanket to parents (direct causes), unmanipulated children (direct effects), and parents of at least one unmanipulated child (spouses). This is the case for REGED1 and MARTI1. If the identify of the manipulated features is unknown to the participants, the feature subset selected is limited to direct causes. The techniques used to learn the local causal graph from training data are all derived from the work of Aliferis and Tsamardinos and their collaborators (Aliferis et al., 2003a; Tsamardinos and Aliferis, 2003; Aliferis et al., 2003b). Gavin Cawley (Cawley, 2008) used directly the "Causal explorer" package provided by the authors (Aliferis et al., 2003b). Laura E. Brown and Ioannis Tsamardinos (L.E.B. & Y.T.) (Brown and Tsamardinos, 2008) improved on their own algorithms by adding methods for overcoming several simplifying assumptions like "faithfulness" and "causal sufficiency". They proposed to address the problem of faithfulness by using a method for efficiently selecting products of features, which may be relevant to predicting the target, using non-linear SVMs, and proposed to address the problem of violations of causal sufficiency and hidden confounders by examining so-called "Y structures". Jianxin Yin and Zhi Geng's group (Yin et al., 2008) also introduced elements of novelty by proceeding in several steps: (1) removing features which are surely independent, (2) looking for parents, children, and descendants of the target and identify all V-structures in the neighborhood of the target, (3) orienting as many edges as possible, (4) selecting a suitable restriction of the Markov blanket (depending on the test set distribution, as explained above), (5) using L1-penalized logistic regression to assess the goodness of causal discover and eventually removing remaining redundant of useless features.

Markov blanket discovery: Discovering the Markov blanket is a by-product of causal discovery algorithms and can also sometimes be thought of as a sub-task. If known exactly, the Markov blanket is a sufficient set of features to obtain best prediction results if the test data are not manipulated. As explained in the previous paragraph, to remain optimal, this feature set must be restricted in the case of manipulated test data to parents (direct causes), unmanipulated children, and parents of unmanipulated children, or to only direct causes (depending on whether the manipulations are known or not). Hence, using the Markov blanket of the natural distribution for all test sets, including those drawn from post-manipulated distributions, is in principle sub-optimal. However, several participants adopted this strategy. One noteworthy contribution is that of Mehreen Saeed (Saeed, 2008), who proposed a new fast method to extract the Markov blanket using Dirichlet mixtures.

Feature selection: There have been a wide variety of feature selection methods, which have proved to work well in practice in past challenges (Guyon et al., 2006a). They do not have any theoretical justification of optimality for the causal discovery problem, except that in some cases it can be proved that they approximate the Markov blanket (Nilsson et al., 2007). Several participants used feature selection methods, disregarding the causal discovery problem, and obtained surprisingly good results. See our analysis in Section 6. The methods employed belong to the family of "filters", "wrappers" or "embedded methods". Vladimir Nikulin (Nikulin, 2008) used a "wrapper" approach, which can be combined with any learning machine, treated as a "black box". The method consists in sampling feature sets at random and evaluating them by cross-validation according their predictive power using any given learning machine. The

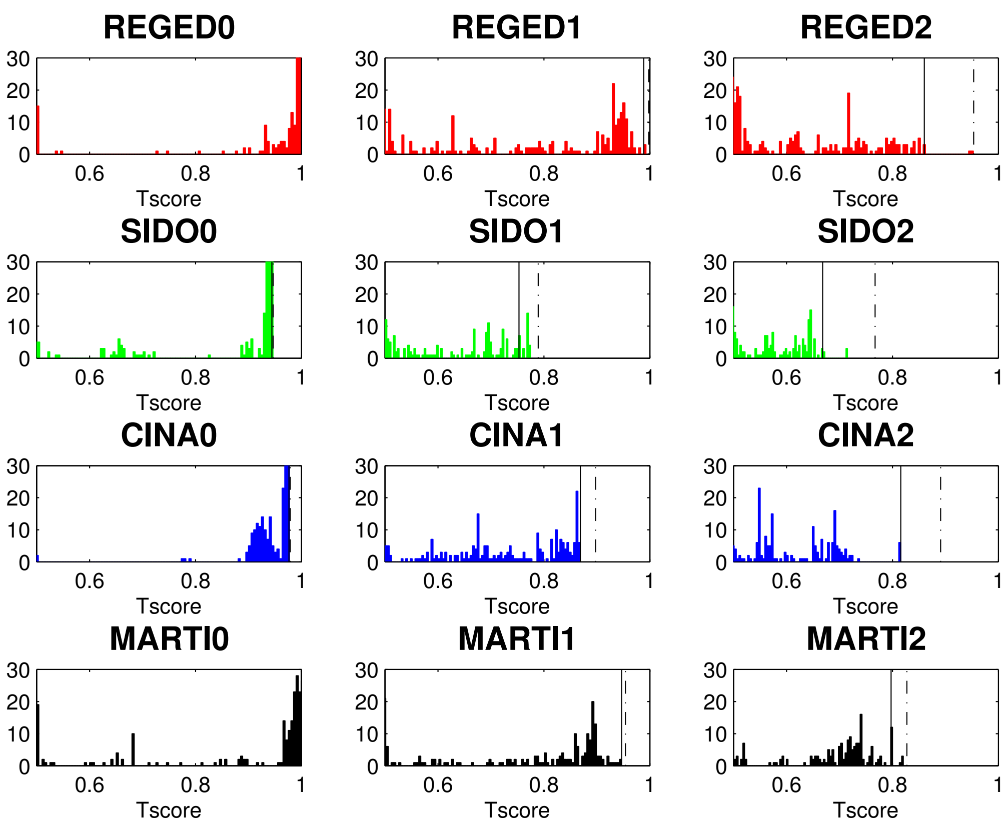


Figure 3: **Performance histograms.** We show histograms of Tscore for all entries made during the challenge. The vertical solid line indicates the best ranked entry (*i.e.,* best among the last complete entries of all participants). The dashed line indicates the overall best, including Reference entries, utilizing the knowledge of causal relationships not available to participants.

features appearing most often in the most predictive subsets are then retained. Yin-Wen Chang and Chih-Jen Lin (Chang and Lin, 2008), Gavin Cawley (Cawley, 2008), and Jianxin Yin and Zhi Geng's group (Yin et al., 2008) used embedded feature selection methods relying on the fact that, in regularized linear discriminant classifiers, the features corresponding to weights of small magnitude can be eliminated without performance degradation. Such methods are generalizable to non-linear kernel methods via the use of scaling factors. They include RFE-SVM (Guyon et al., 2002) and L1-penalized logistic or ridge regression (Tibshirani, 1994; Bi et al., 2003). Marc Boullé (M.B.) used a univariate filter method making assumptions of independence between variables.

5.4 Analysis by dataset

We show in Figure 3 histograms of the performances of the participants for all the entries made during the challenge. We also indicate on the graphs the positions of the best entry counting towards the final participant ranking; *i.e.,* their last complete entry, and the very best entry (among all entries including Reference entries made by the organizers.) As can be seen, the distributions are very different across tasks and test set types. In what follows, we discuss specific results.

For the test sets numbered 0, the best entries closely match the best Reference entries made by the organizers, who used knowledge of feature relevance not available to the competitors (such Reference entries used the Markov blanket of the target variable in the post-manipulation distribution as feature set and a SVM classifier). This is encouraging and shows the maturity of feature selection techniques, whether they are based or not on the extraction of the Markov blanket. For two datasets (REGED0 and CINA0), the univariate method of Marc Boullé (M.B.), which is based on the *naïve Bayes assumption* (independence between features) was best. This method had already shown its strength in previous challenges on the Adult database based on census data, from which CINA is derived. Interestingly, we know by construction of the REGED dataset that the naïve Bayes assumption does not hold, and yet good performance was obtained. This result is a nice illustration of the bias *vs.* variance tradeoff, which can lead biased models to yield superior prediction accuracy when training data are scarce or noisy. In this case, the multivariate methods of Yin-Wen Chang and Chih-Jen Lin (Chang and Lin, 2008) for REGED0 and Vladimir Nikulin (Nikulin, 2008) for CINA0 have results which are not significantly different from the univariate method of Marc Boullé.[14] For SIDO0, the best results achieved by Jianxin Yin and Zhi Geng's group (Yin et al., 2008) are not significantly different from the results of Gavin Cawley (Cawley, 2008), using *no feature selection*. Generally, regularized classifiers have proved to be insensitive to the presence of irrelevant features, and this results confirms observations made in past challenges (Guyon et al., 2006a,,). The best result for MARTI0 is also obtained by Gavin Cawley. His good performance can probably be partially attributed to the sophistication of his preprocessing, which allowed him to remove the correlated noise. In a post-challenge comparison he conducted between a Markov blanket-based feature selection and BLogReg, an embedded method of feature selection based on regularization, both methods performed well and the results were not statistically significantly different, and interestingly the BLogReg method yielded fewer features than the Markov blanket-based method.

For the test sets numbered 1 and 2, the distribution of test data differed from the training data. There is still on several datasets a large difference between the results of the best entrants and the best achievable result estimated by the organizers, using the knowledge of the true causal relationships. Sets 2 were more difficult than sets 1, for various reasons, having to do with the type of manipulations performed. Rather surprisingly, for test sets 1, non-causal methods yielded again very good results. Marc Boullé (M.B.) obtained the best performance on REGED1, with his univariate method making independence assumptions between features and involving no causal discovery. His feature set of 122/999 features does not contain variables which are not predictive (*i.e.,* not connected to the target in the causal graph of the post-manipulation distribution), but in general there are very few such variables. The best ranked competitor on REGED1, Marius Popescu, uses a particularly compact subset of 11 features, obtained with a causal discovery method combining HITON-MB (Aliferis et al., 2003a), some heuristics to orient edges, and the elimination of manipulated children and spouses whose children are all manipulated (see the Fact Sheet for details). According to pairwise comparisons, the next best result is obtained by the causal method of Laura E. Brown and Ioannis Tsamardinos (L.E.B. & Y.T.) (Brown and Tsamardinos, 2008) with only 9 features. Their feature set does not coincide exactly with the Markov blanket of the post-manipulation distribution (which includes 14 features), but it contains no irrelevant feature. For SIDO1, the best performance was obtained with all or nearly all features by Jianming Jin (Yin et al., 2008), Yin-Wen Chang

14. In the rest of this analysis, "not significantly different" means within one sigma, using our approximate error bar of Equation 1.

(Chang and Lin, 2008), and Gavin Cawley (Cawley, 2008). Hence, even when manipulations are performed, feature selection is so hard on this dataset that one is better off not doing any feature selection. The best performing causal discovery method on this dataset is that of Jianxin Yin and Zhi Geng's group (Yin et al., 2008), but their performance is significantly lower than that obtained with no feature selection (Tscore 0.70 instead of 0.75, with an error bar of 0.01). For CINA1 and MARTI1, Vladimir Nikulin (Nikulin, 2008) obtains the best performances with a feature selection method in pairwise comparisons (even though Gavin Cawley comes ahead in Table 2). He uses 30/132 and 400/1024 features, respectively. His fraction of irrelevant features in CINA1 is no better than the original proportion on the entire feature set. Jianxin Yin and Zhi Geng's group (Yin et al., 2008) are second best on those two datasets, with performances which are not statistically significantly different. Their causal discovery method yields fewer features (24/132 and 11/1024) and a smaller fraction of irrelevant features. The next best entries include both causal and non-causal methods. In conclusion, neither causal discovery methods nor feature selection methods seem to come ahead on test sets 1. This result will be further discussed in Section 6.

For test sets 2, making good predictions without causal modeling was expected to be significantly harder. Yet Jianxin Yin and Zhi Geng's group (Yin et al., 2008) are the only ones using causal discovery performing consistently well on sets 2. They are first or second in pairwise comparisons for REGED2, SIDO2, and MARTI2. For REGED2, Laura E. Brown & Ioannis Tsamardinos (L.E.B. & Y.T.) obtained the best performance with a causal discovery method. For SIDO2, E. Mwebaze and J. Quinn perform best according to pairwise comparisons, also using a causal discovery method. But, for MARTI2, none of the other top-ranking entries (in pairwise comparisons) include causal discovery methods, even though there is a very significant correlation between Fscore and Tscore (0.88). We will discuss the case of MARTI2 in more detail in Section 6. On CINA2 all causal discovery methods perform poorly, except that of Florin Popescu. However, since he submitted results only on CINA, it is difficult to say whether he was lucky or has a causal discovery method that is competitive on this problem. The other methods, which did well on CINA according to pairwise comparisons are those of Marc Boullé (M.B.) (naïve Bayes) and Vladimir Nikulin (feature selection with the random subset method). When selecting the best results in nested subsets of features, Yin-Wen Chang obtained significantly better results than anyone else with her SVM feature ranking. Her best feature set included only 4 features, which are all "real" variables. The case of CINA will be further discussed in Section 6.

6. Discussion

Several algorithms have demonstrated effectiveness of discovering causal relationships, as indicated by the Fscore, hence this challenge contributed to demonstrating that causal discovery from observational data is not an impossible task, albeit a very hard one. Yet the performance of causal models on tasks that were purposely designed to demonstrate their effectiveness is somewhat disappointing. It can be argued that causal discovery is a relatively new domain of research, which has not yet reached the maturity of some of the more mainstream machine learning techniques that were applied with success to the challenge. In particular, the use of causal discovery software made freely available may not be straightforward to use appropriately for people new to the field. However, it seems plausible that other factors are at play. In this section we analyze the results of the challenge in a critical manner and invite researchers to further investigate the open problems.

6.1 Correlation between causation and prediction in an interventional setting

One of our main motivations in organizing this challenge was to investigate the extent to which causal modeling is useful for making predictions in an "interventional setting" (a setting in which the test set is distributed differently from the training set as a result of the intervention of an external agent). Hence, in our analysis, we tried to quantify the correlation between "causation" (the accuracy of the causal modeling around the target variable) and "prediction" (the accuracy of the target variable predictions on test data). The former is captured by the Fscore and the latter by the Tscore. After modifying the Fscore, and averaging over all datasets and test set versions, we obtain a significant correlation between Fscore and Tscore (pvalue 2.10^{-19}). But, for individual tasks, there is a lot of variability. In past challenges (Guyon et al., 2006a,,), it was already observed that feature selection does not necessarily improve prediction accuracy when training and test data are drawn from the same distribution. This is due to the fact that state-of-the-art regularized classifiers such as SVMs, ridge regression, Random Forests (RF) and ensembles of neural networks, effectively overcome the curse of dimensionality without requiring a dimensionality reduction performed as preprocessing. In fact, feature selection is sometimes more harmful than useful in this case. For example, the best result on SIDO0 is obtained with no feature selection (in spite of the presence of irrelevant artificial variables or "probes"). More surprisingly, for test sets 1 and 2, although there is a significant correlation between Fscore and Tscore (on average over all tasks), we observe that feature selection methods based on causal discovery methods rarely outperforms feature selection methods ignoring causal relationships.

In a recent analysis paper (Tillman and Spirtes, 2008), the authors investigate the total contribution to prediction error made when non-causal methods use incorrect predictors for a manipulated distribution and when causal methods use incorrect or biased parametric constraints. They give theoretical conditions for manipulations where causal methods for prediction should have no advantage over non-causal methods and for manipulations where causal methods should produce considerably fewer errors. Briefly, the post-manipulation distribution $P(target|predictors)$ is identical to the natural distribution $P(target|predictors)$ only under special conditions, including that there is no manipulated direct effect of the target in the predictor set. The most difficult cases for non-causal methods arise (1) when all variables are manipulated or (2) when the non-manipulated variables (other than the target) are sampled before the effect of the manipulations have propagated.

Following this line of reasoning, we can partially explain why non causal methods performed so well by examining our challenge design. We sampled the variables after the effect of the manipulations propagated, because of the nature of our applications. Had we sampled them before the effect of the manipulations propagated, we would have made the task harder for non causal methods. Far fewer variables would have been predictive of the target, and, in particular, no consequence of the target would have been predictive. However, this limitation of our design was partially compensated by manipulating a large number of variables, including many direct effects of the target. Consequently, non causal methods incurred a larger false positive rate than causal methods for test sets 1 and 2, because many features relevant in the natural distribution were irrelevant in the post-manipulation distribution.

In the next section, we propose another explanation, which sheds light on the difficulty of improving performance with any kind of feature selection, causal or not.

6.2 Omitting good features may be more detrimental than including bad ones

We provide a qualitative explanation of why selecting a relatively large fraction of "irrelevant" features (including features relevant in training data and irrelevant in test data) might not penalize as much predictions as omitting key "relevant" features. The idea of our argument is that, in a predictive model, relevant variables tend to act in the same direction (to build the predictive signal) while, in the large sample limit, irrelevant variables contribute signals which average out to zero.

The results of our calculations provided in Appendix A are summarized in Table 3. We see that for test sets 0 the contribution of the irrelevant features can be very small compared to that of relevant features. To evaluate the number of irrelevant features one can "afford" for a given number of relevant features, we use the crossing point where the contribution of both type of features is equal. We obtain, for test sets 0, $n_b = m\, n_g^2$, for test sets 1, $n_b = n_g^2$, and for test sets 2, $n_b = n_g$. Plugging in some numbers, if there are of the order of $n_g = 30$ relevant features and $m = 500$ training examples, one can afford for test sets 0 of the order of $n_b = m \cdot n_g^2 = 500 \times 900 = 450{,}000$ irrelevant features. No wonder feature selection is not all that important in that case. For test sets 1, it is not that critical either to filter out irrelevant features, even if they are relevant in training data and manipulated in test data. Plugging in some numbers, if there are of the order of $n_g = 30$ relevant features, we can afford of the order of $n_b = n_g^2 = 900$ irrelevant features. This largely explains that feature selection is more needed on sets 1 than on sets 0, but simple feature selection does as well as causal feature selection. Finally, only in the worst case scenario of adversarial manipulations (test sets 2), can we only afford a number of "bad" features of the same order of magnitude of the number of "good" features.

The properties of irrelevant variables, on the basis of which we conclude that omitting relevant variable might more severely impair performance than including irrelevant variables, are obviously distribution dependent, and a case-per-case analysis would be needed to make a more quantitative assessment. We also need to caution against extrapolating our qualitative explanations and concluding that there is no benefit to performing causal discovery because of the relative insensitivity of certain regularized classifiers to the presence of irrelevant features (those who have parameters acting as feature scaling factors). These conclusions apply only to the particular tasks of the challenge and modifying the tasks may yield different conclusions. For example, the problem of finding which variables are the best targets of action to obtain a desired response requires a causal model. Also, we could have made things more difficult to non-causal models by sampling before the effect of the manipulations propagate to the non-target variables, thus making all consequences of the target variable non-predictive. However, this would not have affected the performances for test sets 2 in which all effect or all probes are manipulated.

Table 3: **Noise introduced by irrelevant features.** We computed for a simple univariate predictive model, the influence of relevant and irrelevant features. Both features and target are binary, and it is assumed that all relevant features correlate perfectly with the target and all irrelevant features are randomly drawn. With 98% confidence, the magnitude of the feature weights are lower than the value w quoted in the table and the total contribution $\sum_i w_i x_i$ is lower than the v quoted. n_g is the number of "good" (relevant) features and n_b is the number of "bad" (irrelevant) features, and m is the number of training examples.

Test set	Type	w relevant	w irrelevant	v relevant	v irrelevant
Set 0	unmanipulated	1	$1/\sqrt{m}$	n_g	$\sqrt{n_b/m}$
Set 1	manipulated	1	1	n_g	$\sqrt{n_b}$
Set 2	manipulated	1	1	n_g	n_b

6.3 Insignificant dependencies and spurious dependencies

We are left with explaining why for CINA2 and MARTI2 causal discovery methods do not perform as well as expected.

For CINA, we attribute the problem to the variable coding, which diluted information and led to many insignificant dependencies. By examining the features selected by regular feature selection algorithms and by causal discovery algorithms, we noticed that they were rather different. Feature selection algorithms select features that are individually very predictive, but not part of the Markov blanket. This may be due to the coding of categorical variables that we used: categorical variables taking c values were replaced by c binary variables, implementing a complete disjunctive code 10...0, 01...0, etc. So for instance, "number of years of education", which may be an ancestor variable of "profession", is individually more predictive than any of the individual professions: "clerical", "managerial", etc. Verifications of this explanation are under way by examining the causal graphs inferred by the top-ranking participants and the results will be published as part of the analysis of the second causality challenge (Guyon et al., 2008a).

For MARTI2, the correlated noise was considerably difficult for causal discovery algorithms, which did not perform well unless the noise was efficiently filtered. This is confirmed by the fact that causal discovery methods did well on REGED, a noiseless version of MARTI.

6.4 Is the Markov blanket truly optimal?

The above consideration on the relative insensitivity of predictive modeling to the presence of "irrelevant" variables may not alone explain the good performances of feature selection methods. It is possible that restricting the feature set to the Markov blanket of the test set distribution hampered performances. This strategy was adopted by all the participants performing causal discovery. If the causal paths to the target are not interrupted by the manipulations, adding some predictive non-MB variables (like the light green nodes in Figure 1) may help improving performances when a biased classifier is used, *e.g.,* if the MB is non-linearly related to the target and a linear classifier is used (Tsamardinos and Aliferis, 2003). Furthermore, the MB may include errors when estimated from a finite training set. We noted in the above calculations that it is far more detrimental to omit relevant features than include irrelevant features. Hence, subsets of larger size than the estimated MB are likely to give better predictions. The strategy adopted by the participants performing causal discovery was also sub-optimal in another respect: they selected a subset S of features that should be predictive of the target Y in the post-manipulation distribution, then they trained a "regular" learning machine to estimate directly $P(Y|S)$ with training data from the natural distribution. In some cases, this is *not* equivalent to estimating $P(Y|S)$ in the post-manipulation distribution by using a causal model. Cases of that sort arise when one manipulates children of the target, which have *unmanipulated* children of the target as descendants. For instance, in the LUCAS example of Figure 1, if we had manipulated the variable "coughing", but not the variable "fatigue", both "fatigue" and "coughing" would still be in the Markov Blanket of the post-manipulation distribution ("coughing" would now be a "spouse" of the target), but the direct connection between the target and "coughing" would be broken. Hence the contribution of "coughing" to $P(Y|S)$ would be over-estimated if $P(Y|S)$ was estimated by a statistical learning machine trained from the natural distribution, because this would include the direct effect of the target on "coughing". In contrast, a causal model taking into account the manipulations would factor out such direct effect when estimating $P(Y|S)$.

6.5 Lessons learned for future challenges

We end this discussion with some comments on the challenge protocol. First, as noted before, selecting the best Tscore in nested subsets of features introduced bias in the results and we do not recommend using this paradigm in future challenges of this kind. It is necessary to ask the participants to provide single predictions, or make pairwise comparisons of performance at equal number of features. Second, in our setup, the target variable was never manipulated. This makes sense for problems in which we are seeking to discover causes of a given outcome in order to influence it. For example, in epidemiology, we want to find risk factors of lung cancer such as smoking. But there are problems in which a target variable is manipulated and the goal is to monitor the effects of the manipulation. For example, the disappearance of symptoms can help monitoring the effect of a drug on a disease. Third, in our setup, we perform manipulations and wait before we sample data, until the effects of the manipulations have propagated through the system. In some cases, it makes more sense to sample data before the manipulations are performed and ask the question: what if we did these manipulations to given variables? Fourth, in our setup, we asked the participants to make predictions of a target variable under manipulations of other variables. Emphasis was placed on prediction rather than on variable selection. Another question would be to find those variables which should be manipulated to produce a given desired effect, *i.e.,* a given change in the target value. Finally, we posed a problem in which causal models had to be inferred solely from observational data. In many cases, it is costly but feasible to include manipulated data as part of training.

7. Conclusions

The first causality challenge we have organized allowed many researchers both from the causal discovery community and the machine learning community to try their algorithms on sizeable tasks of real practical interest. It achieved a number of goals that we had set: familiarizing many new researchers and practitioners with causal discovery problems and existing tools to address them, pointing out the limitations of current methods on some particular difficulties, and fostering the development of new algorithms.

The setting of the challenge purposely resembled a classical machine learning competition, with a training set and a test set, with omitted labels, to encourage the participation of data mining and machine learning researchers. The goal was to make optimum predictions on test data, as measured by a Tscore (the area under the ROC curve on test data). Each task had three test sets, with increasing levels of difficulty. The first one was identically distributed as the training set. The two other test sets simulated manipulations by external agents, and thus were not distributed like the training set. In this way we illustrated the relationships between causation and prediction under manipulations and investigated whether causal models using "causally relevant" features would perform better than regular statistical models on manipulated test sets. We proposed a simple score to evaluate the causal relevance of the subset of features selected, called Fscore. Several algorithms have demonstrated effectiveness of discovering causal relationships, as indicated by a large Fscore. On average over all datasets and tasks, the Fscore correlates significantly with the Tscore, confirming the link between causation and prediction. As anticipated, non-causal feature selection methods are doing well on the first type of datasets (training and test data identically distributed): the bulk of them is close to optimal, so if you chose one method at random, you would do well. However, for the other two types of datasets (test data manipulated) the distribution of results is about uniform: if you chose one method at random, you would probably do poorly. In addition, there is room for improvement to reach optimality. Thus, non-causal feature selection methods are inappropriate for these tasks, despite the fact that some of them are top ranked and causal feature selection methods are still not mature and robust enough to significantly outperform non-causal feature selection in the range of tasks of the competition.

The results indicate that informative causal prediction from observational data is possible, although it remains challenging. This points to the need for further research and benchmarks. This challenge investigated an important problem in causal modeling, but there remain many other causal modeling and discovery issues to be explored. Future work includes organizing challenges on a broader range of causal questions.

Acknowledgments

The organization of this challenge was a team effort to which many contributed. We are particularly indebted to Olivier Guyon (MisterP.net) who implemented the back-end of the web site. The front-end is derived from the design of Steve Gunn (University of Southampton), formerly used for the NIPS 2003 feature selection challenge. The kind support of Joachim Buhmann (ETH Zurich), who hosted the competition, is gratefully acknowledged. We are thankful to Thomas Fuchs (ETH Zurich) for administering the computer resources. The beta-testers, Gideon Dror (Academic College, Tel-Aviv-Yaffo, Israel), Amir Saffari (Graz University of Technology, Austria) and Marc Boullé (France Telecom, Lanion, France) performed indispensable work to ensure the platform was working properly, give us feed-back on the protocols and creating "Reference" entries. Alexander Borisov was first person to enter the challenge and he provided us with invaluable feed-back, which helped building the FAQ, which was an important contribution. We also are thankful to Gavin Cawley (University of East Anglia, UK), Joaquin Quiñonero Candale (Microsoft Research, UK), Richard Scheines (Carnegie Mellon University, Pennsylvania), and Lambert Schomaker (University of Groningen, The Netherlands) for helpful advice. We are very grateful to the institutions who originally provided the data: the DTP AIDS Antiviral Screen program of the National Cancer Institute (NCI); the census bureau and Ronny Kohavi and Barry Becker who extracted the data. Hans Bitter (Roche, Palo Alto, California) and Joerg Wichard (Institute of Molecular Pharmacology, Berlin, Germany) who provided features for SIDO are gratefully acknowledged. This project was supported by the Pascal network of excellence funded by the European Commission and by the U.S. National Science Foundation under Grant N0. ECCS-0725746. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Additional support was provided by the U.S. National Institute of Health under grant 2R56LM007948-04A1. The WCCI2008 conference, Health Discovery Corporation, Microsoft, and Unipen provided additional support for the monetary prizes granted to the winners. The comments of Ioannis Tsamardinos were very helpful to improve the manuscript, and we thank him for his effort.

References

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON, a novel Markov blanket algorithm for optimal variable selection. In *2003 American Medical Informatics Association (AMIA) Annual Symposium*, pages 21–25, 2003a.

C. F. Aliferis, I. Tsamardinos, A. Statnikov, and L.E. Brown. Causal explorer: A probabilistic network learning toolkit for biomedical discovery. In *2003 International Conference on*

Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS), Las Vegas, Nevada, USA, June 23-26 2003b. CSREA Press.

J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *JMLR*, 3:1229–1243, 2003.

M. Boullé. Compression-based averaging of selective naive bayes classifiers. *JMLR*, 8:1659–1685, July 2007a.

M. Boullé. Report on preliminary experiments with data grid models in the agnostic learning vs. prior knowledge challenge. In *IEEE/INNS conference IJCNN 2007*, Orlando, Florida, August 12-17 2007b.

L. E. Brown and I. Tsamardinos. A strategy for making predictions under manipulation. In *JMLR W&CP*, volume 3, pages 35–52, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

G. Cawley. Causal and non-causal feature selection for ridge regression. In *JMLR W&CP*, volume 3, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

Y.W. Chang and C.J. Lin. Feature ranking using linear svm. In *JMLR W&CP*, volume 3, pages 53–64, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov. Design and analysis of the causality pot-luck challenge. In *JMLR W&CP*, volume 5: NIPS 2008 causality workshop, to appear, Whistler, Canada, December 12 2008a.

I. Guyon, C. Aliferis, and A. Elisseeff. Causal feature selection. In Huan Liu and Hiroshi Motoda, editors, *Computational Methods of Feature Selection*, pages 63–82. Chapman and Hall/CRC Press. Longer TR: http://clopinet.com/isabelle/Papers/causalFS.pdf, 2007.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Editors. *Feature Extraction, Foundations and Applications*. Studies in Fuzziness and Soft Computing. With data, results and sample code for the NIPS 2003 feature selection challenge. Physica-Verlag, Springer, 2006a.

I. Guyon, A. Saffari, G. Dror, and J. Buhmann. Performance prediction challenge. In *IEEE/INNS conference IJCNN 2006*, Vancouver, Canada, July 16-21 2006b.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Analysis of the IJCNN 2007 agnostic learning vs. prior knowledge challenge. In *Neural Networks*, volume 21, pages 544–550, Orlando, Florida, March 2008b.

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

I. Guyon et al. Datasets of the causation and prediction challenge. Technical Report, 2008.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction*. Springer Verlag, 2000.

R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall series in Artificial Intelligence. Prentice Hall, 2003.

V. Nikulin. Random sets approach and its applications. In *JMLR W&CP*, volume 3, pages 65–76, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

R. Nilsson, J. M. Peña, J. Björkegren, and J. Tegnér. Consistent feature selection for pattern recognition in polynomial time. *J. Mach. Learn. Res.*, 8:589–612, 2007. ISSN 1533-7928.

Judea Pearl. *Causality: models, reasoning and inference*. Cambridge University Press, March 2000.

J. Quiñonero Candela, A. Schwaighofer, and N. Lawrence. Learning when test and training inputs have different distributions, http://different.kyb.tuebingen.mpg.de/pages/home.php 2007.

M. Saeed. Bernoulli mixture models for markov blanket filtering and classification. In *JMLR W&CP*, volume 3, pages 77–91, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, Cambridge, Massachusetts, London, England, 2000.

R. Tibshirani. Regression selection and shrinkage via the lasso. Technical report, Stanford University, Palo Alto, CA, June 1994.

R. E. Tillman and P. Spirtes. When causality matters for prediction: Investigating the practical tradeoffs. In *JMLR W&CP*, volume 5: NIPS 2008 causality workshop, to appear, Whistler, Canada, December 12 2008.

I. Tsamardinos and C.F. Aliferis. Towards principled feature selection: Relevance, filters, and wrappers. In *Ninth International Workshop on Artificial Intelligence and Statistics*, Florida, USA, January 2003.

J. Yin, Y. Zhou, C. Wang, P. He, C. Zheng, and Z. Geng. Partial orientation and local structural learning of causal networks for prediction. In *JMLR W&CP*, volume 3, pages 93–104, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

Appendix A. Influence of irrelevant variables

We made an argument that adding irrelevant variables to the predictive feature set might not be as detrimental as omitting good ones. We base our qualitative analysis on a simple model, assuming that all variables including the target are binary (taking values ±1) and that we use a linear predictive model

$$v = \sum_i w_i x_i,$$

in which the weights are trained with "Hebb's rule"

$$w_i = (1/m) \sum_k x_i^k y^k,$$

where the index k runs over all training examples, and m is the number of training examples. We further assume that the features are either perfectly relevant (identical to y or $-y$) or perfectly irrelevant (random). We wish to compute the relative contribution of relevant and irrelevant features to v in various cases to give insight into the number of irrelevant features, which can be afforded, relatively to the number of good features selected. In all cases, the magnitude (absolute value) of the weight of relevant features are:

$$w_{relevant} = 1.$$

Hence, the overall contribution of relevant features is the number of relevant or "good" features:

$$v_{relevant} = n_g.$$

For irrelevant features, we first examine the case where training and test data are identically distributed (case 0). If the irrelevant features are drawn randomly with equal probability $p = 0.5$, then the expected value of the magnitude of the weights of irrelevant features is 0. The standard deviation of the mean of $x_i^k y^k$ is $\sqrt{p(1-p)/m} = 0.5/\sqrt{m}$. To simplify our calculation, we use 98% confidence intervals, which roughly correspond to 2 sigma error bars by approximating the Binomial distribution with the Normal law. Hence, with 98% confidence, the magnitude of the weights of irrelevant features is less than

$$w^0_{irrelevant} = 1/\sqrt{m}.$$

We therefore verify that, for this model, the contribution of the irrelevant features vanishes to zero in the large sample limit. Similarly, the test set values of x_i are drawn randomly with equal probability $p = 0.5$. Hence, the total contribution has mean 0, and standard deviation bounded by $w\sqrt{n_b\ p(1-p)} = 0.5\ w\sqrt{n_b}$, where n_b is the number of irrelevant or "bad" features and w is our bound on the weight magnitude: $1/\sqrt{m}$. If we again choose a 98% confidence, we obtain a bound on the total contribution of the irrelevant variables of

$$v^0_{irrelevant} = \sqrt{n_b/m}.$$

In contrast, for test sets 1 and 2, in the worst case scenario, a feature perfectly relevant with respect to the training data distribution and perfectly irrelevant in the post-manipulated distribution will receive a weight of magnitude

$$w^1_{irrelevant} = w^2_{irrelevant} = 1.$$

In the scenario of test sets 1, values for such manipulated features are drawn randomly with equal probability $p = 0.5$. Following a calculation previously done, the standard deviation is bounded by $w\sqrt{n_b\ p(1-p)}$, but this time $w = 1$! The resulting bound on the total contribution of the "bad" features is

$$v^1_{irrelevant} = \sqrt{n_b},$$

with at least 98% confidence, because we assumed a worst-case scenario. For test sets 2, adversarial values may be given to the manipulated features, *i.e.,* opposite values than those expected from the training data distribution. So, in the worse case, the total contribution of the bad features is

$$v^2_{irrelevant} = n_b.$$

Appendix B. Verification of Challenge Results

The rules of the challenge prohibited the use of testing data for feature selection and building of the classifier model. However, all testing data with the exception of the response variable was available to the challenge participants. That is why we decided to verify several submissions from the challenge by studying and executing source codes of the participants on our computers. While doing this verification we paid close attention to ease of reproduction of the challenge results and involved computational resources. Such information will be very useful to practitioners who may decide to apply such algorithms to other datasets.

We have selected 6 challenge participants that provided us with software and code for verification: *Gavin Cawley*, *Yin-Wen Chang*, *J. Yin & Z. Geng Gr.*, *L.E.B & Y.T.*, *Vladimir Nikulin*, and *Mehreen Saeed*. The base verification dataset was selected to be REGED due to its empirical difficulty in the challenge and requirement for causal feature selection. However, *Vladimir Nikulin* provided source codes only for CINA dataset and *Mehreen Saeed* provided codes only for CINA and SIDO datasets. Thus, we decided to use CINA dataset for verification of these two participants. Out of six selected participants, three (*J. Yin & Z. Geng Gr.*, *L.E.B & Y.T.*, *Vladimir Nikulin*) used algorithms for selection of a single feature set and the remaining participants (*Gavin Cawley*, *Yin-Wen Chang*, *Mehreen Saeed*) used techniques to selected nested subsets of features.

The verification protocol consisted of two major steps: (i) manual reading of the source code to ensure that it does not employ testing data during feature selection and building of the classifier model and (ii) reproducing results of the challenge in a series of experiments. We considered the following experiments for versions 0, 1, and 2 of the datasets:

Experiment	*Description*
1	Exact reproduction of the challenge submission
2	Using reduced testing dataset with 500 samples (250 positives and 250 negatives, selected at random)
3	Using reduced testing dataset with 200 samples (100 positives and 100 negatives, selected at random)
4	Using reduced testing dataset with 500 samples, selected at random
5	Using reduced testing dataset with 200 samples, selected at random
6	Same as experiment 1 but with randomly permuted variables and samples
7	Same as experiment 2 but with randomly permuted variables and samples
8	Same as experiment 3 but with randomly permuted variables and samples
9	Same as experiment 4 but with randomly permuted variables and samples
10	Same as experiment 5 but with randomly permuted variables and samples

Experiment #1 was designed to verify that the automated code of a participant matched the challenge entry in terms of classification AUC. Experiments #2–5 were primarily used to confirm that the challenge participant did not use testing data to make inferences about the distribution. Experiment #6 was intended to illustrate that the code both does not rely on hard-coded feature indices and is not sensitive to the ordering of variables. Finally, experiments #7–10 were seeking goals of both experiments #2–5 and #6.

First of all, our manual reading of the source codes confirmed that none of the selected challenge participants cheated by using testing data for training of the classifier or feature selection.

Figure 4 and Table 4 report classification AUC's for the above described experiments. The results for versions 0 of the datasets are not reported in Figure 4 because they have near-perfect reproducibility. In summary, the results of the selected challenge participants reproduced in all experiments.

The code submitted by the team *L.E.B. & Y.T.* includes the automation of a step that during the competition was performed manually. The authors declared that the automated step is as close as possible to the subjective method used during the competition. An implementation of the strategy proposed by the authors is now fully automated and produces reproducible and repeatable results.

In all experiments we used Xeon 2.8 GHz CPU's with 4 Gb RAM. For the REGED dataset, the slowest algorithm was the one by *Gavin Cawley* with t_{train} =~20–30 hours and the fastest one was by *Yin-Wen Chang* with t_{train} =1–2 minutes. For all other methods in the REGED dataset, $t_{train} \in [15 \text{ minutes}, 2 \text{ hours}]$. For the CINA dataset, all methods have $t_{train} < 1$ hour. The testing time was negligible for all algorithms and datasets ($t_{test} < 2$ minutes). All algorithms had relatively efficient implementations. The only exception is the code by *Gavin Cawley* that required > 300 Mb for storage of the model for REGED datasets. Another inefficiency was observed in the code of *Yin-Wen Chang* that required ~ 4 Gb of RAM to apply a model to a testing set of 20,000 instances.

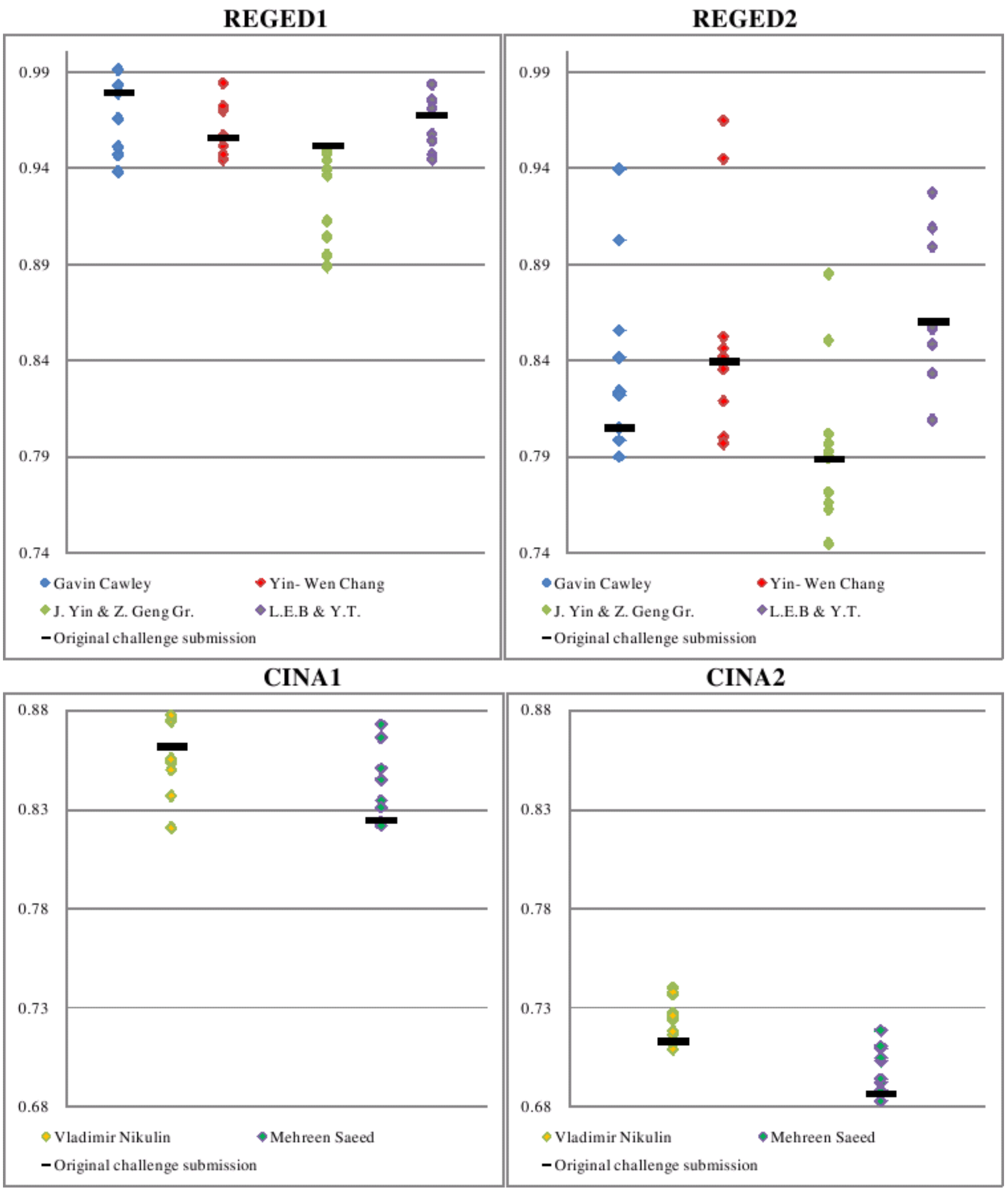


Figure 4: Testing set classification performance (measured by AUC) for 6 participants of the challenge. Each dot corresponds to results of an experiment.

Table 4: Testing set classification performance (measured by AUC) for 6 participants of the challenge.

	REGED0				**CINA0**	
Experiment	*Gavin Cawley*	*Yin-Wen Chang*	*J. Yin & Z. Geng Gr.*	*L.E.B & Y.T.*	*Vladimir Nikulin*	*Mehreen Saeed*
1	0.9997	0.9998	0.9998	0.9999	0.9770	0.9750
2	1.0000	0.9999	0.9998	0.9999	0.9812	0.9865
3	1.0000	1.0000	1.0000	1.0000	0.9512	0.9727
4	1.0000	0.9997	0.9997	0.9999	0.9752	0.9754
5	0.9983	0.9981	0.9997	0.9989	0.9805	0.9744
6	0.9998	0.9998	0.9998	0.9996	0.9760	0.9749
7	0.9998	1.0000	0.9999	0.9998	0.9741	0.9820
8	0.9997	0.9999	0.9996	0.9992	0.9622	0.9712
9	1.0000	1.0000	0.9999	0.9995	0.9709	0.9759
10	1.0000	1.0000	1.0000	1.0000	0.9682	0.9797
Challenge submission	**0.9997**	**0.9998**	**0.9997**	**0.9998**	**0.9764**	**0.9751**

	REGED1				**CINA1**	
Experiment	*Gavin Cawley*	*Yin-Wen Chang*	*J. Yin & Z. Geng Gr.*	*L.E.B & Y.T.*	*Vladimir Nikulin*	*Mehreen Saeed*
1	0.9787	0.9556	0.9442	0.9538	0.8549	0.8233
2	0.9789	0.9445	0.9392	0.9536	0.8532	0.8340
3	0.9825	0.9700	0.9490	0.9572	0.8742	0.8657
4	0.9907	0.9515	0.9478	0.9467	0.8366	0.8305
5	0.9905	0.9720	0.9362	0.9441	0.8206	0.8502
6	0.9469	0.9556	0.8943	0.9743	0.8542	0.8235
7	0.9463	0.9567	0.9125	0.9749	0.8552	0.8213
8	0.9506	0.9555	0.8888	0.9706	0.8746	0.8724
9	0.9378	0.9470	0.8947	0.9536	0.8497	0.8443
10	0.9653	0.9839	0.9042	0.9831	0.8769	0.8444
Challenge submission	**0.9787**	**0.9556**	**0.9517**	**0.9673**	**0.8617**	**0.8248**

	REGED2				**CINA2**	
Experiment	*Gavin Cawley*	*Yin-Wen Chang*	*J. Yin & Z. Geng Gr.*	*L.E.B & Y.T.*	*Vladimir Nikulin*	*Mehreen Saeed*
1	0.8045	0.8392	0.7926	0.8481	0.7159	0.6827
2	0.7984	0.8186	0.7626	0.8328	0.7401	0.7182
3	0.7897	0.8001	0.7660	0.8476	0.7367	0.7092
4	0.8218	0.7968	0.7448	0.8088	0.7238	0.7102
5	0.9025	0.9447	0.8850	0.9268	0.7088	0.6880
6	0.8237	0.8416	0.7896	0.8557	0.7180	0.6877
7	0.8218	0.8355	0.8019	0.8482	0.7271	0.7027
8	0.8413	0.8461	0.7968	0.8566	0.7258	0.6919
9	0.8555	0.8522	0.7715	0.8986	0.7398	0.7044
10	0.9395	0.9646	0.8503	0.9083	0.7377	0.6939
Challenge submission	**0.8045**	**0.8392**	**0.7885**	**0.8600**	**0.7132**	**0.6867**

A Strategy for Making Predictions Under Manipulation

Laura E. Brown [*] LAURA.E.BROWN@VANDERBILT.EDU
Department of Biomedical Informatics, Vanderbilt University, Nashville, TN 37232, USA
Ioannis Tsamardinos TSAMARD@ICS.FORTH.GR
Department of Computer Science, University of Crete and
BMI, ICS, Foundation for Research and Technology Hellas, Heraklion, Crete GR 700 13, GREECE
Department of Biomedical Informatics, Vanderbilt University, Nashville, TN 37232, USA

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

The first Causality Challenge competition 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. Our 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, we 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.

Keywords: Causal Bayesian Networks, Causal Discovery, Manipulations

1. Introduction

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 our analyses is that there exists a CBN that can represent the probability distribution of the data. This in turn implies that we assume 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 1(a).

[*]Conflict of Interest Disclosure: The authors of this paper are and have been affiliated with some of the organizers of the challenge. Consequently, our submission was not eligible for any prizes but, was allowed to participate in the challenge.

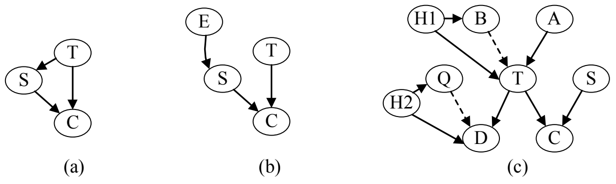


Figure 1: Causal Bayesian Networks: The unmanipulated CBN graph, $G_{\emptyset}$, and CBN graph $G_{\{S\}}$ where S is manipulated, are depicted in (a) and (b). In (c), a network with a hidden variables $H1$ causing both B and T, $H2$ causing both D and Q, and dashed edges (when the marginal over the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

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.

1.1.1 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 and 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, our approach for this task was to identify a Markov Blanket of T, $MB_{\emptyset}(T)$ then learn a predictive model using only these variables.

1.1.2 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 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 A):

Theorem 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 1(a), where the condition does not hold when S is manipulated, and the resulting network 1(b). Then, we have:

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

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

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

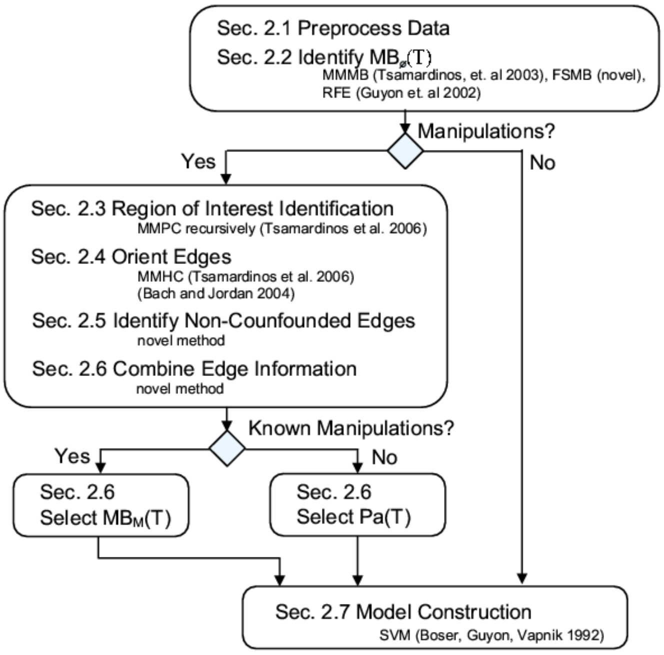

Sec. 2.1 Preprocess Data
Sec. 2.2 Identify MB∅(T)
MMMB (Tsamardinos, et. al 2003), FSMB (novel), RFE (Guyon et. al 2002)
Manipulations?
Yes
No
Sec. 2.3 Region of Interest Identification
MMPC recursively (Tsamardinos et al. 2006)
Sec. 2.4 Orient Edges
MMHC (Tsamardinos et al. 2006) (Bach and Jordan 2004)
Sec. 2.5 Identify Non-Counfounded Edges
novel method
Sec. 2.6 Combine Edge Information
novel method
Known Manipulations?
Yes
No
Sec. 2.6 Select MBM(T)
Sec. 2.6 Select Pa(T)
Sec. 2.7 Model Construction
SVM (Boser, Guyon, Vapnik 1992)

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

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 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 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 2.5). Finally, a predictive model using only the variables in the estimated $MB_{\mathbf{M}}(T)$ was learned.

1.1.3 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.

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 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 1.1.1). 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 1.1.2 and Section 1.1.3). 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 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.

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.

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 (Aliferis et al., 2003a) and MMMB (Tsamardinos et al., 2003a) 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 and Tsamardinos, 2008).

2.2.1 FEATURE SPACE MARKOV BLANKET (FSMB)

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., 2002) 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 $V_1 V_2$ 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 and Tsamardinos (2008).

2.2.2 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., 2003a).

To estimate how good of an approximation we obtained, we employed other feature selection algorithms and constructed models using all feature sets output (see Section 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., 2002, 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).

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 and Thrun, 1999) and MMPC, where MMPC returns the parents and children of T in a network $G_{\emptyset}$ (Tsamardinos et al., 2003a). 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 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 we have conducted however (publication under preparation), 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 2.2 requires v-structures among the parents of T, thus forcing us to induce a region of depth three.

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

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.

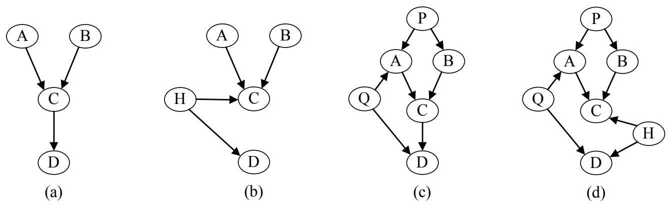


Figure 3: Four example networks to explain the Y-structure analysis.

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., 2006). For the case of continuous or mixed data, the kernel generalized variance scoring metric of Bach and Jordan (2002), with $\kappa = 0.01$ and $\sigma = 1$, was used with a greedy hill-climbing search to learn the structure. In Bach and 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.

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). 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 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 confounder of C and D, as shown in Figure 3(b). If Figure 3(b) was the case, A and D would be dependent given C and so their marginal would not faithful to Figure 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 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 2 *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 $C \to D$ 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 $A - D$.

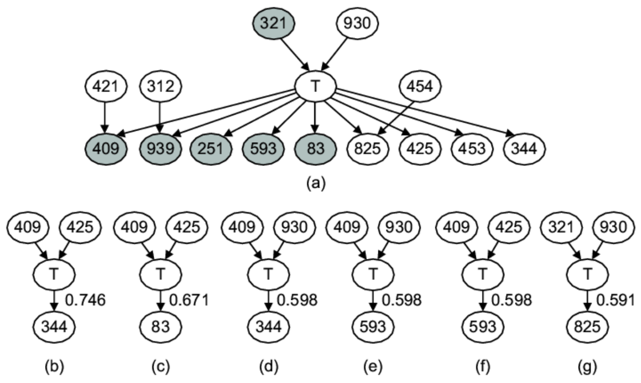
Multiple applications of the Y-test for different quadruple of variables may provide conflicting information for an edge $C \to D$. We devised two weighting schemes to rank the strength of evidence a single Y-test provides. First, a value was calculated as the minimum p-value returned by the independence 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.

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

We used the PDAG at the end of Section 2.4 to obtain the orientation of some edges and the method of Section 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 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 4 illustrates and summarizes the different information sources. Figure 4(a) shows the Markov Blanket variables extracted from the PDAG of Section 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 4(b)-(g) show the top six Y-structures centered on the target node ranked by the maximum p-score. Finally, the table in 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.

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 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.

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 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 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 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 equiva-

Table 1: Results on Challenge Data Sets: The Fnum, Fscore, Dscore, Tscore and Ranking is given for each version of the data sets; the results in (a) represent the final challenge submission and (b) show the results if the $MB_{\emptyset}(T)$ is used for every variable list regardless of considering manipulations. In (a), 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							Unmanipulated MB used for all Data Sets				
	Fnum	Fscore	Dscore	Tscore	Ranking			Fnum	Fscore	Dscore	Tscore	Ranking
CINA0	101	0.8496	0.9717	0.9721	9	Num. Entries/Total	7/277	101	0.8496	0.9717	0.9721	9
CINA1	5	0.4716	0.9316	0.5113	23	Average Tscore	0.6015	101	0.5795	0.9717	0.8581	4
CINA2	5	0.4716	0.9316	0.3210	25	Overall Ranking	23/25	101	0.5795	0.9717	0.6917	8
MARTI0	24	0.5869	0.9952	0.9681	8	Num. Entries/Total	2/233	24	0.5869	0.9948	0.9824	7
MARTI1	17	0.5643	0.9951	0.7837	9	Average Tscore	0.8083	24	0.5985	0.9948	0.8477	9
MARTI2	3	0.4985	0.6973	0.6730	10	Overall Ranking	9/19	24	0.7429	0.9948	0.6971	9
REGED0	15	0.8571	1.0000	0.9998	2	Num. Entries/Total	5/355	15	0.8571	1.0000	0.9998	2
REGED1	9	0.7851	1.0000	0.9673	4	Average Tscore	0.9423	15	0.7825	1.0000	0.9280	14
REGED2	3	1.0000	0.9728	0.8600	1	Overall Ranking	1/30	15	1.0000	1.0000	0.7231	9
SIDO0	13	0.5115	0.9356	0.9230	12	Num. Entries/Total	2/242	13	0.5015	0.9365	0.9237	12
SIDO1	4	0.5003	0.8587	0.6073	12	Average Tscore	0.6909	13	0.5012	0.9365	0.6626	11
SIDO2	4	0.5003	0.8587	0.5426	14	Overall Ranking	12/28	13	0.5012	0.9365	0.5713	11

(a) (b)

lent sample size in the Bayesian Score and the kernel parameters for the scoring metric of Bach and 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.

2.7 Building Predictive Models

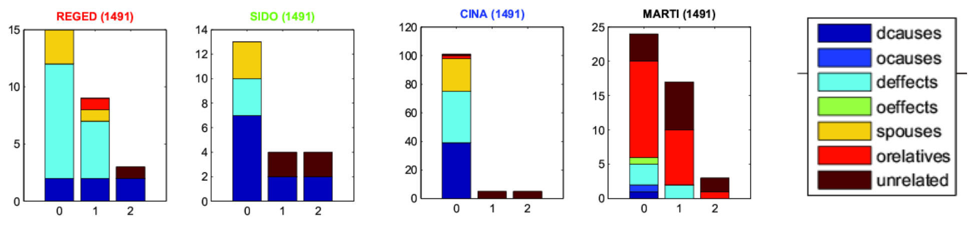
Once the variable list was determined for each data set, a final classification SVM model was trained on only the variable list members (Boser et al., 1992). An n-fold cross-validation design was used to select the optimal parameters: type of kernel (polynomial or Gaussian), kernel parameters (degree 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.

3. Results

The classification performance (AUC reported as Tscore in the challenge results) is ultimately how the challenge submissions were rated. Table 1(a) 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.

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



(a) (b) (c) (d)

Figure 5: The selected features' 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.

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 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 ~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 $MB_{\emptyset}(T)$ 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).

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.

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 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, Challenge Website 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 1(b) and can be contrasted with the challenge results in 1(a). 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 orientation 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.

4. Lessons Learned and Conclusions

The most important outcome of our participation to 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. We now distill some of these in the following.

Knowledge of the causal structure is theoretically necessary for making optimal predictions under manipulations. This is exemplified, in our opinion, in this challenge by the difference between the top non-causal submissions and the theoretical optimum performance; see the organizers' post-challenge analysis online (Challenge Website, 2008, Figures 3-6). Regarding the state-of-the-art in causal discovery, we believe there exists efficient, scalable, and publicly available code to learn the Markov Blanket. In fact, several other top participants also used our package Causal Explorer (Aliferis et al., 2003b) implementing such algorithms. These methods perform well on a range of high-dimensional data sets involving discrete, continuous, and mixed data. However, we also note that 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, we observe that the state-of-the-art methods employed to learn the orientation did not perform well. Consequently, we were unable to reliably identify the manipulated Markov Blanket.

Regarding important implementation issues, we 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, we presented a general strategy for predicting a quantity under manipulations of a system. 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 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.

References

C.F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON, A Novel Markov Blanket Algorithm for Optimal Variable Selection. In *Proceedings of the American Medical Informatics Association Conference(AMIA)*, pages 21–25, 2003a.

C.F. Aliferis, I. Tsamardinos, A. Statnikov, and L.E. Brown. Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS '03)*, pages 371–376, 2003b.

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

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

B. Boser, I. Guyon, and V. Vapnik. An Training Algorithm for Optimal Margin Classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

L.E. Brown and I. Tsamardinos. Markov Blanket-Based Variable Selection in Feature Space. Technical Report TR-08-XX, Vanderbilt Univeristy, 2008.

Challenge Website. Causation and prediction challenge. http://clopinet.com/isabelle/Projects/WCCI2008/Analysis.html, 2008.

L.D. Fu. A Comparison of State-of-the-Art Algorithms for Learning Bayesian Network Structure from Continuous Data. Master's thesis, Vanderbilt University, 2005.

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

S. Mani, P. Spirtes, and G.F. Cooper. A Theoretical Study of Y structures for Causal Discovery. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 314–323, 2006.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS-99)*, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmannn Publishers, San Mateo, CA, 1988.

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

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, 2nd edition, 2000.

Tetrad Project. http://www.phil.cmu.edu/projects/tetrad/.

I. Tsamardinos and C.F. Aliferis. Towards Principled Feature Selection: Relevancy, Filters and Wrappers. In *Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

I. Tsamardinos, C.F. Aliferis, and A. Statnikov. Time and Sample Efficeint Discovery of Markov Blankets and Direct Causal Relations. In *Proceedings of Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678, 2003a.

I. Tsamardinos, C.F. Aliferis, A. Statnikov, and L.E. Brown. Scaling-Up Bayesian Network Learning to Thousands of Variables Using Local Learning Techniques. Technical Report TR-03-02, Vanderbilt University, March 2003b.

I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65(1):31–78, 2006.

Appendix A.

Theorem 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 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. ■

Feature Ranking Using Linear SVM

Yin-Wen Chang B92059@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science, National Taiwan University
Taipei 106, Taiwan

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

Feature ranking is useful to gain knowledge of data and identify relevant features. This article explores the performance of combining linear support vector machines with various feature ranking methods, and reports the experiments conducted when participating the Causality Challenge. Experiments show that a feature ranking using weights from linear SVM models yields good performances, even when the training and testing data are not identically distributed. Checking the difference of Area Under Curve (AUC) with and without removing each feature also gives similar rankings. Our study indicates that linear SVMs with simple feature rankings are effective on data sets in the Causality Challenge.

Keywords: SVM, feature ranking.

1. Introduction

The Causality Challenge (Guyon et al., 2008) aims at investigating situations where the training and testing sets might have different distributions. The goal is to make predictions on manipulated testing sets, where some features are disconnected from their natural cause. Applications of the problem include predicting the effect of a new policy or predicting the effect of a new drug. In both examples, the experimental environment and the real environment differ.

In order to make good predictions on manipulated testing sets, we use several feature ranking methods to gain knowledge of the data. Among existing approaches to evaluate the relevance of each feature, some are related to certain classification methods, but some are more general. Those independent of classification methods are often based on statistic characteristics. For example, we experimented with Fisher-score, which is the correlation coefficient between one of the features and the label. In this work, we select Support Vector Machines (SVMs) (Boser et al., 1992) as the classifier, and consider one feature ranking method specific to SVM (Guyon et al., 2002).

This article is organized as follows. In Section 2 we introduce support vector classification. Section 3 describes several feature ranking strategies. Section 4 presents experiments conducted during the development period of the competition, our competition results, and some post-challenge analysis. Closing discussions are in Section 5.

2. Support Vector Classification

Support vector machines (SVMs) are useful for data classification. It finds a separating hyperplane with the maximal margin between two classes of data. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in R^n, y_i \in \{1, -1\}, i = 1, \ldots, l$, SVM solves the following unconstrained optimization problem:

$$\min_{\boldsymbol{w}, b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i), \tag{1}$$

where $\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$ is a loss function, and $C \geq 0$ is a penalty parameter on the training error. Two common loss functions are:

$$\max(1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b), 0) \text{ and } \max(1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b), 0)^2, \tag{2}$$

where ϕ is a function that mapped training data into higher dimensional space. The former is called L1-loss SVM, and the latter is L2-loss SVM. When participating in the challenge, we choose the L2-loss function. Post-challenge experiments show that the two loss functions result in similar performances. We give detailed results of using both loss functions in Section 4.3.

For any testing instance $\boldsymbol{x}$, the decision function (predictor) is

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b\right). \tag{3}$$

Practically, a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ may be used to train the SVM. A linear SVM has $\phi(\boldsymbol{x}) = \boldsymbol{x}$ so the kernel function is $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$. Another popular kernel is the radial basis function (RBF):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2), \text{ where } \gamma > 0. \tag{4}$$

We use linear SVM for both feature ranking and classification in the challenge. We also conduct some post-challenge experiments using SVM with RBF kernel as the classifier. The results will be discussed in Section 4.3.

We use grid search to determine the penalty parameter C for linear SVM, and both C and γ for SVM with RBF kernel. For each value of C or (C, γ), we conduct five-fold cross validation on the training set, and choose the parameters leading to the highest accuracy.

3. Feature Ranking Strategies

In this section, we describe several feature ranking strategies that we experiment with in the challenge. All methods assign a weight to each feature and rank the features accordingly.

3.1 F-score for Feature Ranking

F-score (Fisher score) is a simple and effective criterion to measure the discrimination between a feature and the label. Based on statistic characteristics, it is independent of the classifiers. Following Chen and Lin (2006), a variant of F-score is used. Given training instances $\boldsymbol{x}_i, i = 1, \ldots, l$, the F-score of the jth feature is defined as:

$$F(j) \equiv \frac{\left(\bar{\boldsymbol{x}}_j^{(+)} - \bar{\boldsymbol{x}}_j\right)^2 + \left(\bar{\boldsymbol{x}}_j^{(-)} - \bar{\boldsymbol{x}}_j\right)^2}{\frac{1}{n_+ - 1}\sum_{i=1}^{n_+}\left(x_{i,j}^{(+)} - \bar{\boldsymbol{x}}_j^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{i=1}^{n_-}\left(x_{i,j}^{(-)} - \bar{\boldsymbol{x}}_j^{(-)}\right)^2}, \tag{5}$$

```
Algorithm 1 Feature Ranking Based on Linear SVM Weights
Input: Training sets, $(\boldsymbol{x}_i, y_i), i = 1, \dots, l$.
Output: Sorted feature ranking list.
    1. Use grid search to find the best parameter $C$.
    2. Train a L2-loss linear SVM model using the best $C$.
    3. Sort the features according to the absolute values of weights in the model.
```

where n_+ and n_- are the number of positive and negative instances, respectively; $\bar{\boldsymbol{x}}_j$, $\bar{\boldsymbol{x}}_j^{(+)}$, $\bar{\boldsymbol{x}}_j^{(-)}$ are the average of the jth feature of the whole, positive-labeled, and negative-labeled data sets; $x_{i,j}^{(+)}/x_{i,j}^{(-)}$ is the jth feature of the ith positive/negative instance. The numerator denotes the inter-class variance, while the denominator is the sum of the variance within each class. A larger F-score indicates that the feature is more discriminative.

A known deficiency of F-score is that it considers each feature separately and therefore cannot reveal mutual information between features. However, F-score is simple and generally quite effective.

3.2 Linear SVM Weight for Feature Ranking

After obtaining a linear SVM model, $\boldsymbol{w} \in R^n$ in (1) can be used to decide the relevance of each feature (Guyon et al., 2002). The larger $|w_j|$ is, the jth feature plays a more important role in the decision function (3). Only $\boldsymbol{w}$ in linear SVM model has this indication, so this approach is restricted to linear SVM. We thus rank features according to $|w_j|$. The procedure is in Algorithm 1.

3.3 Change of AUC with/without Removing Each Feature

We determine the importance of each feature by considering how the performance is influenced without that feature. If removing a feature deteriorates the classification performance, the feature is considered important. We select the cross validation AUC as the performance measure. Features are ranked according to the AUC difference.

This performance-based method has the advantage of being applicable to all classifiers. The disadvantage is that it takes a huge amount of time to train and predict when the number of features is large. Besides, by removing only one feature at a time, the method does not take into account how features affect each other.

3.4 Change of Accuracy with/without Removing Each Feature

This method is the same as the one described in Section 3.3, except that the measure of performances is the accuracy rate.

4. Experimental Results

In the Causality Challenge, there are four competition tasks (REGED, CINA, SIDO and MARTI) and two small toy examples (LUCAS and LUCAP). All tasks have three versions of data sets, each with the same training set, and different testing sets. Testing sets with digit zero indicates unmanipulated testing set, while digit one and two denote manipulated testing sets. Table 1

Table 1: Challenge data sets. All of them have two classes.

Dataset	Feature type	# Feature	# Training	# Testing
REGED	numerical	999	500	20,000
SIDO	binary	4,932	12,678	10,000
CINA	mixed	132	16,033	10,000
MARTI	numerical	1,024	500	20,000
LUCAS	binary	11	2,000	10,000
LUCAP	binary	143	2,000	10,000

shows the data set descriptions. Details can be found at http://www.causality.inf.ethz.ch/challenge.php.

We preprocess data via scaling, instance-wise normalization, and Gaussian filtering. We scale each feature of REGED and CINA to $[0,1]$, and apply the same scaling parameter to their testing sets. In contrast, training and testing sets in MARTI are separately scaled to $[-1,1]$ for each feature, since this way results in a better performance. Another reason is that the training data in MARTI are perturbed by noises, while the testing data are free of noises. After applying a Gaussian filter on the training set to filter out the noises, there is an unknown bias value that we would like to substrate or add. We might use information from the distribution of testing data to gain knowledge of the unknown bias value, and then scale the training and testing data using the same scaling parameter. Alternatively, we can ignore the bias value, and scale the training and testing data separately. For SIDO, LUCAS, and LUCAP, the range of their features are already in $[0,1]$. We normalize each instance of these three problems to have the unit length.

According to the data set description, two kinds of noise are added to MARTI. First, to obtain 1,024 features, 999 features in REGED are complemented by 25 calibrant features, each of which has a value zero plus a small Gaussian noise. Second, the training set is perturbed by a zero-mean correlated noise. Since we cannot get into the first quartile of the competition results without regarding the noise, we use a Gaussian filter to eliminate the low frequency noise in the training set before scaling. For each instance, we rearrange the 1,024 features into a 32×32 array and apply the Gaussian filter, according to the fact that neighboring positions are similarly affected. The low pass spatial Gaussian filter is defined as:

$$g(x_0) = \frac{1}{G(x_0)} \sum_x e^{-\frac{1}{2}(\frac{\|x-x_0\|}{\sigma})^2} f(x), \text{ where } G(x_0) = \sum_x e^{-\frac{1}{2}(\frac{\|x-x_0\|}{\sigma})^2} \tag{6}$$

where $f(x)$ is the value at position x in the 32×32 array. For each position x, we take the Gaussian weighted average of all values in the array. The resulting $g(x)$ is the approximated low frequency noise we derive, and $f'(x) = f(x) - g(x)$ is the feature value that we would like to use. The σ is set to 3.2 after experimenting with several values.

Since testing sets may not follow the same distribution as training sets, and it is intended to hide their distributions, no validation sets are provided during the development period, which is the time between the start and the termination of the challenge. Instead, an on-line submission page shows which quartile that submission belongs to among all submissions. Besides, testing AUC of toy examples are available.

The linear SVM classifier that we use is LIBLINEAR[1] (Fan et al., 2008), and we use LIBSVM[2] (Chang and Lin, 2001) for SVM with RBF kernel. While LIBSVM can handle linear kernel as

1. http://www.csie.ntu.edu.tw/~cjlin/liblinear
2. http://www.csie.ntu.edu.tw/~cjlin/libsvm

Algorithm 2 Training and Prediction

```
Input: Training sets, testing sets.
Output: predictions on nested subsets.
  1. Use a feature ranking algorithm to compute the sorted feature list $f_j, j = 1, \dots, n$.
  2. For each feature size $m \in \{1, 2, 4, \dots, 2^i, \dots, n\}$.
     (a) Generate the new training set that has only the first $m$ features in the sorted feature
         list, $f_j, j = 1, \dots, m$.
     (b) Use grid search to find the best parameter $C$.
     (c) Train the L2-loss linear SVM model on the new training set.
     (d) Predict the testing set using the model.
```

well, we use LIBLINEAR due to its special design for linear SVM. Our implementation extends from the framework by Chen and Lin (2006)[3]. All sources for our experiments are available at `http://www.csie.ntu.edu.tw/~cjlin/papers/causality`.

We experiment with the feature ranking methods described in Section 3. We use F-score, W, D-AUC, D-ACC to denote the methods in Sections 3.1-3.4, respectively. The linear SVM weights are derived from LIBLINEAR model files. The procedure is described in Algorithm 2.

We summarize the methods that we experiment with:

- F-score: feature ranking using F-score described in Section 3.1.
- W: feature ranking using linear SVM weights described in Section 3.2.
- D-AUC: feature ranking by checking the change of AUC with/without removing each feature. Details are in Section 3.3.
- D-ACC: feature ranking by checking the change of accuracy with/without removing each feature. Details are in Section 3.4.

4.1 Development Period

During the development period, we took into account the cross validation AUC on training sets, the testing AUC of toy examples, and the quartile information to decide the method for the final submission.

Since we did not develop a strategy to deal with different training/testing distributions, we use the same model to predict each task's three testing sets. We did not use the provided information of the manipulated features in REGED and MARTI, and the 25 calibrant features in MARTI.

With AUC being the evaluation criterion, we submitted the decision values of linear SVM predictions. Nested subsets according to sorted feature lists are used since their performances are better. That is, one of the predictions based on a subset outperforms the one based on the whole feature set.

For F-score and W, it takes less than one minute to train all the models for nested-subset submissions for REGED, CINA, and MARTI, while it takes about 13 minutes for SIDO. Excluding preprocessing, the time required to predict one testing set is around five minutes for REGED

3. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`

Table 2: Best five-fold cross validation AUC and the corresponding feature size. The best feature ranking approach is bold-faced.

Dataset	REGED		SIDO[4]		CINA		MARTI	
F-score	0.9998	(16)	0.9461	(2,048)	0.9694	(132)	0.9210	(512)
W	**1.0000**	(32)	**0.9552**	(512)	**0.9710**	(64)	0.9632	(128)
D-AUC	**1.0000**	(16)	–		0.9699	(128)	**0.9640**	(256)
D-ACC	0.9998	(64)	–		0.9694	(132)	0.8993	(32)

Table 3: Comparisons of the performance on toy examples. The testing AUC is showed. Sorted feature list and nested subsets on it are used.

Dataset	LUCAS 0	LUCAS 1	LUCAS 2	LUCAP 0	LUCAP 1	LUCAP 2
F-score	0.9208	0.8989	0.7446	**0.9702**	0.8327	0.7453
W	0.9208	0.8989	**0.7654**	**0.9702**	**0.9130**	**0.9159**
D-AUC	0.9208	0.8989	**0.7654**	0.9696	0.8648	0.8655
D-ACC	0.9208	0.8989	0.7446	0.9696	0.7755	0.6011

and MARTI, 16 seconds for CINA, and three minutes for SIDO. SIDO is more computational costly to train and predict due to a larger number of features and training instances. For D-AUC and D-ACC, it takes a few hours to get the feature rank.

We submitted totally 60 entries before the challenge ended. Among methods we have tried, W has testing AUC in the first quartile for all data sets. This result seems to indicate that it is better than others. We used cross-validation with AUC in order to get more definitive conclusions.

Table 2 shows the five-fold cross validation AUC of using the best feature size. We conduct cross validation on all feature size $\in \{1,2,4,\ldots,2^i,\ldots,n\}$, where n is the total number of features. W and D-AUC seem to perform better than other methods, while D-ACC is the worst.

We find that D-ACC differs most from others, while the other three methods are more similar. Especially, the top ranked features chosen by W and D-AUC are alike. For example, W and D-AUC have exactly the same top four features for CINA, and the same set of top eight features with slightly different rankings for REGED.

In Table 3, we compare different feature ranking methods according to the testing AUC of the toy examples, LUCAS and LUCAP. We can see that W still outperforms others. It is much better than other methods especially on manipulated testing data sets (see LUCAP 1 and LUCAP 2). Similar to the cross validation results, D-ACC is the worst.

4.2 Competition Results

Table 4 shows the results of our final submission. Fnum is the best number of features to make prediction. It is determined by the organizers according to the nested-subset submissions. Fscore indicates how good the ranking is according to the causal relationships known only to the organizers. Tscore is the testing AUC. Top Ts is the maximal score of the last entry made by all participants, and Max Ts is the best score reachable, estimated using causal relationship knowledge not available to participants.

4. D-AUC and D-ACC are infeasible for SIDO due to the large number of features of SIDO.

Table 4: The results of our final submission in the Causality Challenge. We obtain feature ranking using linear SVM weights. The column "Fnum" shows the best feature size to make prediction and the total number of features.

Dataset	Fnum	Fscore	Tscore	Top Ts	Max Ts	Rank
REGED 0	16/999	0.8526	0.9998	1.0000	1.0000	
REGED 1	16/999	0.8566	0.9556	0.9980	0.9980	
REGED 2	8/999	0.9970	0.8392	0.8600	0.9543	
mean			0.9316			1
SIDO 0	1,024/4,932	0.6516	0.9432	0.9443	0.9467	
SIDO 1	4,096/4,932	0.5685	0.7523	0.7532	0.7893	
SIDO 2	2,048/4,932	0.5685	0.6235	0.6684	0.7674	
mean			0.7730			2
CINA 0	64/132	0.6000	0.9715	0.9788	0.9788	
CINA 1	64/132	0.7053	0.8446	0.8977	0.8977	
CINA 2	4/132	0.7053	0.8157	0.8157	0.8910	
mean			0.8773			1
MARTI 0	256/1,024	0.8073	0.9914	0.9996	0.9996	
MARTI 1	256/1,024	0.7279	0.9209	0.9470	0.9542	
MARTI 2	2/1,024	0.9897	0.7606	0.7975	0.8273	
mean			0.8910			3

We explain that on CINA 2, our method might benefit from good feature ranking. Our result is the best among all submissions. The four features used might be the direct cause of the label. As mentioned earlier, W and D-AUC identify exactly the same top four features. Similarly for MARTI 2 and REGED 2, Fnum is small and W and D-AUC select the same set of features, although the rankings are slightly different.

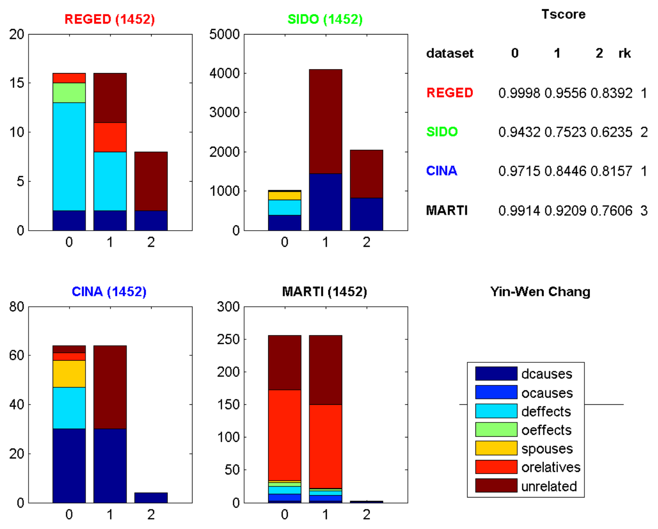
We also observe that the Fnums of the final submission are similar to the best feature size given by the cross validation results on the training data. However, we benefit from the nested-subset submission, since we do not select the best feature size. According to the rule, the best feature size is selected according to the testing AUC, so the testing set information is used indirectly.

Although the challenge is designed in a way that casual discovery is required to make good predictions, our simple feature ranking method performs rather well. It is interesting that our simple method outperforms some more complicated casual discovery methods.

However, the good performances do not indicate that the highly ranked features are important causes. Our methods rank the features according to their relevance, not their causal importance, and, thus, they do not enhance our knowledge of the underlying causal relationships between features.

Our linear SVM classifier has excellent performances on version 0 on all tasks. Our Tscore is close to Top Ts. However, compared with the best performance by other participants, the performance on version 1 is slightly worse, and the performance on version 2 is still worse than that on version 1. As the ranking for each task is determined according to the average of the performances on the three testing sets, we might take the advantage of good performances on version 0, where the testing and training distributions are the same.

Figure 1 shows the profile of the selected features (i.e., top Fnum features). This figure is provided by the organizers. The noise filtering method we used might not be good enough since



Tscore

dataset	0	1	2	rk
REGED	0.9998	0.9556	0.8392	1
SIDO	0.9432	0.7523	0.6235	2
CINA	0.9715	0.8446	0.8157	1
MARTI	0.9914	0.9209	0.7606	3

Figure 1: Profile of features selected (provided by the competition organizers). dcause: direct cause, deffect: direct effects, ocauses: other causes, oeffects: other effects, spouses: parent of a direct effect, orelatives: other relatives, unrelated: completely irrelevant.

for MARTI 0 and MARTI 1, the ratios of "direct causes" features are low compared with other methods. Besides, our feature ranking method ranks both direct causes and direct effects in the front of the list. They together make up most of the features on version 0. This result is reasonable since our methods do not consider causal relationships and therefore not necessarily rank true causes on the top. In Table 4, we have excellent performances on version 0 of all tasks. On manipulated testing sets, the ratio of unrelated features become higher, and our performance of these two versions are not as good as version 0. The only exception is CINA 2, where we did not obtain any unrelated features.

4.3 Post-Challenge Experiments

After the challenge, the testing AUC values of our past submissions are available. We are able to compare the results of all methods, including L2-loss linear SVM with different feature ranking methods and a direct use of SVM without feature ranking. We also conduct post-challenge experiments to compare the feature ranking methods using L1-loss SVM. Besides, in order to see if nonlinear SVMs help to improve the performance, we apply the feature rankings obtained from L2-loss linear SVM to nonlinear SVM with the RBF kernel.

Table 5 shows the testing AUC revealed after the challenge ended. LINEAR stands for a direct use of L2-loss linear SVM. It is worth noticing that similar to Tables 2 and 3, W is generally the best. This result is interesting as for testing AUC in Table 5, training and testing sets are from different distributions. An exception where F-score has better testing AUC than W is REGED. D-ACC is still the worst though the difference to other methods becomes much smaller.

In order to understand the difference between using L1-loss and L2-loss functions, we experiment with L1-loss linear SVM to rank features and classify data instances. The results are in Table 6. In general, the testing AUC values do not differ much from those of L2-loss SVM in

Table 5: Comparison of different feature ranking methods using L2-loss linear SVM. It shows testing AUC and the corresponding Fnum, revealed after the challenge has ended. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

	Feature ranking methods								SVM
Dataset	F-score		W		D-AUC		D-ACC		LINEAR
REGED 0	**0.9998**	(64)	**0.9998**	(16)	0.9997	(16)	0.9987	(128)	0.9970
REGED 1	0.9555	(32)	**0.9556**	(16)	0.9528	(16)	0.9438	(999)	0.9438
REGED 2	**0.8510**	(8)	0.8392	(8)	0.8392	(8)	0.8113	(32)	0.7442
mean	**0.9354**		0.9316		0.9306		0.9179		0.8950
SIDO 0	0.9430	(4096)	**0.9432**	(1024)					0.9426
SIDO 1	0.7515	(4932)	**0.7523**	(4096)					0.7515
SIDO 2	0.6184	(4096)	**0.6235**	(2048)					0.6143
mean	0.7710		**0.7730**						0.7695
CINA 0	0.9706	(132)	**0.9715**	(64)	0.9712	(128)	0.9706	(132)	0.9706
CINA 1	0.8355	(128)	**0.8446**	(64)	0.8416	(128)	0.8348	(132)	0.8348
CINA 2	0.6108	(64)	**0.8157**	(4)	**0.8157**	(4)	0.8140	(8)	0.6095
mean	0.8057		**0.8773**		0.8761		0.8732		0.8050
MARTI 0	0.9899	(512)	**0.9914**	(256)	0.9860	(1024)	0.9903	(512)	0.9860
MARTI 1	0.8960	(1024)	**0.9209**	(256)	0.9134	(32)	0.8960	(1024)	0.8960
MARTI 2	0.7571	(4)	**0.7606**	(2)	**0.7606**	(2)	0.7282	(1024)	0.7282
mean	0.8810		**0.8910**		0.8867		0.8715		0.8701

Table 5. However, here we do not have a solid conclusion that W outperforms other methods. Instead, most methods win on some data sets.

We applied the L1-loss SVM with RBF kernel on the list of features given by L2-loss linear SVM in order to clarify the performance in the case if feature rankings are combined with a nonlinear kernel. The results are shown in Table 7. Approach W still outperforms other methods when using a nonlinear SVM classifier. For these challenge data sets, W seems to be a good method regardless of the classifier used. Note that the Fnum values are not always the same in Tables 5 and 7, even though the same feature rankings are applied.

We also tried to incorporate Recursive Feature Elimination (RFE) (Guyon et al., 2002). For a given set of features, we use linear SVM weights to obtain the rankings, output ranks of those in the second half, and continue the same procedure on the first half features. To be more precise, subsets S_j of size $|S_j| \in \{n, 2^{\lfloor \log n \rfloor}, \ldots, 2^i, \ldots, 2, 1\}$ are generated, where n is the total number of features and $j = 0, \ldots, \lceil \log n \rceil$. After we train on subset S_j, we use the linear SVM weights to rank features in S_j and let S_{j+1} include the first half features. The results are not very different from that without RFE.

5. Discussion and Conclusions

In this challenge, we have experimented with several feature ranking methods. Among them, feature ranking based on F-score is independent from classifiers, feature ranking based on linear SVM weights require a linear SVM classifier, and the other two performance-based methods can use any classifier.

Table 6: Comparison of different feature ranking methods using L1-loss linear SVM. It shows testing AUC and the corresponding Fnum. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

	Feature ranking methods								SVM
Dataset	F-score		W		D-AUC		D-ACC		LINEAR
REGED 0	0.9996	(32)	**0.9997**	(16)	0.9991	(16)	0.9981	(256)	0.9964
REGED 1	0.9528	(32)	**0.9558**	(64)	0.9392	(256)	0.9551	(64)	0.9348
REGED 2	0.8562	(8)	0.8419	(8)	0.8504	(8)	**0.8777**	(16)	0.7396
mean	0.9362		0.9325		0.9296		**0.9436**		0.8903
SIDO 0	0.9407	(4096)	**0.9419**	(512)					0.9397
SIDO 1	0.7588	(4932)	**0.7590**	(4096)					0.7588
SIDO 2	0.6687	(4932)	**0.6701**	(2048)					0.6687
mean	0.7894		**0.7903**						0.7891
CINA 0	0.9713	(132)	0.9713	(132)	**0.9716**	(128)	0.9713	(132)	0.9713
CINA 1	0.8373	(128)	0.8369	(132)	**0.8425**	(128)	0.8369	(132)	0.8369
CINA 2	0.6377	(128)	0.6377	(128)	**0.8094**	(4)	0.6347	(132)	0.6347
mean	0.8154		0.8153		**0.8745**		0.8143		0.8143
MARTI 0	0.9872	(512)	0.9896	(256)	**0.9933**	(512)	0.9916	(512)	0.9858
MARTI 1	0.8950	(1024)	0.9046	(512)	**0.9168**	(512)	0.9078	(512)	0.8950
MARTI 2	0.7694	(8)	**0.7790**	(4)	0.7710	(2)	0.7369	(8)	0.7299
mean	0.8839		0.8911		**0.8937**		0.8787		0.8703

We focus on simple methods, so in this competition we can conduct quite complete validation procedures to select good models. However, although we have excellent performance on predictions, our methods do not provide information on the underlying causal relationships between features. Without causal discovery, the performance of our methods on manipulated data sets are not as good as that on unmanipulated data sets. Our methods might be improved by using causality, and how it can be done will need more investigations.

Acknowledgments

This work was supported in part by grants from the National Science Council of Taiwan.

References

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

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

Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors, *Feature extraction, foundations and applications*. Springer, 2006.

Table 7: Comparison of different feature ranking methods using L1-loss SVM with RBF kernel as classifier. It shows testing AUC and the corresponding Fnum. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

	Feature ranking methods								SVM
Dataset	F-score		W		D-AUC		D-ACC		RBF
REGED 0	**0.9997**	(64)	0.9995	(16)	**0.9997**	(16)	0.9989	(64)	0.9968
REGED 1	0.9709	(32)	**0.9753**	(32)	0.9748	(16)	0.9531	(128)	0.9419
REGED 2	**0.8881**	(8)	0.8676	(8)	0.8676	(8)	0.8189	(32)	0.7459
mean	**0.9529**		0.9475		0.9474		0.9236		0.8949
SIDO 0	0.9339	(4096)	**0.9444**	(4096)					0.9259
SIDO 1	0.7339	(4096)	**0.7634**	(4096)					0.7124
SIDO 2	0.5862	(4096)	**0.6255**	(4096)					0.5686
mean	0.7513		**0.7778**						0.7357
CINA 0	0.9732	(64)	**0.9754**	(32)	0.9716	(32)	0.9718	(128)	0.9683
CINA 1	0.8387	(64)	**0.8646**	(32)	0.8306	(4)	0.8383	(128)	0.8249
CINA 2	0.6855	(64)	**0.8358**	(4)	**0.8358**	(4)	0.8164	(8)	0.6739
mean	0.8325		**0.8919**		0.8793		0.8755		0.8224
MARTI 0	0.9883	(512)	**0.9916**	(256)	0.9848	(1024)	0.9896	(512)	0.9848
MARTI 1	0.8877	(1024)	**0.9181**	(256)	0.9057	(32)	0.8877	(1024)	0.8877
MARTI 2	**0.7659**	(8)	0.7616	(16)	0.7609	(2)	0.7308	(1024)	0.7308
mean	0.8806		**0.8904**		0.8838		0.8694		0.8678

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Isabelle Guyon, Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov. Design and analysis of the causation and prediction challenge. *JMLR: Workshop and Conference Proceedings*, 2008.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

Random Sets Approach and its Applications

Vladimir Nikulin V.NIKULIN@UQ.EDU.AU
Suncorp, Actuary Department
Brisbane, QLD, Australia

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

The random sets approach is heuristic in nature and has been inspired by the growing speed of computations. For example, we can consider a large number of classifiers where any single classifier is based on a relatively small subset of randomly selected features or random sets of features. Using cross-validation we can rank all random sets according to the selected criterion, and use this ranking for further feature selection. Another application of random sets was motivated by the huge imbalanced data, which represent significant problem because the corresponding classifier has a tendency to ignore patterns with smaller representation in the training set. Again, we propose to consider a large number of balanced training subsets where representatives from both patterns are selected randomly. The above models demonstrated competitive results in two data mining competitions.

Keywords: causal relations, random forest, boosting, SVM, CLOP, cross validation

1. Introduction

It is a well known fact that for various reasons it may not be possible to theoretically analyze a particular algorithm or to compute its performance in contrast to another. The results of the proper experimental evaluation are very important as these may provide the evidence that a method outperforms alternative approaches.

Feature selection (FS) represents a very essential component of data mining, as it will help to reduce overfitting and make prediction more accurate (see, for example, Nikulin (2006)). According to (Guyon et al., 2007) causal discovery may be regarded as a next step with the aim of uncovering causal relations between features and target variable. In many cases it is theoretically impossible to solve full graphical structure of all relations between features and target variable but it may be possible to uncover and approximate some essential relations. This knowledge will help to understand data better and will give some hints which methods will be more efficient.

A graphical model is a family of probability distributions defined in terms of a directed or undirected graph (Jordan, 2004). The nodes in the graph are identified with random variables, and joint probability distributions are defined by taking products over functions defined on connected subsets of nodes. By exploiting the graph-theoretic representation, the formalism provides general algorithms for computing conditional probabilities of interest.

In line with Bayesian Networks graphical semantics (Tsamardinos et al., 2004) every edge from a feature x_1 to a feature $x_2, x_1 \to x_2$, means that x_1 probabilistically and directly causes

x_2, see Figure 1. Bayesian Networks represent the joint probability distribution. For the given target variable y, the set of parents, children and spouses (i.e. parents of common children) is called the Markov Blanket (MB) of y. Markov Blanket as a set of features is sufficient in relation to y. Any other features becomes superfluous. Ideally, we would be interested to find MB and investigate its structure (see Section 2.1 for more details).

Usually, any dataset may be viewed as a matrix with two dimensions: 1) data entries and 2) features. In the Section 3.3 we consider application of the random sets (RS) approach to data-entries.

2. Methods

Let $\mathbf{X} = (\mathbf{x}_t, y_t), t = 1..n$, be a training sample of observations where $\mathbf{x}_t \in \mathbb{R}^\ell$ is ℓ-dimensional vector of features, and y_t is binary label: $y_t \in \{-1, 1\}$. Boldface letters denote vector, whose components are labeled using a normal typeface.

In practical situation the label y_t may be hidden, and the task is to estimate it using vector of features. Area under receiver operating curve (AUC) will be used as an evaluation and optimisation criterion.

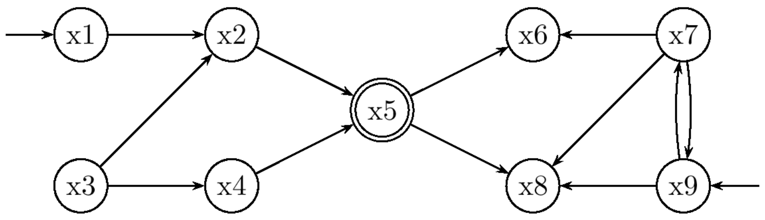


Figure 1: Illustrative Example.

According to Holland (1986) we shall assume that there are two causes of treatment, denoted by e (the experiment) and c (the control, which may be automatic or natural).

Fundamental problem of causal inference: it is impossible to observe the values of $y_t(e)$ and $y_t(c)$ on the same unit (that meant at the same time and for the same client) and, therefore, it is impossible to observe the effect of e on y.

For example, price of premium in insurance industry represents one of the most important features. Based on this price and available alternatives the customer will make a decision whether or not to renew an insurance contract. Suppose that the customer decided to accept renewal. In this case the Company would be interested to know decision if the price will be slightly higher. In an alternative case, if the customer decided to decline the proposed contract, the Company would be interested to know decision if the price will be slightly lower.

Classical randomized designs (Rubin, 1978) stand out as especially appealing assignment mechanisms designed to make inference for causal effects.

Let us consider another example where e represents a novel year-long study of arithmetics, c represents a standard arithmetic program, and target variable y is a score on a test at the end of the year. Obviously, for any particular student we can observe $y_t(e)$ or $y_t(c)$ but not both. Respectively, it appears to be natural to split randomly available field of students into several groups where we can apply either e or c. Absolutely similarly we can formulate example with medical applications.

During the Causal Discovery competition participants were able to take into account some specific information and assumptions. By the given definitions[1], there are two types of data: purely artificial and semi-artificial. In the latter case, there are two types of features: 1) real features and 2) probes where the last ones were artificially created variables as a functions of real features and other probes. Probes may be manipulated in order to highlight the importance of the proper feature selection. The real features and the target variable are never manipulated. As a result, in semi-artificial systems, only non-causes of the target may be manipulated.

Definition 1 *Manipulations are actions or experiments performed by an external agent on a system, whose effect disrupts the natural functioning of the system.*

Consider the example of Figure 1 where x_5 is a target variable, which graphically represents a presumed semi-artificial system. Features $x_1 - x_4$ (to the left side from x_5) cannot be manipulated as they cause (directly or indirectly) x_5. On the other hand, all features $x_6 - x_9$ (to the right side from x_5) may be manipulated because they may be viewed as consequences of the target variable. Let us consider two particular examples. Firstly, suppose that x_7 is a probe. Then, x_6, x_8 and x_9 must have the same status of probes. Secondly, suppose that x_6 is a probe. In this case, $x_7 - x_9$ may be probes or real features.

In the example of the Figure 1 MB consists of 6 members: 1) parents (x_2 and x_4); 2) children (x_6 and x_8); 3) spouses (x_7 and x_9). We know that direct causal features (parents) cannot be manipulated. Respectively, it will be the most disappointing to loose these features as a result of the filtering process.

The main assumption: we assume that direct causal features (parents) have stronger influence on the target variable and, therefore, are more likely to be selected by the Algorithms 1 and 2.

Algorithm 1: Basic Iterative Feature Selection (BIFS)

```
1: Input: training sample $\mathbf{X}$ including set of all features $S$.
2: Select loss function (or evaluation criterion) $D$, algorithm $g$ for the prediction and
   forward threshold parameter $\Delta_F$.
3: Set Z = $\emptyset$.
4: Select feature $f \in S$, which optimizes criterion $D$ applied to the prediction $g(f \cup Z)$.
5: Transfer feature $f$ from $S$ to Z in the case if improvement is sufficient (not smaller
   than $\Delta_F$).
6: Stop the algorithm if there are no sufficient improvement, or no features left in $S$.
   Alternatively, goto step 4.
```

Remark 2 *Essentially, BIFS-process is not uniform: initially, overfitting is limited because size of the set Z is small, and we can apply very simple algorithm (like linear regression). Then, the size of Z will grow and we will be able to use more advanced technique (for example, SVM). But, overfitting will grow at the same time and application of the cross validation (CV) may become unavoidable.*

Remark 3 *Note that we can add additional step (after step 5, Algorithm 1) with trimming (see Algorithm 3) or with test for independence within subset Z, see definition of HITON (Aliferis et al., 2003). On the one hand, this test will require additional computational time, but, on the*

1. http://www.causality.inf.ethz.ch/challenge.php

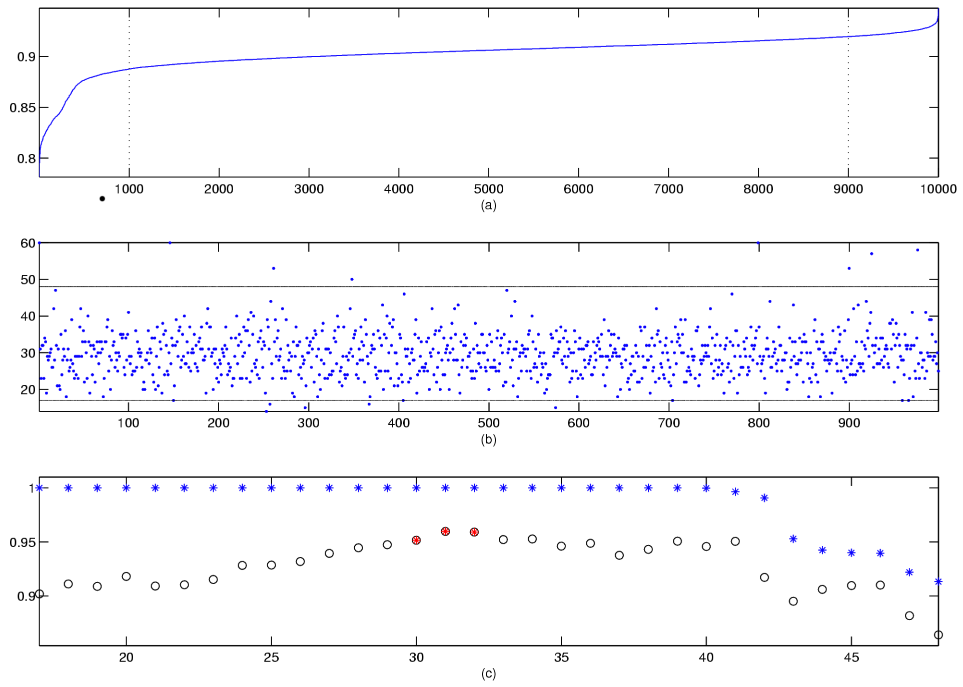


Figure 2: Illustration for RS-Algorithm 2 in the case of MARTI-set (see for more details Section 3). First, we evaluated 10000 random sets using AUC, (a) illustrates results sorted in an increasing order; (b) illustrates number of the occurrences for 999 features in the block B of 10% top performing random sets; (c) illustrates results of the secondary CV where features were selected according to the numbers of occurrences in the block B, stars correspond training results, circles correspond test results.

Algorithm 2: Random Sets (RS)

```
1: Evaluate long sequence of random subsets of features using CV.
2: Sort results in an increasing order (see Figure 2(a)).
3: Select block $B$ of the best (or worst in the case of deductive strategy, see Remark 5)
   performing sets of features.
4: Compute for any feature number of occurrences in the block $B$ (see Figure 2(b)).
5: Select range of occurrences for detailed investigation, which may be conducted using
   secondary CV (see Figure 2(c)).
```

other hand, it may be viewed as barrier to prevent growth of Z. As a consequence, the whole procedure may quickly reach state of equilibrium (subject to the proper selection of forward and backward threshold parameters). Respectively, it will be stopped.

Remark 4 *Note that we can facilitate Algorithm 1 using projections method as it is described in (Stoppiglia et al., 2003).*

Remark 5 *By definition, random set τ represents a relatively small subset of features. In the case if classifier requires bigger than 50% of all features it will be better to apply deductive strategy. That means, we will form a new subset of features $\gamma = S \setminus \tau$, which will be used in the*

Step 1 of the algorithm 2. Accordingly, we can classify subset τ as the worst if subset γ was classified as the best.

The functioning of the proposed Algorithm 2 is uniform, in difference to the Algorithm 1. As a consequence, we can apply any base algorithm, which appears to be appropriate for the given data (see for more details Figure 2 and Section 3).

Algorithm 3: Trimming

```
1: Input: training sample $\mathbf{X}$ including set of all features $S$.
2: Select loss function (or evaluation criterion) $D$, algorithm $g$ for the prediction, block
   of features $Z \subset S$ and backward threshold parameter $\Delta_B$.
3: Compute $\alpha = D(g(Z))$ - initial optimal value of the target function.
4: Select feature $f \in Z$, which optimises criterion $D$ applied to the prediction $g(Z \setminus f)$.
5: Compute $\beta = D(g(Z \setminus f))$ - new optimal value of the target function;
6: $Z := Z \setminus f$ if $|\alpha - \beta| < \Delta_B$, $\alpha = \beta$, and goto Step 4 if $Z \neq \emptyset$;
7: stop the algorithm if $Z = \emptyset$ or $|\alpha - \beta| \geq \Delta_B$.
```

Algorithm 3 may be used independently or in conjunction with Algorithms 1 or 2. The role of threshold parameters Δ_F and Δ_B is important and similar to the role of regularisation. Essentially, the online combination of the Algorithms 1 and 3 represents a modification of the Iterative Associative Markov Blanket (IAMB) algorithm (Aliferis et al., 2002).

2.1 Bayesian framework for the Markov blanket construction

Binary data-sets represent an ideal case for the illustration of the concepts of the Bayesian approach. Suppose that events $x_6 = 1$ and $y = 1$ represent coughing and lung cancer (see Figure 1). Using available data we can calculate two empirical probabilities:

$$1)\mathbb{P}(y = 1, x_6 = 1 | y = 1); \quad 2)\mathbb{P}(y = 1, x_6 = 1 | x_6 = 1).$$

We can expect that the first probability will be significant in difference to the second probability. As a next step, we can check stability of the values using standard bootstrapping technique. Based on the results of our analysis, we can make conclusion that there is a relation between y and x_6 where y is a parent (lung cancer) and x_6 is a child (coughing). However, there may be some complications. For example, x_6 may be a child of a child. The target of the Algorithm 3 is to detect and to resolve such problems. Similarly, we can investigate relations of the target variable with all other features. As an outcome we will obtain subset of features which have direct relations with target variable either as children or as parents where the last ones are the most important. Finally, we can detect field of spouses considering any particular child as a target variable.

Similarly, we can consider any discreet features. Note that consideration of continuous (numerical) features may be much more difficult. In this case we can apply transformation with several splitters for any particular feature. It works similarly to the method of classification trees.

3. Experiments

The list of 6 datasets which were used during WCCI-2008 Causal Discovery competition is given in the Table 1.

Table 1: List of datasets including sizes and main methods plus software which were used during the competition.

Data	# Train (positive)	# Test	ℓ	Method	Software
LUCAS	2000 (1443)	10000	11	neural+gentleboost	MATLAB-CLOP
LUCAP	2000 (1443)	10000	143	neural+gentleboost	MATLAB-CLOP
REGED	500 (59)	20000	999	SVM-RBF	C
SIDO	12678 (452)	10000	4932	binaryRF	C
CINA	16033 (3939)	10000	132	adaBoost	R
MARTI	500 (59)	20000	1024	svc+standardize	MATLAB-CLOP

The case of *MARTI*-set appears to be the most complicated because of the 25 given calibrants: the training set was perturbed by a zero-mean correlated noise model. As far as the test sets have no added noise, we used linear regression model in order to filter noise from the training set. Then, we considered sequence of 10000 sets with 40 randomly selected features (without repeats). Based on some preliminary experiments, we applied *svc* function from MATLAB-CLOP (deductive strategy: means, we used all features without features from random set, see Remark 5) for the evaluation. We sorted all sets in an increasing order (see Figure 2(a)) according to the meanTestAUC (used CV with 20 folds), and computed number of occurrences for any particular feature according to the block B of the worst 10% sets (see Figure 2(b)). Based on the visual consideration, we conducted detailed examination of the subinterval [17..48]. In this experiment features were selected according to the condition: $n_j \geq a, a \in [17..48]$ where n_j is number of repeats in the block B for the feature j.

Table 2: Results of the final submissions in terms of AUC (first 4 lines). LUCAS and LUCAP were used for validation and learning only.

Data	Submission	CASE0	CASE1	CASE2	Mean	Rank
REGED	vn14	0.9989	0.9522	0.7772	0.9094	4
SIDO	vn14	0.9429	0.7192	0.6143	0.7588	6
CINA	vn14a	0.9764	0.8617	0.7132	0.8504	2
MARTI	vn14	0.9889	0.8953	0.7364	0.8736	4
LUCAS	vn1	0.9209	0.9097	0.7958	0.8755	validation
LUCAP	vn10b+vn1	0.9755	0.9167	0.9212	0.9378	validation
CINA	vn1	0.9765	0.8564	0.7253	0.8528	all features
CINA	vn11	0.9778	0.8637	0.718	0.8532	CE

Figure 2(c) illustrates the final CV experiment where blue-stars correspond to the training and black-circle to the test results. Some marginal numerical values: 17) 994, 0.9019; 31) 410, 0.9597; 48) 8, 0.8641 where first and second numbers indicate number of the selected features and meanTestAUC. We can see some decline after point $a = 31$ (as a consequence of overfitting). Accordingly, the cases of $a \in [30..32]$ may be suitable for the submission in the normal situation when training and test samples have the same probability distribution.

It is interesting to note that in the initial submission *"vn1"* for CINA-set we used all 132 features. The best CINA-result was obtained using committee of experts (CE) method (*"vn11"*) applied to the following 7 submissions: *"vn1"* and *"vn10-vn10e"*.

Random Forest (Breiman, 2001) model proved to be the most suitable in the case of SIDO-set. We used RF model with 1000 trees where 70 randomly selected features were used for any splitter. Then, we computed number of occurrences in the RF-object for any particular feature. These occurrences were used for further feature selection. For example, we used in the final submission 1030 features for SIDO0, 517 features for SIDO1 and only 203 features for SIDO2.

Table 3: Some additional results.

Data	Submission	# features	Fscore	TrainAUC	TestAUC
REGED1	vn14	400	0.7316	1	0.9522
REGED1	vn11d	150	0.8223	1	0.9487
REGED1	vn1	999	0.5	1	0.9445
REGED1	vn8	899	0.5145	1	0.9436
MARTI1	vn12c	500	0.5784	1	0.8977
MARTI1	vn14	400	0.5554	1	0.8953
MARTI1	vn3	999	0.5124	1	0.8872
MARTI1	vn7	899	0.4895	1	0.8722
SIDO0	vn9	203	0.5218	0.9684	0.946
SIDO0	vn9a	326	0.536	0.9727	0.9459
SIDO0	vn1	1030	0.5785	0.9811	0.943
SIDO0	vn14	527	0.5502	0.9779	0.9429

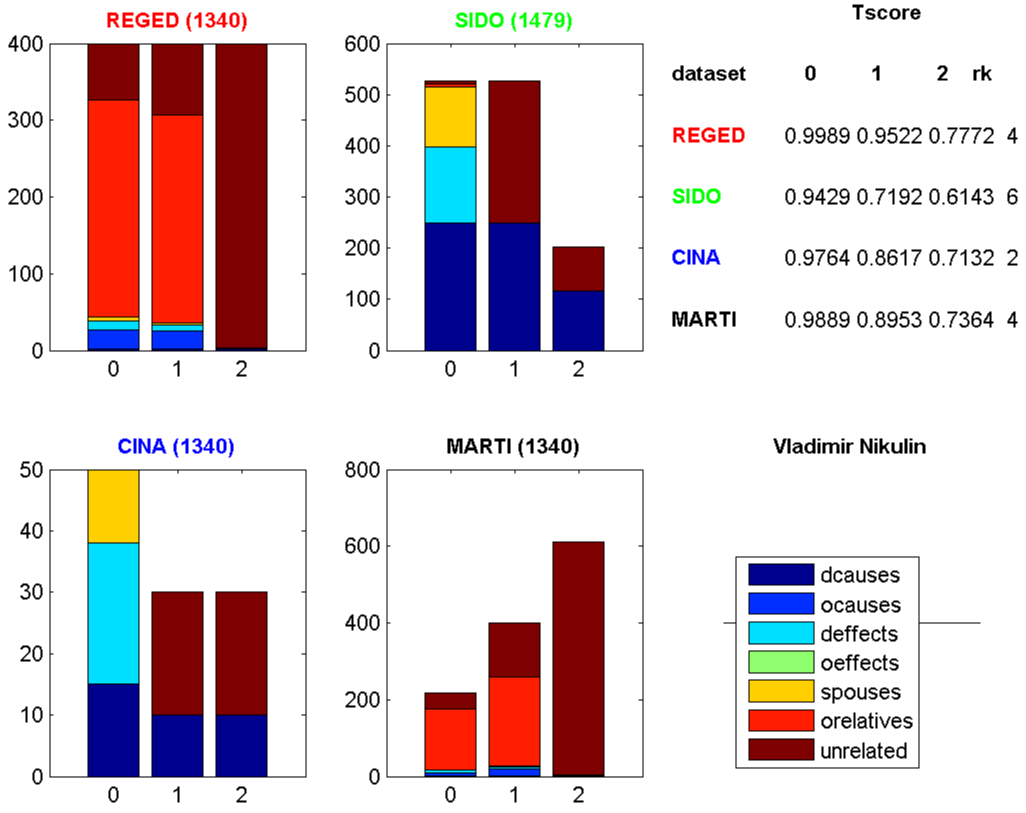
Lists of 100 manipulated features were given in the cases of REGED1 and MARTI1, but, according to our experience, this information was not really helpful, see Table 3 where submissions "vn1" (REGED) and "vn3" (MARTI) represent cases with all features. After removal of the manipulated features, test-results were slightly worse: see submissions "vn8" (REGED) and "vn7" (MARTI). Also, we have noticed surprising fact that value of Fscore for "vn7" is smaller comparing with Fscore for "vn3" (MARTI). Respectively, FS was conducted in the space of all features for the final submission "vn14". In both cases of REGED and MARTI we used 400 features including 33 manipulated features for REGED and 42 manipulated features for MARTI. It appears that in case of SIDO0 Fscore reflects rather relations with TrainAUC but not with TestAUC, see submissions "vn9" and "vn1".

Remark 6 *As a feedback the participants were able to view colour of their submission. For example, all TestAUC for SIDO0 in the Table 3 were green (means top 25% of all current results). Generally, this feedback appears to be too rough, and, definitely, cannot be accepted as a sufficient in the case when distributions of the training and test datasets are different.*

During competition we made the following number of full submissions (given in brackets): CINA(8), REGED(12), MARTI(32) and SIDO(6) plus some partial submissions. We did not use an opportunity of nested submissions (that means picking up the best out of the table of results) during the competition, and have found afterwards that this option may give significant advantage (see Table 4). Note that the methods which we used did not orient edges and cannot discover Market blanket as an expected outcome. Also, we did not use HITON-algorithm or similar as a component of our methods.

Figure 3 was downloaded from the web-site of the competition. Similar histograms of Jianxin Yin and Prof. Zhi Geng's Group (who won best overall contribution award) demonstrate much larger proportions of *dcauses* and *ocauses*. However, Jianxin Yin and his team did not produce significant improvement in terms of average AUC.

We have found that our results against unmanipulated datasets are quite competitive (see column "CASE0" in the Table 2). In particular, CINA0-result is the best.



Tscore

dataset	0	1	2	rk
REGED	0.9989	0.9522	0.7772	4
SIDO	0.9429	0.7192	0.6143	6
CINA	0.9764	0.8617	0.7132	2
MARTI	0.9889	0.8953	0.7364	4

Figure 3: Histograms of selected features (evaluated by the competition organizers) where dcause: direct cause, deffect: direct effects, ocauses: other causes, oeffects: other effects, spouses: parent of a direct effect, orelatives: other relatives, unrelated: completely irrelevant.

3.1 Post-challenge submissions

Using an opportunity of post-challenge submissions we were able to improve all results against manipulated sets significantly. It is interesting to note that very competitive results for REGED and MARTI-sets (see Table 4) were produced using the most simplest linear regression. Regularization was not necessary here because of the ultimate reduction of the number of features. The property when TrainAUC is smaller comparing with TestAUC (MARTI-set) may be viewed as a very interesting side effect of manipulation. Also, we were trying to use AdaBoost algorithm against data with the same feature selection as in the Table 4 but results were very poor.

Based on our experience, feature selection was the most important in order to achieve all results of the Table 4. Also, we have found that the Algorithm 1 is particularly efficient if we have prior information that the number of required features should be very small, and, consequently, classification algorithm may be very simple. It appears that design of the SIDO1 and SIDO2-sets was essentially different. Respectively, the number of features in the best submission was a quite significant, and the numbers of previous submissions were three times greater comparing with other sets.

Table 4: Results of the post-challenge submissions against manipulated sets where the following abbreviations were used: 1) NoF – number of used features; 2) NoS – number of previous submissions; 3) LR – linear regression with squared loss function; 4) RF – random forest; 5) Exp. – optimisation with exponential loss function (1); 6) BestChAUC – best challenge AUC.

Data	Method	NoF	Fscore	TrainAUC	TestAUC	NoS	BestChAUC
REGED1	LR	8	0.7133	0.9855	0.9861	9	0.9787
REGED2	LR	5	0.9985	0.9571	0.9467	8	0.8392
REGED1	Exp.	8	0.7133	0.9885	0.9867	10	0.9787
REGED2	Exp.	5	0.9985	0.9605	0.9513	9	0.8392
SIDO1	RF	128	0.5348	0.8681	0.7512	28	0.7532
SIDO2	RF	128	0.5348	0.8681	0.7359	28	0.6684
CINA1	AdaBoost	4	0.5455	0.8758	0.8694	5	0.8691
CINA2	AdaBoost	4	0.5455	0.8758	0.872	5	0.8157
MARTI1	LR	4	0.6429	0.8433	0.9407	8	0.947
MARTI2	LR	3	0.9995	0.7542	0.8049	9	0.7975
MARTI1	Exp.	4	0.6429	0.845	0.9469	9	0.947
MARTI2	Exp.	3	0.9995	0.7613	0.8296	10	0.7975

3.2 An exponential loss function

The following exponential loss function

$$\exp\{-\rho \cdot y_t \cdot u_t\}, \; u_t = \sum_{j=1}^{\ell} w_j \cdot x_{tj}, \; \rho > 0, \tag{1}$$

appears to be more natural comparing with squared loss function which over-punish large values of the decision function. However, application of the loss function (1) may not be simple because we cannot optimize step size in the case of the gradient-based optimization. Respectively, we will need to maintain low level of the step size in order to ensure stability of the algorithm. As a result, convergence of the algorithm may be very slow. In the cases of REGED or MARTI training sets with 3-8 features (see Table 4) we don't need to be worried about time problem: 100000 iterations until full convergence were conducted within 3min.

3.3 UCF-2008 data-mining competition

This recent competition was organized by the department of statistics and actuarial science of the University of Central Florida[2].

The available data are strongly imbalanced: 858620 (where 9737 positive and 848883 negative) units for training (labeled) and 95960 for testing (unlabeled). Any data-entry includes label, id and 61 features which are not necessarily numerical. Using special Perl software we transformed data into sparse format with 530 binary features. Then, we split the labeled data into 2 parts for training (90%) and testing (10%), and applied $k = 1000$ balanced training subsets where representatives from the larger pattern were selected randomly. As an outcome, the system produced matrix of linear regression coefficients M where rows represent random subsets and columns represent features. Based on this matrix we made an assessment of how stable is

2. http://dms.stat.ucf.edu/competition08/home.htm

influence of the particular features. It is proposed to keep in the model only features with stable influence (the ratio of the mean to StDev must be bigger or equal comparing with selected value of threshold parameter $\Delta = 0.5$). As a consequence, number of binary features was reduced to 320.

Our entry produced $AUC = 0.6645$ - third best result.

3.3.1 UNCERTAINTY ESTIMATION

Using above matrix of regression coefficients M we can estimate uncertainty associated with any particular data entry. First, we compute k predictions where k is number of random sets. Then, we can measure the corresponding standard deviation or empirical probabilities of deviation from the sample mean for any given margin.

3.4 Computation Time and Used Hardware

A Dell desktop with 3GB RAM, 2.4GHZ INTEL CORE 2 DUO, was used for the most of computations. For example, experiment with 10000 random sets as it is described in the Section 3 took about 17 hours according to the special program written in C. We spent about 2 hours in order to generate random forest for SIDO-set with 1000 trees where each tree had up to 8 levels of depth.

4. Concluding Remarks

Computational statistics is a relatively new scientific area, which may be viewed as one of the most promising areas of contemporary science. High technologies are generating large data sets and new problems, which must be addressed. Data mining competitions represent a rapidly growing and very important part of computational statistics. Practically any large commercial company in the world has data mining department, which is responsible for data analysis and modeling. Additionally, companies are hiring consultants in order to produce an alternative solutions and check effectiveness of their own results. These activities may be quite expensive, but unavoidable.

Generally, practical experience is the best way to learn, and participation in data mining competitions may be useful for wide range of researchers including academics, consultants and students in particular.

We understand that Causal Discovery competition was motivated by some interesting theoretical papers. However, in practical applications we are dealing not with pure probability distributions, but with mixtures of distributions, which reflect changing in time trends and patterns. Accordingly, it appears to be more natural to form training set as an unlabeled mixture of subsets derived from different (manipulated) distributions, for example, REGED1, REGED2,...,REGED9. As a distribution for the test set we can select any "pure" distribution.

Another point, "blind learning" (case when training and test data-sets have different distributions) appears to be interesting as a form of gambling. But in most practical applications proper organized validation is the most important. Respectively, it will be good to apply traditional strategy: split randomly available test-set into 2 parts 50/50 where one part will be used for validation, second part for the testing.

We considered in this paper several methods which may be used independently or in conjunction. We cannot expect that any of the methods may demonstrate an absolute superiority against the others. Therefore, performance of the particular method depends on the dataset, and the main strength of our approach rests on flexibility.

Acknowledgments

I would like to thank actuarial department of Suncorp for the valuable support. The paper was improved thanks to the comments and suggestions of the reviewers and organizers of the WCCI-2008 Causal Discovery competition.

References

C. Aliferis, I. Tsamardinos, and A. Statnikov. Large-scale feature selection using markov blanket induction for the prediction of protein-drug binding. In *Technical Report DSL 02-06*, 2002.

C. Aliferis, I. Tsamardinos, and A. Statnikov. HITON: a novel Markov blanket algorithm for optimal feature selection. pages 21–25. AMIA 2003, 2003.

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

I. Guyon, C. Aliferis, and A. Elisseeff. Causal feature selection. In H. Liu and H. Motoda, editors, *Computational Methods of Feature Selection*. Chapman and Hall, 2007.

P. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396): 945–960, 1986.

M. Jordan. Graphical model. *Statistical Science*, 19(1):140–155, 2004.

V. Nikulin. Learning with mean-variance filtering, SVM and gradient-based optimization. In *International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 16-21*, pages 4195–4202. IEEE, 2006.

D. Rubin. Bayesian inference for causal effects: the role of randomisation. *The Annals of Statistics*, 6(1): 34–58, 1978.

H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

I. Tsamardinos, C. Aliferis, and A. Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Technical Report DSL 03-06*, 2004.

Bernoulli Mixture Models for Markov Blanket Filtering and Classification

Mehreen Saeed MEHREEN.SAEED@NU.EDU.PK
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore Campus, Pakistan

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

This paper presents the use of Bernoulli mixture models for Markov blanket filtering and classification of binary data. Bernoulli mixture models can be seen as a tool for partitioning an *n*-dimensional hypercube, identifying regions of high data density on the corners of the hypercube. Once Bernoulli mixture models are computed from a training dataset we use them for determining the Markov blanket of the target variable. An algorithm for Markov blanket filtering was proposed by Koller and Sahami (1996), which is a greedy search method for feature subset selection and it outputs an approximation to the optimal feature selection criterion. However, they use the entire training instances for computing the conditioning sets and have to limit the size of these sets for computational efficiency and avoiding data fragmentation. We have adapted their algorithm to use Bernoulli mixture models instead, hence, overcoming the short comings of their algorithm and increasing the efficiency of this algorithm considerably. Once a feature subset is identified we perform classification using these mixture models. We have applied this algorithm to the causality challenge datasets. Our prediction scores were ranked fourth on SIDO and our feature scores were ranked the best for test sets 1 and 2 of the same dataset.

Keywords: Markov blanket filtering, mixture models, feature selection

1. Introduction

The term Markov blanket was coined by Pearl (1988). The Markov blanket (MB) of a feature variable represents the set of features/attributes required to exactly predict the behavior of that variable. In a Bayesian network the Markov blanket consists of parents, children and spouses of the node representing that feature variable. The work described in this paper concentrates on identifying the MB of the target variable. Practically, it is not possible to identify the MB of the target variable exactly because of computational issues or lack of sufficient data. Koller and Sahami (1996) presented a feature selection algorithm called Markov Blanket filtering (MBF) which outputs an approximation to the MB of the target variable. Their algorithm can also output a sorted feature list according to the relevance of a feature with respect to the target variable. To run this algorithm we have to assume K which is the size of the conditioning sets. However, large K values are not possible due to computational issues.

In order to find the Markov blanket of target variable, we adapted the algorithm of Koller and Sahami to use Bernoulli mixtures so that large K values can be used. The use of Bernoulli mixture models for classification is not new. The basic formula for a Bernoulli mixture model was first proposed by Duda and Hart (1973). They have been successfully used for OCR tasks by Juan and Vidal (2004) and Grim *et al.* (2000) and in supervised text classification tasks (Juan and Vidal, 2002). Bernoulli mixtures have also been used for supervised dimensionality reduction tasks (Sajama and Orlitsky, 2005).

Recently, we used Bernoulli mixtures for dimensionality reduction and showed how this transformed data can be used as input to a classifier giving rise to a hybrid model of learning (Saeed, 2008; Saeed and Babri, 2008). We used Bernoulli mixtures for mapping raw input data onto a new probability space. Such a transformation results in an immense reduction in the dimensionality of original data. It also achieves better classification results than individual models. The algorithm, described in this paper, calculates the entropy values from Bernoulli mixtures instead of calculating them from the actual training data. We employed this technique to the causality challenge datasets (WCCI, 2008a), i.e., SIDO and CINA datasets. Our results were amongst the top ranked entries in the competition.

The outline of this paper is as follows: In Section 2, Koller and Sahami's MBF algorithm is presented. In Section 3, Bernoulli mixtures are briefly introduced and the Bernoulli mixtures based MBF algorithm is described. The simulation results on the causality challenge datasets are presented in Section 4 and finally the conclusions are given in Section 5.

2. Markov Blanket Filtering (MBF) By Koller and Sahami (1996)

Koller and Sahami (1996) presented an algorithm for Markov blanket filtering (MBF) which is a greedy search algorithm, for identifying the MB of the target variable, based upon an optimal feature selection criterion. We will briefly describe their algorithm in this section. Let $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ be a set of features and $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ be a corresponding assignment of values. Let $\mathbf{G}$ be a subset of the feature set $\mathbf{F}$ and $\mathbf{f_G}$ be a projection of $\mathbf{f}$ onto the variables in $\mathbf{G}$. The MBF algorithm minimizes the divergence between $P(C|\mathbf{F} = \mathbf{f})$ and $P(C|\mathbf{G} = \mathbf{f_G})$ using an expected conditional entropy measure given by:

$$\delta_{\mathbf{G}} = \sum_{\mathbf{f}} P(\mathbf{f}) D_{KL}\Big(P(C|\mathbf{f}) \parallel P(C|\mathbf{f_G})\Big)$$

where C is the class label and D_{KL} is the Kullback-Leibler (KL) divergence between the true distribution $P(C|\mathbf{F} = \mathbf{f})$ and its estimated distribution $P(C|\mathbf{G} = \mathbf{f_G})$. This divergence is given by $D_{KL}(p \parallel q) = \sum_{x\in\Omega} p(x) \log \frac{p(x)}{q(x)}$. The probability space Ω is the set of all possible target labels. If we can find a feature set $\mathbf{G}$ for which $\delta_{\mathbf{G}}$ is very small then $\mathbf{G}$ can be used as an approximation to the full feature set $\mathbf{F}$ for predicting the target class.

Theoretically, Koller and Sahami have shown that a feature F_i can be omitted from a possible feature set $\mathbf{G}$ by finding the Markov Blanket, $\mathbf{M}$, for F_i. If the MB for F_i can be identified then F_i can be safely removed from $\mathbf{G}$ without an increase in the divergence from the true distribution. Practically, it is not possible to pinpoint exactly the MB of F_i, hence, heuristics have to be applied. The algorithm selects a candidate set $\mathbf{M}_i$ for each feature F_i and estimates how close $\mathbf{M}_i$ is to being the MB of F_i. The candidate set $\mathbf{M}_i$ is composed of those features which have the highest correlation with F_i. The feature F_i for which $\mathbf{M}_i$ is closest to being the MB is omitted. The approximation is based upon the following expected cross entropy measure:

$$\delta_{\mathbf{G}}(F_i|\mathbf{M}_i) = \sum_{\mathbf{f}_{\mathbf{M}_i}, f_i} P(\mathbf{M}_i = \mathbf{f}_{\mathbf{M}_i}, F_i = f_i) D_{KL}\Big(P(C|\mathbf{M} = \mathbf{f_M}, F_i = f_i) \parallel P(C|\mathbf{M} = \mathbf{f_M})\Big)$$

If $\mathbf{M}_i$ is the MB for F_i then $\delta_G(F_i|\mathbf{M}_i) = 0$ and this value will be small for an approximate MB. Algorithm 1 outlines the basic steps for selecting an approximate feature set for predicting the target variable. In this algorithm K determines the size of the conditioning set. We would like K to be as large as possible, however, practically larger values of K lead to fragmentation of the dataset and reduce the accuracy of the probability estimates used in estimating the cross entropy measure. Also, large values of K are not practical to implement as they require the counting of $2^{(K+1)}$ combination of values to calculate the cross entropy measure. This algorithm can also be used to order variables, according to priority or relevance to target variable, based upon the cross entropy measure.

```
1. Let $\mathbf{G} = \mathbf{F}$
2. Repeat until desired number of features in $\mathbf{G}$
   a. For each feature $F_i \in \mathbf{G}$, let $\mathbf{M}_i$ be the set of $K$ features $F_l \in \mathbf{G} - \{F_i\}$ which have the highest
      correlation with $F_i$.
   b. Compute $\delta_G(F_i|\mathbf{M}_i)$ for each $i$.
   c. Choose the $i$ for which this term is minimal and define $\mathbf{G} = \mathbf{G} - \{F_i\}$
```

Algorithm 1: MBF Algorithm by Koller and Sahami (1996)

3. Multivariate Bernoulli Mixtures

Suppose we have a sample of training data, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, consisting of m input vectors. Each input vector $\mathbf{x} \in R^n$. If we want to estimate D mixture components from this data, then a finite mixture model is described by a probability (density) function given by $p(\mathbf{x}) = \sum_{d=1}^{D} \pi_d p(\mathbf{x}|d)$. Here, π_d is the prior of each mixture and $p(\mathbf{x}|d)$ is its component-conditional probability (density) function.

A multivariate Bernoulli mixture model assumes that each component of the model is an n-dimensional multivariate Bernoulli probability distribution, each component or mixture having its own set of parameters. For a single binary vector $\mathbf{x}_k \in \{0,1\}^n$, the form of this distribution, in the d^{th} mixture is given by (Bishop, 2006):

$$p(\mathbf{x}_k|d) = \prod_{i=1}^{n} (p_{di})^{x_{ki}} (1 - p_{di})^{1-x_{ki}} \qquad \forall k, 1 \le k \le m, \forall d, 1 \le d \le D$$

Here $p_{di} \in [0,1]$ is the probability of success of the i^{th} component of vector $\mathbf{x}_k$ for the d^{th} mixture, i.e., $p_{di} = p(x_{ki} = 1|d)$. Also, we are assuming that the n-dimensional vector $\mathbf{x}$ has n independent component attributes. The parameter θ to be determined is the probability of success for each attribute of vector $\mathbf{x}$, i.e., $\theta = \mathbf{p}$ where $\mathbf{p} \in [0,1]^n$.

We can use expectation maximization (EM) algorithm to find the parameters of each mixture component as described in Appendix A. Also, Appendix B explains the use of these mixtures for dimensionality reduction and classification. For details we refer the reader to our recent work in this area (Saeed, 2008; Saeed and Babri, 2008).

3.1 Adaptation of MBF algorithm Using Bernoulli Mixture Models

We found that the main problem with the MBF algorithm is in computing the expected cross entropy from training data. It is not possible to compute this measure using large values of K because of computational issues and problems with data fragmentation. However, we can compute an approximation to this measure via the use of Bernoulli mixture models. Bernoulli mixtures identify regions of high data density on the corners of a hypercube and we can exploit this fact to estimate the cross entropy measure for large K values.

Let's look at one Bernoulli mixture more closely. The parameters of the d^{th} Bernoulli mixture are completely specified by the probability vector $\mathbf{p}_d$ and its prior π_d. Here, the i^{th} component of $\mathbf{p}_d$, i.e., p_{di} represents the probability that the i^{th} binary feature is one in the d^{th} Bernoulli mixture. We can threshold these probability values to see which corner of the hypercube is represented by this mixture. A probability value greater than 0.5 can be taken as a one and zero otherwise.

As an example lets take a 3 feature case. The mixture $(0.7, 0.9, 0.1)$ with prior π_d represents the corresponding feature vector $(1, 1, 0)$. It shows us that within the dataset this mixture occurs π_d fraction of times and within this mixture, feature 1 is one with probability 0.7, feature 2 is 1 with probability 0.9 and feature 3 is 0 with probability 0.9. We can also estimate the probability of occurrence of this feature vector. If we make an optimistic estimate then we can say that feature 1 will be 1 when features 2 and 3 are 1 and 0 respectively at the most 70% of times. However, we can also say that feature 1 will be 1 at least 50% times when feature 2 and 3 are 1 and 0 respectively. So making an optimistic guess, we can estimate the probability of feature vector (1,1,0) to be at the most $0.7 * \pi_d$. This also tells us that $(1, 1, 1)$ occurs at the most 10% times and similarly (0,1,0) occurs at the most 30% times.

We can formalize the above scheme and express the probability of feature vector, $\mathbf{x}$, when given the probability vector $\mathbf{p}_d$ of the d^{th} mixture, as:

$$p(\mathbf{x}|d) = \pi_d * \min_i p_{di}^{x_i}(1 - p_{di})^{1-x_i} \tag{1}$$

Suppose $\mathcal{X}$ represents the set of all binary vectors in $\{0, 1\}^n$, hence, $|\mathcal{X}| = 2^n$. We will use the term 'main vector' for the feature vector that can be derived from a mixture density and has the highest probability of occurrence according to Eq. (1). Hence, 'main vector' $\mathbf{v}$ is defined as:

$$\mathbf{v} = \arg\max_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}|d)$$

The main vector can also be derived by thresholding the probability values of a Bernoulli mixture with probability vector $\mathbf{p}$. The i^{th} feature of the main vector is assigned binary 1 value if its probability $p_i \geq 0.5$ and binary 0 value otherwise.

The algorithm for finding the MBF is the same as Algorithm 1, except that Step 2b for calculating the cross entropy measure is replaced by Algorithm 2. Here, the cross entropy measure for a feature F_i, $1 \leq i \leq n$, is approximated from the Bernoulli mixtures instead of training data. To run this algorithm, we first determine the Bernoulli mixtures, from the training data, for each class label separately. d_k^+ and d_k^- are the k^{th} mixtures for the positive and negative classes (C^+ and C^-) respectively. The total mixtures, D^+ and D^-, for positive and negative classes respectively, are specified as one of the initial parameters to the EM algorithm and they don't have to be the same. Hence, the total mixtures generated from the entire training data is $D = D^+ + D^-$. Effectively, this algorithm finds the probability of each main vector within all Bernoulli mixtures of positive and negative classes. If the number of Bernoulli mixtures for each class is small then this can be done efficiently for large values of K.

Time Complexity: Section 4.1 shows that this algorithm has particular advantage when total training examples and features are very large in a dataset. If we don't count the initial step of computing correlation matrix and D Bernoulli mixtures then subsequent steps of the algorithm have roughly a time complexity of $O(rnKD^2c)$, where r is the number of features to eliminate, n is the total features and c is the total number of classes. In contrast, Koller and Sahami's algorithm for m training examples takes $O(rnKm2^Kc)$ as the cross entropy measure

Take all sub-vectors $\mathbf{v}_j$, $1 \le j \le D$, of main vectors, in the mixture models found by EM so that $\mathbf{v}_j \in \{0,1\}^K$, i.e., $\mathbf{v}_j$ is composed of all features that have the highest correlation with the target (corresponds to $\mathbf{M}$ of Algorithm 1). As before, F_i is a feature and f_i is a corresponding assignment of value to F_i. In our case $f_i \in \{0,1\}$. The cross entropy measure $\delta'_G(F_i|\mathbf{M}_i)$ for each feature F_i, $1 \le i \le n$, can be computed as:

1. Find the probability of $\mathbf{v}_j$ in the positive class as $p^+ = \sum_{k=1}^{D^+} \pi_k p(\mathbf{v}_j|d_k^+)$
2. Find the probability of $\mathbf{v}_j$ in the negative class as $p^- = \sum_{k=1}^{D^-} \pi_k p(\mathbf{v}_j|d_k^-)$
3. Calculate $P(C^+|\mathbf{v}_j) = \frac{p^+}{p^+ + p^-}$, assuming equal priors for both classes and similarly calculate $P(C^-|\mathbf{v}_j)$
4. Repeat 1,2,3 to calculate $P(C^+|\mathbf{v}_j, f_i = 0)$ and $P(C^+|\mathbf{v}_j, f_i = 1)$
5. Sum over all sub-vectors and all values of f_i to calculate cross entropy measure, $\delta'_G(F_i|\mathbf{M}_i)$, approximated by: $\sum_{l\in\{0,1\}} \sum_{j=1}^{D} D_{KL}(P(C|\mathbf{v}_j, f_i = l)||P(C|\mathbf{v}_j))$

Algorithm 2: Approximating the cross entropy measure using Bernoulli mixtures

is being computed from mxK sized data and we need to look at 2^K combination of values. However, we are computing our cross entropy measure from DxK sized data, where we are only looking at D main vectors in each Bernoulli mixture, resulting in a dramatic reduction in time.

4. Simulations

To get an insight of MBF using Bernoulli mixtures we applied it to the SIDO and CINA datasets of the causality challenge. Each dataset has three types of test sets corresponding to the same training data. Test sets with subscript 0 are un-manipulated datasets and have the same distribution as the training data. Datasets with subscripts 1 and 2 have features that have been manipulated by some external agent and can have distracters or probes. Details can be found on the challenge's website (WCCI, 2008a) and are summarized in Table 1. The identity of each feature was not revealed to the challenge participants.

SIDO dataset is a pharmacology dataset that has only binary features. The features denote descriptor molecules which have been tested against AIDS HIV virus. The labels of each instance indicate molecular activity. Some of the features are actual molecular descriptors and some of them are artificially generated probes. The identity of each feature was not revealed to the contestants of the challenge. CINA dataset is an econometrics dataset derived from UCI machine learning repository 'Adult' dataset. In this case also, the details of each feature were not revealed to us.

The evaluation of results was based upon the average accuracy, 'Tscore', of the three test sets for each dataset (Guyon et al., 2008). The contestants had the option of submitting a sorted nested feature subset list of r ($r \le n$) features, sorted according to the importance of each feature in predicting the target. Hence, multiple predictions could be made using the first s features, where s varies by powers of 2, i.e., $1, 2, 4, 8, \ldots$ features. An 'Fscore' was computed to indicate how well the predicted set of features matched with the actual MB of the target variable (known only to the organizers) (Guyon et al., 2008). For our submissions, we submitted a sorted list of features ranked according to a feature's relevance in predicting the target, and a set of predictions corresponding to the nested feature subsets. Submitting a set of predictions gives us an added advantage over those participants who submitted only a single set of predictions.

During the challenge we made 29 complete entries on SIDO and 16 complete entries on CINA dataset. A complete entry consists of predictions on all three test files of a dataset (i.e., subscript 0, 1 and 2). Once we submitted an entry, we got immediate feedback on our performance, whether our entry was in the first, second, third or fourth quartile, as compared to the other participants. Only our last submitted entry was considered for final ranking. We used the quartile information and 2 fold cross validation accuracy to determine the final models for the last entry. Our entire source code was written in C++ and Matlab in the CLOP framework (Saffari and Guyon, 2006).

Table 1: SIDO and CINA challenge datasets

Dataset	**Domain**	**Continuous Features**	**Binary Features**	**Positive Examples**	**Negative Examples**	**Total**
SIDO	Pharmacology	0	4932	452	12226	12678
CINA	Econometrics	24	108	3939	12094	16033

4.1 Results on SIDO, The Pharmacology Dataset

Table 2: Results on the SIDO dataset. Fnum is the number of features, Fscore is the score on the feature set (see Section 4), Tscore is the accuracy on the test set. The entries marked by '*' were classified using the naive Bayes' classifier. For the rest of the entries the classifier used was a combination of Bernoulli mixtures and ensemble of neural networks.

	SIDO0			**SIDO1**			**SIDO2**		
K	**Fnum**	**Fscore**	**Tscore**	**Fnum**	**Fscore**	**Tscore**	**Fnum**	**Fscore**	**Tscore**
10	1024	0.4717	0.9332	1024	0.6895	0.7140	1024	0.6895	0.6145
15	1024	0.4857	0.9407	1024	0.6928	0.7464	128	0.6928	0.7155
20	512	0.4831	0.9457	2048	0.6635	0.6565	1024	0.6635	0.6134
25	512	0.4921	0.9381	1024	0.7335	0.7124	1024	0.7335	0.6325
30	1024	0.4592	0.9434	4096	0.7242	0.7246*	512	0.7242	0.6216
40	4096	0.4095	0.9391*	4096	0.694	0.7309*	4096	0.694	0.5788*

We generated Bernoulli mixtures on the SIDO dataset. The initial values specified to the algorithm for number of Bernoulli mixtures was 10 for both classes, however, we ended up with 4 and 7 Bernoulli mixtures for the positive and negative class respectively. There were 16,033 training examples in this set and we were unable to run Koller and Sahami's algorithm on this set for $K \geq 6$ due to limited computing resources available to us. However, we easily ran it for large values of K with the Bernoulli mixture version of MBF as there are only 11 mixtures and the combinations needed to compute cross entropy values is immensely smaller. Hence, we compute the MB from 11×4932 sized data instead of the full 16033×4932 sized data. The feature scores along with test scores on different values of K for the SIDO0, SIDO1 and SIDO2 datasets are given in Table 2.

For this dataset we returned our predictions on nested feature subsets, as described in Section 4. The challenge organizers chose the feature set for which the classification accuracy was maximum and hence, this decides the 'fnum' value in Table 2. Depending upon the feature subset we used two different classifiers. Either the naive Bayes' classifier or a combination of Bernoulli mixtures and ensemble of neural networks was used to classify data (see Appendix B). In an ensemble of neural networks, the overall prediction was made by combining outputs from individual neural networks (Saffari and Guyon, 2006). We used maximum of 10 neural network models, each trained with a different number of neurons in the hidden layer.

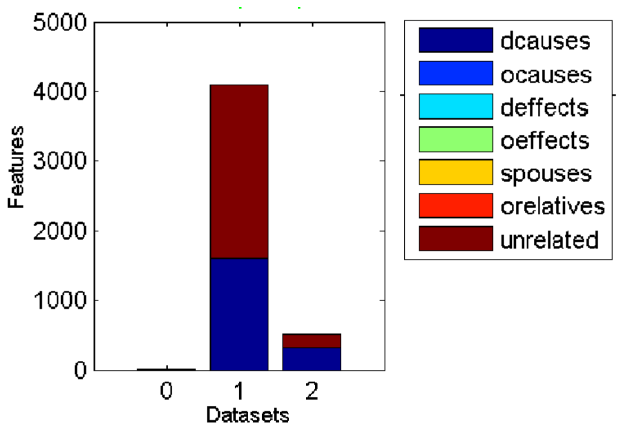


Figure 1: Profile of features selected for SIDO. Legend: dcause=direct cause, deffect=direct effects, ocauses=other causes (indirect), oeffects=other effects (indirect), spouses=parent of a direct effect, orelatives=other relatives, unrelated=completely irrelevant (see WCCI, 2008b)

Our final entry for SIDO1 and SIDO2 was made using $K = 30$. It has the best Fscores amongst all ranked entries in the challenge on SIDO1 and SIDO2. Figure 1 shows the profile of features selected with $K = 30$. It is interesting to note that the algorithm finds only the direct causes of target and irrelevant features in case of SIDO1 but the accuracy of the classifier is quite good even in the presence of the irrelevant features. However, in case of SIDO2, the algorithm is able to find out a higher fraction of direct causes as compared to irrelevant features and the classifier performs quite well.

The classification accuracy on SIDO0 is almost the same for all values of K. Interestingly, the performance of a simple naive Bayes' classifier and the hybrid model is almost identical on SIDO0 and SIDO1. The best classification results are obtained for $K = 15$ for both SIDO1 and SIDO2 where the algorithm gives best results using 1024 and 128 features respectively and classification was done using a hybrid model of Bernoulli mixtures and an ensemble of neural networks.

4.2 Results on CINA, The Econometrics Dataset

Table 3: Results on the CINA dataset. See Table 2 for explanation of Fnum, Fscore and Tscore

	CINA0			**CINA1**			**CINA2**		
No.	**Fnum**	**Fscore**	**Tscore**	**Fnum**	**Fscore**	**Tscore**	**Fnum**	**Fscore**	**Tscore**
1	32	0.5069	0.9751	16	0.7858	0.8248	16	0.7858	0.6867
2	21	0.4875	0.9727	21	0.5196	0.8188	16	0.5196	0.6627

Table 3 shows the results obtained on two of our submitted entries for CINA dataset. The first row shows the feature scores and accuracy of the three test sets. Here, feature subset selection algorithm was used where forward selection was done based on the accuracy achieved by the Naive Bayes' classifier on the training set. The second row shows the results of feature selection when MBF using Bernoulli mixtures was used for selecting binary features and feature subset selection was used for selecting continuous features. Here, the total features used include

6 continuous features and the rest are binary. For this dataset classification was performed using an ensemble of neural networks on the selected feature values after standardizing the data. We can see that the two methods have almost the same percentage accuracy on the three types of datasets, however, feature subset selection gives much better feature scores.

Figure 2 shows the profile of features selected for CINA on our last entry. It is interesting to see that our forward selection algorithm with naive Bayes' is able to find many direct causes of the target variable for all three datasets. The percentage of irrelevant features identified is very small compared to the direct causes that have been identified. For dataset 0 this algorithm is able to identify many indirect causes of the target and also the parents of the direct causes.

4.3 Comparison and Discussion of Results

Table 4: Comparison of results. Top ranking is the Tscore of the winning participant. Max Test score is the highest achieved using actual knowledge of causal features.

	K=15		**Last Submission**			
Dataset	**Features**	**TScore**	**Features**	**Tscore**	**Top ranking**	**Max Test Score**
SIDO0	1024	0.9407	8	0.9391	0.9443	0.9467
SIDO1	1024	0.7464	4096	0.7246	0.7532	0.7893
SIDO2	128	0.7155	512	0.6216	0.6684	0.7674
	K=3					
CINA0	21	0.9727	32	0.9751	0.9765	0.9788
CINA1	21	0.8188	16	0.8248	0.8691	0.8977
CINA2	16	0.6627	16	0.6867	0.8157	0.891

Table 4 shows a comparison of results with the top ranking entries on both SIDO and CINA datasets. The table shows our results with $K = 15$ on SIDO dataset, results with our last submitted entry and test scores of the winning participants. The column with the 'Max Test Score' indicates the best score reachable, as estimated by reference entries, using the knowledge of true causal relationships, not available to participants. Our last entry counted towards the ranking of participants and was ranked fourth. The average test score with K=15 was the best accuracy

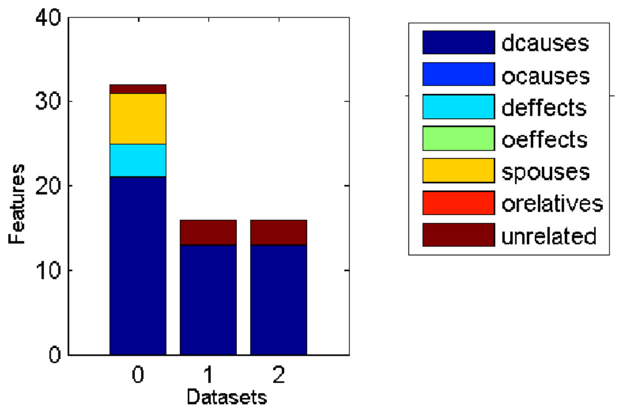


Figure 2: Profile of features selected for CINA. For legend see Figure 1 (WCCI, 2008b)

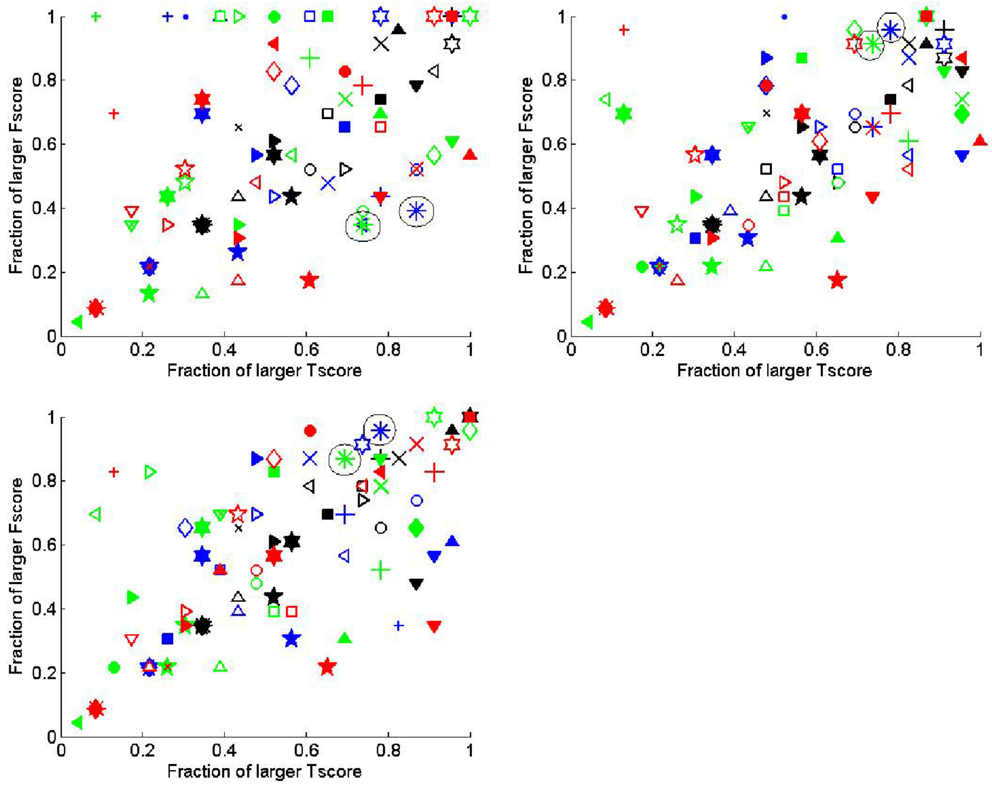


Figure 3: Pairwise performance of participants on test sets 0, 1 and 2. Green is SIDO and blue is CINA (see WCCI, 2008b). Our entry is encircled and denoted by '$*$'.

rate amongst all participants, however, it did not count towards the final ranking as it was not our last submitted entry. The original dataset had 4932 features and we can see from the table that we were able to make quite good predictions using a very small set of features.

In case of CINA, we attained good results by using only 21 features on CINA0 and CINA1 and 16 features for CINA2 for $K = 3$. Also, our last entry was made using subset feature selection, with the forward selection algorithm with Naive Bayes' algorithm (Alpaydin, 2005). Here the classifier used was an ensemble of neural networks. We can see that the two methods give almost the same accuracy, even though, they use a different feature set. Our test set accuracies are quite comparable with the top ranking participant for CINA0 and CINA1. However, our algorithm failed to produce good results for CINA2. Our last submitted entry was ranked sixth amongst all other entries.

Our feature scores (Fscores) were the best amongst all participants for test sets 1 and 2 of SIDO dataset and also for test sets 1 and 2 of the CINA dataset (Guyon et al., 2008). In order to further compare performances, the challenge organizers made pairwise comparisons between different challenge participants and counted the fraction of times our Tscore was better than other participants for a fixed number of features (Guyon et al., 2008). They did the same for Fscores also. Figure 3 shows the pairwise comparison. We have encircled our entries in black. Our feature scores are not very good for the test set 0, i.e., the data with no manipulations, but SIDO has a high Tscore as compared to the rest of the entries. Our feature scores for test sets 1

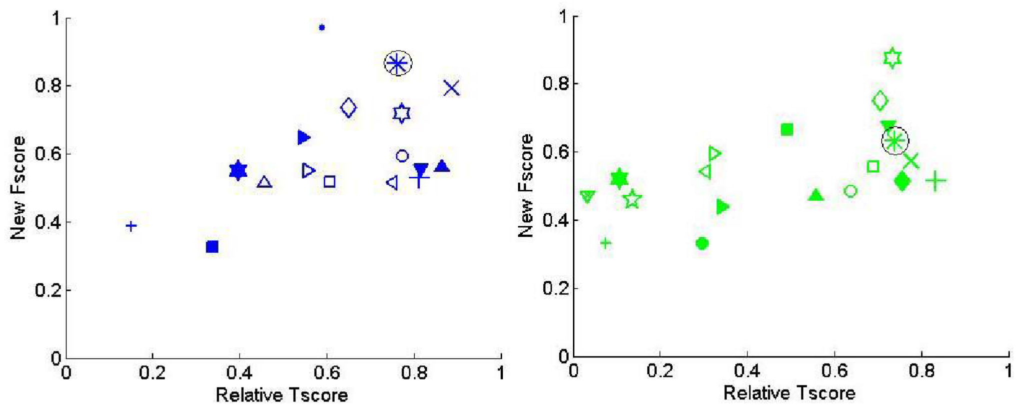


Figure 4: TScores Vs. new FScore for all participants for CINA (left) SIDO (right) datasets (see WCCI, 2008b). Our entry is encircled and denoted by '$*$'.

and 2 are significantly better than the rest of the participants on both SIDO and CINA datasets. Also, our Tscore is higher than the rest of the participants more than 70% of the times for test sets 1.

To further evaluate the results of the challenge, the organizers defined a new Fscore. The new Fscore was defined using the 3 versions of precision and recall, using as a set of features, the MB in set 1, MB and causes and effects in set 2 and all variables connected to the target in set 3 (Guyon et al., 2008). Figure 4 shows the plot of Tscore vs. the new Fscore of different participants for SIDO and CINA datasets. Again our entries are encircled in black. It can be seen that we do quite well on CINA dataset as compared to the rest of the participants and our results are considerably better than other participants for the SIDO datasets.

5. Conclusions

In this paper we discussed the use of Bernoulli mixture models for Markov Blanket filtering by adapting the original algorithm proposed by Koller and Sahami in 1996. Instead of using the training data for estimating entropy values we have used Bernoulli mixtures to approximate these values. Hence, in this way we can reduce the computations required for approximating the cross entropy values and use larger conditioning sets. We tried our method on the SIDO and CINA datasets of the causality workbench challenge. Our method was ranked fourth on SIDO and sixth on CINA. Our feature scores were ranked best on the test sets 1 and 2 of both SIDO and CINA. Also, one of our entries, submitted during the challenge, had the best results on SIDO, out of all challenge entries, but did not count towards the final ranking as it was not our last submitted entry.

As part of future work we would like to extend our algorithm to do causal feature selection. Right now we determine only the MB of the target variable but do not pin point the causes or effects of the target variable. Also, we are working on developing methods that determine a suitable value of K for a particular dataset so that model selection techniques can be applied to datasets that have probes or noise added to them.

Acknowledgments

Many thanks to Prof. Haroon Babri and Mr. Kashif Javed for their valuable comments and discussion during the challenge. The author would also like to thank Isabelle Guyon for her help and support during the causality challenge.

Appendix A. The EM Algorithm for Learning Bernoulli Mixtures

A finite mixture model is described by a probability function given by $p(\mathbf{x}) = \sum_{d=1}^{D} \pi_d p(\mathbf{x}|d)$. Learning the parameters of a finite mixture model is a statistical parameter estimation problem and we can use expectation maximization (EM) algorithm to estimate these parameters from a sample of training data $X = \{\mathbf{x}_k\}_{k=1}^{m}, \mathbf{x} \in R^n$. The EM algorithm maximizes the log likelihood function of data given by:

$$\mathcal{L}(\Theta|X) = \sum_{k=1}^{m} log\Big(\sum_{d=1}^{D} \pi_d p(\mathbf{x}_k|d)\Big) \tag{2}$$

Here Θ denotes the unknown variables to be estimated and consists of the priors, π_d, of each mixture and the parameters, θ_d, of each mixture distribution, i.e., $\Theta = \{\pi_d, \theta_d\}_{d=1}^{D}$.

A Bernoulli mixture model assumes that each component of the model is an n-dimensional multivariate Bernoulli probability distribution, each component or mixture having its own set of parameters. The form of this distribution for a single vector $\mathbf{x}_k \in \{0,1\}^n$ in the d^{th} distribution is given by (Bishop, 2006):

$$p(\mathbf{x}_k|d) = \prod_{i=1}^{n} (p_{di})^{x_{ki}} (1-p_{di})^{1-x_{ki}} \qquad (\forall k, 1 \le k \le m, \forall d, 1 \le d \le D)$$

Here $p_{di} \in [0,1]$ is the probability of success of the i^{th} component of vector $\mathbf{x}_k$ for the d^{th} mixture. Here the parameter θ to be determined is the probability of success of each attribute of vector $\mathbf{x}$, i.e., $\theta = \mathbf{p}$ where $\mathbf{p} \in [0,1]^n$.

The EM algorithm assumes that the observed data is incomplete and associates a vector of latent variables $\mathbf{z}_k = \{z_{k1}, z_{k2}, \ldots, z_{kD}\}$ with each data point. The latent variables are indicator variables, with $z_{kd} = 1$ indicating that the d^{th} mixture component generated the k^{th} data point. The EM optimization takes place iteratively in two steps. In step 1, also called the expectation step (E-Step), we estimate the expected values of the hidden variables assuming that the model parameters θ_d are known. In step 2, also called the maximization step (M-Step), we estimate the parameter values θ_d to maximize the likelihood of data, given by Eq. (2), on the basis of the latent variables calculated in the E-step. This is done iteratively until the parameters converge to stable values. To start the EM algorithm we initialize the probabilities with random values.

The form of E-step is the same for more or less all distributions and it is given by:

$$z_{kd} = \frac{\pi_d p(\mathbf{x}_k|d)}{\sum_{j=1}^{D} \pi_j p(\mathbf{x}_k|j)} \qquad (\forall d, 1 \le d \le D, \forall k, 1 \le k \le m) \tag{3}$$

The M-step determines the maximum likelihood estimate of the priors, of each distribution, as given below:

$$\pi_d = \frac{1}{m} \sum_{k=1}^{m} z_{kd} \qquad (\forall d, 1 \le d \le D)$$

Also, in this step the parameters of the particular probability distribution are estimated. These parameters depend on the probability function being used. For a Bernoulli mixture model, the M-step finds the maximum likelihood estimate of the probability of success of each vector component as given below:

$$\mathbf{p}_d = \frac{\sum_{k=1}^{m} z_{kd}\mathbf{x}_k}{\sum_{k=1}^{m} z_{kd}} \qquad (\forall d, 1 \le d \le D)$$

The simulations, described in this paper, use the regularized version of EM algorithm described by Li *et al.* (2005). Also, Laplacian prior can be used to smooth the probability estimates, hence, the probability values are estimated as below (γ is the regularization constant):

$$\mathbf{p}_d = \frac{1 + \sum_{k=1}^{m} z_{kd}(1 + \gamma \ln z_{kd})\mathbf{x}_k}{2 + \sum_{k=1}^{m} z_{kd}(1 + \gamma \ln z_{kd})}$$

The parameter D, i.e., the total mixtures to be determined is input to the algorithm by the user. We determine this number through cross-validation. Normally, we end up with fewer mixtures than originally specified, as the priors for many mixtures converge to zero and we can ignore those mixtures. This is especially true for sparse binary data with a high dimensional feature space. In such a case, no matter what the original value of D, specified to the regularized EM algorithm, we always end up with more or less the same number of mixtures with non-zero priors. Hence, the regularized version of EM algorithm is able to find an optimal number of clusters within data.

Appendix B. Classification Using Bernoulli Mixtures

In this section we give a brief overview of how we use Bernoulli mixtures for feature transformation and classification of instances. For details, we refer the reader to our previous work (Saeed, 2008; Saeed and Babri, 2008). We extend our terminology to include the class labels assigned to each example point. Suppose we have a set of c labeled classes $Q = \{q_1, q_2, \ldots, q_c\}$. We generate a total of D_i mixtures for class q_i. Let s_{id} represent the d^{th} mixture/cluster in the i^{th} class with prior for that mixture being given by π_{id} and $\sum_{d=1}^{D_i} \pi_{id} = 1$. Then the class conditional probability function, for the i^{th} class having D_i mixture components (given by $\{s_{id}\}_{d=1}^{D_i}$), is given by:

$$p(\mathbf{x}|q_i) = \sum_{d=1}^{D_i} \pi_{id}\, p(\mathbf{x}|s_{id})$$

Given the original dataset X, we perform a transformation on X given by: $T : X \to \Phi, X \in R^n, \Phi \in R^N$. The l^{th} component of the new feature vector $\boldsymbol{\phi}_k$ is given by:

$$\phi_{kl} = \frac{p(\mathbf{x}_k|s_{id})\pi_{id}}{\sum_{i=1}^{c}\sum_{j=1}^{D_i} \pi_{ij}\, p(\mathbf{x}_k|s_{ij})} \qquad (\forall k, 1 \le k \le m)$$

The index l has a value corresponding to every mixture of every class. For a sub-cluster s_{id}, the subscript l is given by $l = \sum_{j=1}^{i-1} D_j + d$. Here $\boldsymbol{\phi}_k \in R^N$, where $N = \sum_{i=1}^{c} D_i$. We can see that the numerator in the above expression is the same as that of Eq. (3) which represents the expectation of the latent variable in a mixture. The term in the denominator is used to normalize the entire feature vector over all the classes. If $N < n$ then our new feature set will lie in a lower dimensional space as compared to the original feature space and hence, this method can be used for dimensionality reduction.

We use the expectation of the latent variables as the transformed set of features for input to a classification algorithm. The feature vector is normalized so that the sum of components is unity. We have shown in our previous work (Saeed, 2008; Saeed and Babri, 2008) that these set of transformed features lead to an immense reduction in the dimensionality of data, especially when the data is sparse. We can apply any classification algorithm to classify the data, e.g., SVM, neural networks, boosting models, etc. The work described in this paper uses an ensemble of neural networks.

References

Ehem Alpaydin. *Introduction to Machine Learning*. Prentice-Hall of India Private Limited, 2005.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

Jiri Grim, Pavel Pudil, and Petr Somol. Multivariate structural Bernoulli mixtures for recognition of handwritten numerals. In *Proceedings of International Conference on Pattern Recognition (ICPR'00)*, 2000.

Isabelle Guyon, Constantin Aliferis, Greg Cooper, Andre Elissee, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov. Design and analysis of the causation and prediction challenge. In *Proceedings of Journal of Machine Learning Research*, 2008. to appear.

Alfons Juan and Enrique Vidal. On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710, December 2002.

Alfons Juan and Enrique Vidal. Bernoulli mixture models for binary images. In *Proceedings of 17th International Conference on Pattern Recognition (ICPR'04)*, 2004.

Koller and Sahami. Toward optimal feature selection. In *Machine Learning: Proceedings of the* 13^{th} *international conference*. Morgan Kaufman, 1996.

Haifeng Li, Keshu Zhang, and Tao Jiang. The regularized EM algorithm. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 807–812, 2005.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Mehreen Saeed. *Hands-on pattern recognition challenges in data representation, model selection, and performance prediction*, chapter Hybrid learning using mixture models and artificial neural networks. 2008. To appear, see http://www.clopinet.com/ChallengeBook.html.

Mehreen Saeed and Haroon Babri. Classifiers based on Bernoulli mixture models for text mining and handwriting recognition. In *Proceedings of International Joint Conference on Neural Networks, IEEE WCCI*, 2008.

Amir Saffari and Isabelle Guyon. *Quick start guide for CLOP*, May 2006. Available at http://ymer.org/research/files/clop/QuickStartV1.0.pdf.

Sajama and Alon Orlitsky. Supervised dimensionality reduction using mixture models. In *Proceedings of the 22nd international conference on machine learning*, pages 768–775, Bonn, Germany, 2005.

IEEE WCCI. Causality challenge #1: Causation and prediction, 2008a. See http://www.causality.inf.ethz.ch/challenge.php.

IEEE WCCI. Causation and prediction: Challenge analysis, 2008b. See http://clopinet.com/isabelle/Projects/WCCI2008/Analysis.html.

Partial orientation and local structural learning of causal networks for prediction

Jianxin Yin JIANXINYIN@MATH.PKU.EDU.CN
You Zhou ZHOUYOU@PKU.EDU.CN
Changzhang Wang CHANGZHANG@PKU.EDU.CN
Ping He SUNHP@PKU.EDU.CN
Cheng Zheng ZZHENGCCHENG@PKU.EDU.CN
Zhi Geng ZGENG@MATH.PKU.EDU.CN
School of Mathematical Sciences
Peking University
Beijing 100871, China

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

For a prediction problem of a given target feature in a large causal network under external interventions, we propose in this paper two partial orientation and local structural learning (POLSL) approaches, Local-Graph and PCD-by-PCD (where PCD denotes Parents, Children and some Descendants). The POLSL approaches are used to discover the local structure of the target and to orient edges connected to the target without discovering a global causal network. Thus they can greatly reduce computational complexity of structural learning and improve power of statistical tests. This approach is stimulated by the challenge problems proposed in IEEE World Congress on Computational Intelligence (WCCI2008) competition workshop. For the cases with and without external interventions, we select different feature sets to build prediction models. We apply the L1 penalized logistic regression model to the prediction. For the case with noise and calibrant features in microarray data, we propose a two-stage filter to correct global and local patterns of noise.

Keywords: Causal network, Local structural learning, Partial orientation.

1. Introduction

Correlations between variables are useful for prediction in the case that individuals to be predicted come from the same population as the training data. If we want to predict them after the system is manipulated by external interventions, prediction models based only on correlations may lead to awful results. For example, there is a strong correlation between a rooster's crying and sun rising. But killing the rooster cannot stop sun rising. No matter how advanced techniques and models are used based only on correlations, there may always exist some cases of external interventions which make the prediction inaccurate without causal discovery. Causal discovery is one of most important goals in various sciences, such as natural and social sciences (Pearl, 2000; Spirtes et al., 2000). In causal discovery, a key issue is to discover causes of a target feature of interest, whose main causes are generally not too many. Generally, it is difficult to discover causes and effects only from observational data, and even harder to distinguish causes from effects. Discovering causal structures and further distinguishing causes from effects of a target are useful not only for prediction in the cases with external interventions, but also valuable for studying causal mechanisms, making decision and evaluating treatment effects.

Most of the traditional prediction approaches are based on correlations without causal discovery. For example, it is well known that for a Bayesian network a Markov blanket (MB) of a target variable is often used for prediction of the target because the target is independent of other variables conditionally on the Markov blanket. A Bayesian network is called a causal network if directed edges have causal interpretation. The causation challenge organized by Guyon et al. (Guyon et al., 2008) for IEEE WCCI2008 is to predict the effect of external interventions. When the neighbor nodes of the target in the causal network are manipulated by external interventions, we have to distinguish parent nodes (cause features) from children nodes (effect features), and then we use parent nodes (and unmanipulated children nodes if we know) to predict the target. Although there are many structural learning approaches for discovering a global network, it is well known that learning a global network is an NP-Hard problem. If we are only interested in a prediction of a target, it is inefficient and unnecessary to learn a global network.

For a prediction problem with external interventions, we propose in this paper two partial orientation and local structural learning (POLSL) approaches, Local-Graph and PCD-by-PCD (PCD means Parents, Children and some Descendants). In the POLSL approaches, we discover locally the edges connected to the target and only try to orient these edges so that we can distinguish the parents from the children of the target. We can theoretically show that the approaches can correctly obtain the edges connected to the target and their orientations. The POLSL approaches can greatly reduce computational complexity of structural learning, and their statistical test is more powerful than a global learning approach. After we select a subset of all variables according to the local structure, we use the L1 penalized logistic regression model to fit the prediction model and use the estimated conditional probability of the target variable for each individual in the test set for its classification. The L1 penalized approach is a shrinkage method which can reduce mean squared error (MSE) of prediction.

In Section 2, we describe the preprocessing and we propose a two-stage filter. In Section 3, we propose two POLSL algorithms and theoretically show their correctness. In Section 4, we use the L1 penalized logistic regression model to fit the prediction model. In Section 5, we show results of simulation and the causal challenge. Advantages of our approaches are discussed in Section 6. Details of the preprocessing are described in Appendix A, and the proofs of theorems are presented in Appendix B.

2. Preprocessing

In this section, we propose a two-stage process for filtering noise in microarray data, and we use a feature screen method to remove unnecessary features for the prediction.

2.1 A two-stage filter

For the case of observed data with noise and calibrant features (e.g., MARTI), we first centralize observations and then filter noise using a two-stage process. At the first stage, we correct the global noise pattern. Then we treat every micro-array data separately to get a smoother output in the second stage. More details on this two-stage filter can be found in Appendix A.

After we build the model for prediction with the corrected training data, given a new microarray with noise for predicting its target feature, we first correct it with the global regression models obtained at the first stage, then filter the noise of every feature with the local models obtained at the second stage, and finally predict its target based on the corrected features $\{\hat{r}_0^{(i)}, i = 1, \ldots, 999\}$ and a prediction model discussed in Section 4.

2.2 Feature screen and discretization

For a data set with very high dimensional space (e.g., 4932 features for SIDO), we first screen features using a sure independence screening (SIS) procedure (Fan and Lv, 2008) to reduce the dimensionality to a tractable size (e.g., 1000 features for SIDO). The SIS method is a screening method based on correlation learning which has the property that all the important variables survive after variable screening with probability tending to one.

This screen step is not necessary for other data sets, and even for a higher dimensional data set if CPU time for the following computations is not a problem.

For continuous variables, we suppose that they have a normal distribution, or we first discretize them using the supervised discretization process in the causal explorer (Aliferis et al., 2003), and suppose that the discretized variables have a multinomial distribution.

3. Partial orientation and local structural learning

After finding a Markov blanket of a target, we can obtain edges connected to a target of interest, but it is not sufficient to orient the edges connected to the target using only the variables in the Markov blanket. In this section we propose two approaches for local structural learning and partial orientation of the edges connected to the target. Let $PC(X)$ denote a set which contains all parents and children of node X, and let $PCD(X)$ denote a set which contains $PC(X)$ and may contain some descendants of X. There are a lot of algorithms which can be used to find $PCD(X)$, such as Min Max Parents and Children (MMPC) algorithm (Tsamardinos et al., 2006).

3.1 Two Algorithms: Local-Graph and PCD-by-PCD

The first approach called Local-Graph tries to find a variable set such that all v-structures connected to a target T of interest can be discovered correctly. It first finds $PCD(T)$ and then finds $PCD(X)$ for all $X \in PCD(T)$. Let $V = \{T\} \cup PCD(T) \cup [\cup_{X \in PCD(T)} PCD(X)]$. Finally it learns a directed acyclic graph (DAG) over the node set V calling an algorithm. The recursive algorithm (Xie and Geng, 2008) was used in our algorithm Local-Graph, which recursively decomposes structural learning of a large network into local learning of several small networks.

```
Algorithm: Local-Graph (Data $D$; Target $T$)
(a) $V = \{T\} \cup PCD(T) \cup [\cup_{X \in PCD(T)} PCD(X)]$.
(b) Construct a DAG over $V$ with the recursive algorithm.
(c) Return the partially oriented local structure around T.
```

We can show below that algorithm Local-Graph can discover all v-structures connected to the target T even if the local graph returned from Local-Graph may not be a correct subgraph of the underlying DAG.

Theorem 1 *Suppose that a causal network is faithful to a probability distribution and that independence test is correctly performed by using data. Algorithm Local-Graph can correctly discover all edges and v-structures connected to a target T of interest.* ■

In the second approach called PCD-by-PCD, we extend Algorithm Local-Graph and find PCDs sequentially. In the algorithm PCD-by-PCD, we first find $PCD(T)$ of the target T and $PCD(X)$ for feature $X \in PCD(T)$, and then we sequentially find $PCD(X)$ for a feature X which is contained in the previous PCD's. During the sequential process, we find local v-structures and try to orient the edges connected to the target T as much as possible. When all of the edges connected to the target T are oriented, we stop the process and obtain all direct causes and effects of the target T. There may be some undirected edges which cannot be oriented even after we have found the PCD for every feature in the full set U of all features.

These undirected edges may have different directions in DAGs of the Markov equivalent class. Theoretically we can show that the PCD-by-PCD algorithm is correct, that is, it can correctly find, at each step, edges and local v-structures of the global DAG. Let $A||B$ denote an operation adding the list B to the tail of the list A. For example, $[1,3,5]||[2,4] = [1,3,5,2,4]$ which is an ordinal sequence.

```
Algorithm: PCD-by-PCD (Data $D$; Target $T$)
1. Initialization:
Set $canV = PCD(T)$. ($canV$ is an ordinal waiting list whose PCD will be found)
Set $V = \{T\}$. ($V$ is a set of variables whose PCD has been obtained)
2. Repeat
(a) Take $X$ from the head of the list $canV$.
(b) Get $PCD_X = PCD(X)$.
(c) $V = V \cup \{X\}$.
(d) For each $Z \in (V \cap PCD_X)$, create an undirected edge $(X,Z)$ if $Z \in PCD_X$
and $X \in PCD_Z$.
(e) Within $V$, discover possible v-structures only for the triple of $X$ and other
two variables in $V$ if an intermediate node is not in the separator set of two
nonadjacent nodes.
(f) If we find new v-structures, orient other edges between nodes in $V$ if each opposite
of them creates either a directed cycle or a new $v$-structure (Meek, 1995).
(g) $canV = canV||(PCD_X \setminus V)$. (Add new variables to the tail of the waiting list)
Until (1) all edges connecting $T$ are oriented, or (2) $canV = \emptyset$, or (3) $V = U$.
3. Return The partially oriented local structure around T.
```

Theorem 2 *Suppose that a causal network is faithful to a probability distribution and that independence test is correctly performed by using data. Then algorithm PCD-by-PCD correctly obtains edges connected to the target T, and further it returns the same orientations of these edges as a partially directed graph for the Markov equivalence class of the underlying global causal network.* ■

Algorithm PCD-by-PCD sequentially finds $PCD(X)$ of node X that is nearest to the target T among all nodes whose PCDs have not been found at the present step, and it finds $PCD(X)$ at most once for each node X. Thus its computational complexity depends on the algorithm for finding $PCD(X)$. If the number of nodes in the full set U is too large to find all PCDs, then we can stop the algorithm by limiting the maximum size of the set V. The likelihood ratio test statistic G^2 is used in our algorithms for testing conditional independencies.

3.2 Comparison between algorithms

There are several other approaches which can be used for local structural learning. One is the MB-based approach in which we first find the MB of the target and then learn the local structure over the MB and the target. Another is the Markov Blanket Fan Search (MBFS) algorithm proposed by Ramsey (2006). Below we use examples to make comparisons of the Local-Graph, PCD-by-PCD, MB-based and MBFS algorithms.

Example 1. We use the underlying causal network in Figure 1 (a) to compare the MB-based and Local-Graph algorithms. The local structures obtained from the MB-based and Local-Graph algorithms are shown in Figure 1 (b) and (c) respectively. The dashed line between nodes 1 and 7 in Figure 1 (b) denotes the edge which may be false. It can be seen that the MB-based algorithm cannot orient the v-structure $7 \rightarrow T \leftarrow 1$.

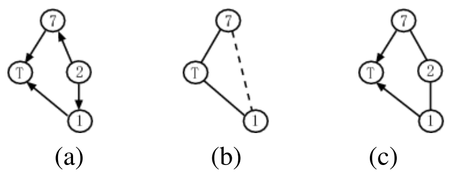


Figure 1: Comparison between the MB-based and Local-Graph algorithms.

Example 2. The underlying causal network in Figure 2 (a) is used to compare the Local-Graph and MBFS algorithms. The local structures in Figure 2 (b) and (c) are obtained from the Local-Graph and MBFS algorithms respectively. The dashed lines in Figure 2 (b) and (c) denote the edges which may be false. For example, the dashed lines $(2,4)$ and $(3,4)$ in Figure 2 (b) are determined a true edge and a false edge at the later step respectively, see Figure 2 (c). Although the v-structure $7 \rightarrow T \leftarrow 1$ is obtained from the Local-Graph algorithm, it cannot orient the undirected edge $T-2$. The MBFS algorithm can correctly orient the edge as $T \leftarrow 2$.

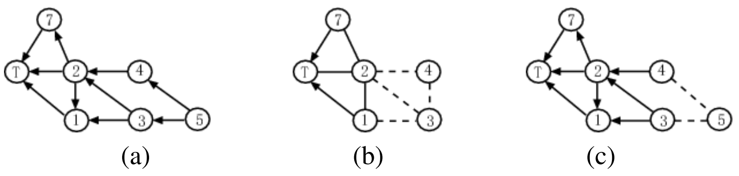


Figure 2: Comparison between the Local-Graph and MBFS algorithms.

Example 3. For the underlying causal network in Figure 3 (a), the MBFS and PCD-by-PCD algorithms output the local structures in Figure 3 (b) and (c) respectively. The MBFS algorithm cannot orient the undirected edge $T-2$, while the PCD-by-PCD algorithm can do that correctly. The dashed lines in Figure 3 have a similar meaning to those in Example 2.

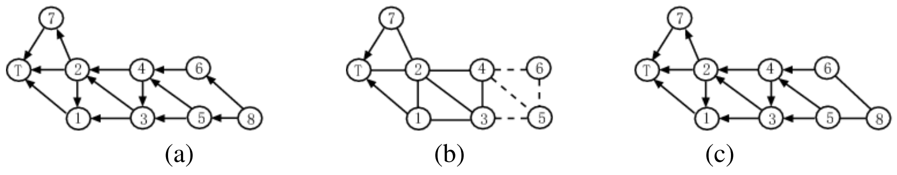


Figure 3: Comparison between the MBFS and PCD-by-PCD algorithms.

From the above examples, we can see that these algorithms discover local structures over different neighbor areas. The MB-based algorithm tries the smallest neighbor area, the Local-graph one the second smallest, the MBFS one the third, and the PCD-by-PCD one extends the neighbor area continuously until all edges connected to the target are oriented or the neighbor area has been extended to all variables. Thus the MB-based, Local-Graph, MBFS and PCD-by-PCD algorithms become in turn to be more complete in terms of orientations.

3.3 Computational complexity of the algorithms

From the previous subsection, it can be seen that all these algorithms need to find PCDs. Thus the number $\#PCD$ of times of finding PCDs can be used as the computational complexity of an algorithm. Let K denote the maximum size of PCDs for all nodes. For the MB-based algorithm, we need to find $\#PCD = O(K)$ PCDs to obtain the MB of the target and then we find a local structure over the MB. For the Local-Graph algorithm, we also need to find $\#PCD = O(K)$ PCDs to obtain the set V and then we find a local structure over V. For the MBFS algorithm, we find $\#PCD = O(K^2)$ PCDs. For the PCD-by-PCD algorithm, the number $\#PCD$ depends on the underly network, which may be smaller than that of the MB-based algorithm, such as the underlying network in Figure 1 (a), or which may larger than that of the MBFS algorithm, such as the network in Figure 3 (a). When there is an undirected path connected to the target T with a length L in the process, the number $\#PCD$ is $O(K^L)$. To stop the PCD-by-PCD algorithm early for the presence of a long undirected path, we can add a stop condition (4): the size of V is larger than a given constant C. Notice that the MB-based and Local-Graph algorithms need additional computation for finding a local structure over the MB and the set V. Computational complexity of structural learning is exponential with respect to the size of a node set, although the sizes of the MB and V are generally small.

4. Prediction

We first select features based on the causal discovery results discussed in the previous section and return a single set of selected features without rank. For the cases with and without external interventions, we select different feature sets to build prediction models. For the data set without manipulation (numbered 0), all the features in the Markov blanket (MB) of a target T are used to predict the target. For the data set with a known manipulated feature set (numbered 1), we drop the manipulated variables in the children set and drop the spouses of T whose children common with T have been all dropped, and we use all parent variables and unmanipulated children and the parents of unmanipulated children in the MB of T. For the data set with an unknown manipulated variable set (numbered 2), only the parent features of the target are used. When the feature sets that are used for prediction are sensitive to significance levels and other parameters, we may use a union of these sets and then predict the target with a shrinkage method to remove the redundant features. This approach of feature selection is defensibly heuristic since it may drop useful variables in some cases.

Next we apply the L1 penalized logistic regression model with the single set of selected features to the target prediction. We use the estimated probability of the target feature for each individual in the test set for its classification. Let X denote a feature vector, and let Y denote a binary target feature of interest with mean $\mu = E(Y)$. Consider a generalized linear model (GLM) with the logit link function

$$\log \frac{\mu}{1-\mu} = \beta^{\top} x.$$

The objective function is defined as $-\log$ (likelihood function) with a penalization on the L_1-norm of coefficients, $||\beta||_1$,

$$f(\beta, \lambda) = -l(\beta) + \lambda ||\beta||_1, \qquad (1)$$

where λ is a constant. Then a L_1-regularization path algorithm (Park and Hastie, 2007) is used to minimize the objective function $f(\cdot)$ with respect to β and to find the full solution path for (1). On the solution path we select a λ value with 5-fold cross validation (CV) in the training data set which minimizes the prediction error.

5. Numerical Studies

In this section, we first evaluate POLSL algorithms via simulations and then we interpret our results of the causal challenge.

5.1 Evaluation via Simulation

We consider the toy-example: LUCAS (LUng CAncer Simple set) network as shown in Figure 1 (a) in Guyon et al. (2008). We repeatedly do 100 simulations and give average values for each case of different sample size n and significance level α. For each simulation, we draw a training data from the distribution with parameters given on the website: `http://www.causality.inf.ethz.ch/data/LUCAS.html`. The manipulated features for LUCAS1 and LUCAS2 are shown respectively in Figure 1 (b) and (c) in Guyon et al. (2008), and the manipulated features for test data are drawn randomly which are independent of their parents.

In Table 1, we show the simulation results of discovering the parent set (PA), the MB set and the children (CH) set of the target 'Lung Cancer' with Min Max Hill Climbing (MMHC), MB-based, Local-Graph and PCD-by-PCD algorithms. Feature scores (Fscores) increase with sample size n increasing and are not significantly different. The MMHC algorithm takes CPU time the most, the MB-based algorithm the second, Local-Graph the third and PCD-by-PCD algorithm the least.

In Table 2, we show test scores (Tscores) for different logistic regressions. The middle columns are for linear models, the last two columns are for logistic models with the second order interaction terms. Tscores increase with sample size n increasing and are not significantly different for test data sets labeled 0 and 1. But for test data labeled 2, Tscores based on causal knowledge (here we use the true causal structure) are higher than those without causal knowledge. The methods without/with shrinkage are not significantly different for linear models, but Tscores are quite different for models with interactions. It may because a linear model has a few of parameters, but a model with interactions has a larger number of parameters.

5.2 Results of Causal Challenge

The problems and results of feature selection and prediction for four data sets in the causal challenge are introduced by Guyon et al. (2008). We apply both of our two algorithms Local Graph and PCD-by-PCD on each of the four task data sets. The parameters used are the default value of the MMPC algorithm in Causal Explorer toolkit (Aliferis et al., 2003). Our results are shown in Figure 4. We just select a single unsorted feature subset (ulist) without ranking features (slist), and we submitted a single set of predictions based on the ulist for each test data set. We focused on causal discovery and we tried to minimize the number of features (ulist) selected for prediction. Using the POLSL approaches, we discover a small number of important features which can dominate main causal relationships with a target of interest. As shown in

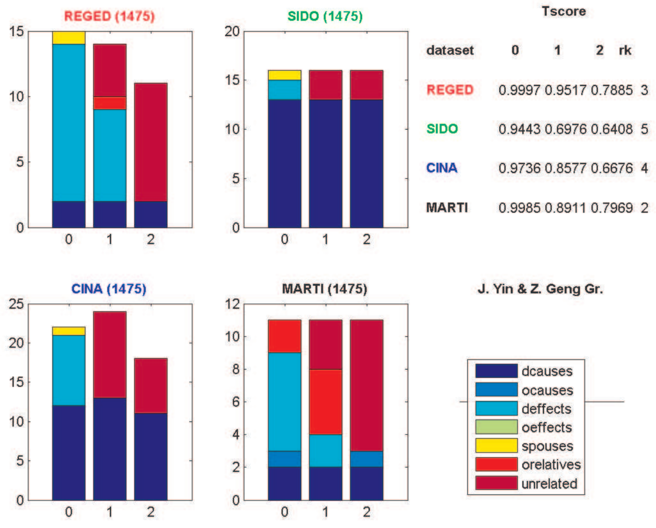
Table 1: Feature selection comparison with Fscore (Mean ± std); the unit of CPU time is second. This is a simulation study on the LUCAS data set.

n	α	Set / Time	MMHC	MB-based	Local-Graph	PCD-by-PCD
100	.05	PA	.657 ± .169	.723 ± .145	.728 ± .137	.702 ± .160
		MB	.764 ± .103	.764 ± .077	.781 ± .087	.742 ± .089
		CH	.688 ± .125	.681 ± .092	.688 ± .121	.671 ± .114
		CPU time	67.4	41.4	27.0	7.54
	.10	PA	.668 ± .162	.712 ± .153	.729 ± .140	.740 ± .167
		MB	.774 ± .099	.781 ± .077	.775 ± .094	.773 ± .088
		CH	.686 ± .120	.675 ± .107	.662 ± .134	.676 ± .124
		CPU time	71.5	51.6	35.0	8.10
200	.05	PA	.823 ± .105	.847 ± .098	.825 ± .095	.854 ± .122
		MB	.870 ± .074	.872 ± .062	.873 ± .082	.827 ± .064
		CH	.621 ± .094	.605 ± .066	.657 ± .122	.637 ± .070
		CPU time	75.2	59.5	38.6	8.52
	.10	PA	.831 ± .099	.844 ± .095	.806 ± .093	.871 ± .111
		MB	.876 ± .071	.873 ± .064	.869 ± .091	.821 ± .066
		CH	.626 ± .101	.606 ± .075	.678 ± .133	.657 ± .095
		CPU time	79.1	66.5	47.7	8.39
500	.05	PA	.863 ± .033	.870 ± .029	.832 ± .047	.921 ± .032
		MB	.927 ± .063	.930 ± .050	.939 ± .070	.841 ± .058
		CH	.676 ± .124	.665 ± .118	.743 ± .111	.707 ± .117
		CPU time	84.2	74.2	49.6	7.50
	.10	PA	.862 ± .031	.867 ± .030	.808 ± .072	.917 ± .031
		MB	.932 ± .065	.935 ± .053	.914 ± .093	.839 ± .063
		CH	.685 ± .127	.677 ± .122	.738 ± .105	.727 ± .125
		CPU time	88.7	79.4	61.7	8.13

Table 2: Prediction comparison with Tscore (Mean ± std). NC: no causal knowledge ; Cause: using causal knowledge ; Full: a full logistic regression model; Shrink: Using shrinkage; Interaction: model with interactions. This is a simulation study on the LUCAS data set.

		Tscore without / with causal knowledge				Regression with interactions	
n	Dataset	NC-Full	NC-Shrink	Cause-Full	Cause-Shrink	NC-Full	Cause-Shrink
100	0	.873 ± .028	.876 ± .035	.895 ± .025	.886 ± .031	.799 ± .039	.861 ± .036
	1	.856 ± .044	.868 ± .045	.903 ± .023	.896 ± .028	.765 ± .054	.829 ± .064
	2	.747 ± .072	.725 ± .078	.857 ± .012	.838 ± .075	.659 ± .057	.695 ± .076
200	0	.893 ± .032	.894 ± .033	.895 ± .025	.887 ± .030	.794 ± .041	.874 ± .051
	1	.888 ± .034	.893 ± .033	.903 ± .023	.892 ± .029	.763 ± .061	.853 ± .065
	2	.774 ± .064	.760 ± .072	.857 ± .012	.840 ± .063	.656 ± .058	.741 ± .063
500	0	.911 ± .013	.912 ± .013	.895 ± .025	.884 ± .035	.863 ± .019	.889 ± .032
	1	.910 ± .012	.912 ± .013	.903 ± .023	.894 ± .026	.820 ± .046	.874 ± .055
	2	.795 ± .032	.788 ± .044	.857 ± .012	.843 ± .059	.703 ± .060	.753 ± .066

Figure 4, we selected 15 features from 999 features for REGED, 11 from 999 for MARTI, 16 from 4932 for SIDO and 24 from 132 for CINA. There are only two direct causes of the target in the underlying causal graphs of REGED and MARTI, and both of them are contained in our feature sets, see the histograms of REGED and MARTI in Figure 4. Also for CINA and SIDO, a large proportion of our features are direct causes; especially for SIDO, 13 direct causes are contained in our set of 15 features. The Tscore and the rank (rk) of our results in Figure 4 may be improved by chance by using a slist of ranked features because the best Tscore over all feature set sizes is retained under the rules of the challenge. This can be seen in Figure 2 (a) in Guyon et al. (2008) that relative Tscore of our results are not the best comparing with the Tscore which is the best over nested feature subsets. However, under the rule of pairwise comparison using the same number of features, our Fscore and Tscore are at the Pareto front, as shown in Figure 2 (b) in Guyon et al. (2008). All of our computations are performed on a computer with CPU 3.0GHz and 2.49 GB RAM. The CPU times for the four data sets are shown in Table 3. Note that the preprocess time for REGED and CINA is long enough, which is mainly due to the discretization method (Aliferis et al., 2003). And the additional requirement of preprocessing time for MARTI is due to the two-stage filter. SIDO needs a relatively shorter preprocess time because the SIS process is simply a correlation computing process.



Tscore

dataset	0	1	2	rk
REGED	0.9997	0.9517	0.7885	3
SIDO	0.9443	0.6976	0.6408	5
CINA	0.9736	0.8577	0.6676	4
MARTI	0.9985	0.8911	0.7969	2

Figure 4: Profile of features selected. Legend: dcause=direct cause, deffect=direct effects, ocauses=other causes (indirect), oeffects=other effects (indirect), spouses=parent of a direct effect, orelatives=other relatives, unrelated=completely irrelevant.

6. Discussion

For discovering causal and effect features of a target, the POLSL approaches proposed in this paper only try to find the local structure near a target but not to find the whole network, thus they can greatly reduce computational complexity of structural learning. The POLSL approaches are efficient for large causal networks if we are interested only in prediction of a target. We can theoretically show that the approaches can correctly obtain the edges connected to the target and

Table 3: CPU times for our results.

Data set	Preprocessing	Structure Learning	Prediction
REGED	12 hours	15 minutes	5 minutes
SIDO	2 minutes	3 hours	10 minutes
CINA	14 hours	16 hours	10 minutes
MARTI	24 hours	15 minutes	5 minutes

their orientations. Although the Markov blanket of a target is useful for predicting the target without manipulation, it cannot be used for prediction with manipulation, and the MB-based algorithm is incomplete in terms of orientations of the edges connected to the target.

Acknowledgments

We would like to thank the five reviewers for their valuable comments and suggestions. We would appreciate I. Guyon and the competition committee for their encouragement and support to our work. We also thank Xianchao Xie for providing us with his software of structural learning. This research was supported by NSFC 10771007, NBRP 2003CB715900, 863 Project of China (2007AA01Z437), MSRA and MOE-Microsoft Key Laboratory of Statistics and Information Technology of Peking University.

Appendix A.

In this appendix we describe the filtering process in details.

First stage of the filtering process

We use a regression model for each of gene expression features, in which calibrant features are treated as explanatory variables and a gene expression as a response variable. For the jth observation (microarray), let $x_j^{(i)}$ denote a centralized observed value of the ith feature, $s_j^{(i)}$ the latent true value of the ith feature and $\epsilon_j^{(i)}$ the noise. Suppose

$$x_j^{(i)} = s_j^{(i)} + \epsilon_j^{(i)}, \tag{2}$$

for $i = 1,\ldots,F$ ($F = 999$ for MARTI) and $j = 1,\ldots,n$ ($n = 500$ for MARTI), where $s_j^{(i)}$ is independent of $\epsilon_j^{(i)}$. To remove noise, some calibrant features spread regularly across the microarray and they have mean zero. Let y_{kj} denote the kth calibrate feature of the jth observation for $k = 1,\ldots,c$ ($c = 25$ for MARTI) and $Y_j = (y_{1j},\ldots,y_{cj})$. We assume that the noise at spot i has the following model related to noise at c calibrant spots

$$\epsilon_j^{(i)} = f^{(i)}(\beta^{(i)}, Y_j) + e_j^{(i)}, \tag{3}$$

where $f^{(i)}(\cdot)$ is a known function (we used a linear one), $\beta^{(i)}$ is an unknown parameter vector and $e_j^{(i)}$ is a residue with mean zero which is independent of Y_j. From (2) and (3) we have

$$x_j^{(i)} = f^{(i)}(\beta^{(i)}, Y_j) + (e_j^{(i)} + s_j^{(i)}).$$

We treat $e_j^{(i)} + s_j^{(i)}$ as an error with mean 0 which is independent of Y. Using the least squares method, we can get estimates $\hat{\beta}^{(i)}$, $\hat{\epsilon}_j^{(i)} = f^{(i)}(\hat{\beta}^{(i)}, Y_j)$, and $\hat{s}_j^{(i)} = x_j^{(i)} - \hat{\epsilon}_j^{(i)}$.

Second stage of the filtering process
We treat each microarray separately and thus we omit subscript j. We locally filter the residual noise of each corrected feature $\hat{s}^{(i)}$ using features near the spot i. Suppose that the model for the ith corrected feature is

$$\hat{s}^{(i)} = r^{(i)} + \eta^{(i)}, \tag{4}$$

where $r^{(i)}$ and $\eta^{(i)}$ denote the true latent value and the residual noise respectively, and $r^{(i)}$ is independent of $\eta^{(i)}$. Let $Z^{(j)} = (z_1^{(j)}, z_2^{(j)})$ denote the geometric coordinate of spot j relative to the origin spot i, which is a pair of integers. Define the neighbor area of spot i as $\Omega^{(i)} = \{j : j \neq i, |Z^{(j)} - Z^{(i)}| \leq L\}$ where $|\cdot|$ denotes a distance and L is the upper bound (a user chosen constant) of the distance between spot i and any spot j in the neighbor area. Assume that $\eta^{(j)}$ has a polynomial surface in the neighbor area

$$\eta^{(j)} = g^{(i)}(\alpha^{(i)}, Z^{(j)}) + \xi^{(j)} \tag{5}$$

for $j \in \Omega^{(i)} \cup \{i\}$, where $g^{(i)}$ is a known function (we used a quadratic one), $\alpha^{(i)}$ is an unknown parameter vector, and $\xi^{(j)}$ is an error term with mean zero. From (4) and (5) we have

$$\hat{s}^{(j)} = g^{(i)}(\alpha^{(i)}, Z^{(j)}) + (r^{(j)} + \xi^{(j)})$$

for $j \in \Omega^{(i)}$. Treating $(r^{(j)} + \xi^{(j)})$ as an error term with mean zero, we first find the model and remove 'outliers' to keep informative signals of features. Then using estimates $\hat{\alpha}^{(i)}$, we obtain $\hat{\eta}^{(i)}$ by (4), and finally we get $\hat{r}^{(i)}$ by (3), which is the estimate of the ith feature to be used for prediction modeling.

Appendix B.

In this appendix we prove theorems presented in Section 3.
Proof of Theorem 1. Define $W = \{T\} \cup PCD(T)$. In algorithm Local-Graph, $PCD(X)$ is obtained for each $X \in W$, and V contains all of them. For two nodes u and v, either u is not a descendant of v or v is not a descendant of u. A node is d-separated from its non-descendant by its parent set. Then two nodes u and v in W are not adjacent if and only if they are d-separated by a subset S_{uv} of V. Thus algorithm Local-Graph can correctly find all edges between nodes in W. If there is a pattern $u - T - v$ and T is not contained in the separator S_{uv}, then we can discover a v-stricture $u \rightarrow T \leftarrow v$. Thus we have proven Theorem 1.
Proof of Theorem 2. In the PCD-by-PCD algorithm, we find $PCD(T)$ and set $canV = PCD(T)$ where $canV$ denotes a list of nodes whose PCDs will be found at the latter steps.

At step 2 we repeatedly find $PCD(X)$ for X in $canV$ at step 2 (b). Let V denote the set of variables whose PCD has been found. Suppose that the algorithm for finding $PCD(X)$ is correct, such as the algorithm MMPC (Tsamardinos et al., 2006).

At step 2 (d), we can correctly obtain an undirected edge $X - Z$ if we have that both $Z \in PCD(X)$ and $X \in PCD(Z)$. For both Z and X in V, we have obtained $PCD(Z)$ and $PCD(X)$. At step 2 (d), we only need to treat $Z \in (V \cap PCD_X)$ since $Z \notin PCD(X)$ implies no edge $X - Z$, and every other pair of Z and Z' contained in V has been treated at the previous step 2 (d) when Z' or Z entered in V.

At step 2 (e), we try to discover v-structures which contain X as a node since all the undirected edges obtained newly at step 2 (d) contain X and other v-structures without X have been discovered at the previous step 2 (e).

At step 2 (f), we try to orient undirected edges via v-structures obtained newly at step 2 (e).

At step 2 (g), we add nodes of $PCD(X)$ to the end of $canV$.

Finally, we discuss the stop rule. The condition (1) means that all edges connecting T have been oriented and thus the algorithm can stop. The condition (2) means that there is no more node whose PCD needs to be found, which implies other nodes disconnecting T. The condition (3) means that we have found PCDs for all nodes and thus we cannot orient some edges connecting T. If the algorithm stops by the condition (3), we have found the global skeleton graph of the underlying causal network and all v-structures, and thus we obtained the Markov equivalence class. If the algorithm stops by the condition (2), then the underlying causal network is not connected, and we have found the skeleton graph and all v-structures in the connected component. ■

References

C. Aliferis, I. Tsamardinos and A. Statnikov. *'Causal Explorer: A Probabilistic Network Learning Toolkit for Biomedical Discovery.'* The 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences METMBS'03, June 23-26, 2003.

J. Fan and J. Lv. Sure independence screening for ultra-high dimensional feature space. To appear in *J. R. Statist. Soc.* B 70, 849-911, 2008.

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J. Pellet, P. Spirtes and A. Statnikov. Design and analysis of the causation and prediction challenge, in: JMLR: Workshop and conference Proceedings 1-16, 2008.

S. Lauritzen. *Graphical Models*. Clarendon Press, London, 1996.

C. Meek. Causal inference and causal explanation with background knowledge, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 403-410, Morgan Kaufmann, San Francisco, 1995.

M. Y. Park and T. Hastie. L_1-regularization path algorithm for generalized linear models. *J. R. Statist. Soc.* B 69, 659-677, 2007.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

J. Ramsey. A PC-style Markov blanket search for high dimensional datasets. Technical Report No. CMU-PHIL-177, 2006.

P. Spirtes, C. Glymour and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, the second edition, 2000.

I. Tsamardinos, L. Brown and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65, 31-78, 2006.

X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *J Machine Learning Research*, 9, 459-483, 2008.

Causal & Non-Causal Feature Selection for Ridge Regression

Gavin C. Cawley GCC@CMP.UEA.AC.UK
School of Computing Sciences
University of East Anglia
Norwich, Norfolk, NR4 7TJ, United Kingdom

Editor: I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

Abstract

In this paper we investigate the use of causal and non-causal feature selection methods for linear classifiers in situations where the causal relationships between the input and response variables may differ between the training and operational data. The causal feature selection methods investigated include inference of the Markov Blanket and inference of direct causes and of direct effects. The non-causal feature selection method is based on logistic regression with Bayesian regularisation using a Laplace prior. A simple ridge regression model is used as the base classifier, where the ridge parameter is efficiently tuned so as to minimise the leave-one-out error, via eigen-decomposition of the data covariance matrix. For tasks with more features than patterns, linear kernel ridge regression is used for computational efficiency. Results are presented for all of the WCCI-2008 Causation and Prediction Challenge datasets, demonstrating that, somewhat surprisingly, causal feature selection procedures do not provide significant benefits in terms of predictive accuracy over non-causal feature selection and/or classification using the entire feature set.

Keywords: regularisation, feature selection, causal inference

1. Introduction

A common assumption underpinning the majority of classical statistical pattern recognition techniques holds that the training data represent an independent and identically distributed (i.i.d.) sample drawn from the same underlying distribution as the operational or test data. Unfortunately, in many practical applications this assumption may not be valid. For example one might train a classifier to diagnose lung cancer using historical data from a particular hospital. However, through changes in referral procedures, diet and lifestyle (for example through government initiatives to restrict smoking in enclosed public places), the distribution of symptoms presented by patients may become progressively more and more different from that of the training sample; a phenomenon known as covariate shift (Quiñonero Candela et al., 2009). Nevertheless, the classifier may still be of diagnostic value, especially if designed from the outset to be robust to covariate shift, for instance through careful feature selection. It seems a reasonable assumption that features in a close causal relationship with the target are likely to remain more reliable under covariate shift than those with a more tenuous link. A model that is robust in this sense would also be valuable in predicting the effects of interventions, for instance in planning more effective referral procedures. Therefore there are practical reasons for attempting to infer the causal relationships between the explanatory and target variables in order to improve predictive performance rather than uncovering the structure of the data. In this paper we evaluate the effectiveness of several causal and non-causal feature selection procedures in such situations, using ridge regression as the base classifier, using all of the datasets comprising in the WCCI-2008 Causation and Prediction Challenge.

2. Method

Ridge regression ensembles are used as the base classifier for the empirical study. Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}$ represent the training sample, where $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ is a vector of explanatory features for the i^{th} sample, and $y_i \in \{+1, \; -1\}$ is the corresponding response indicating whether the sample belongs to the positive or negative class respectively. Ridge regression provides a simple and effective classifier that is equivalent to a form of regularised linear discriminant analysis. The output of the ridge regression classifier, $\hat{y}_i$, and vector of model parameters, $\boldsymbol{\beta} \in \mathbb{R}^d$, are given by

$$\hat{y}_i = \boldsymbol{x}_i \cdot \boldsymbol{\beta} \qquad \text{and} \qquad \left[\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right]\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}, \tag{1}$$

where $\boldsymbol{X} = [\boldsymbol{x}_i]_{i=1}^{\ell}$ is the data matrix, $\boldsymbol{y} = (y_i)_{i=1}^{\ell}$ is the response vector and the ridge parameter, λ, controls the bias-variance trade-off (Geman et al., 1992). Note that classifiers used throughout this study included an unregularised bias parameter, which has been neglected here for notational convenience. Careful tuning of the ridge parameter allows the ridge regression classifier to be used even in situations with many more features than training patterns (i.e. $d \gg \ell$) without significant over-fitting (e.g. Cawley, 2006). Fortunately the ridge parameter can be optimised efficiently by minimising a closed-form leave-one-out cross-validation estimate of the sum of squared errors, i.e. Allen's PRESS statistic (Allen, 1974),

$$P(\lambda) = \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\hat{y}_i^{(-i)} - y_i\right]^2 \qquad \text{where} \qquad \hat{y}_i^{(-i)} - y_i = \frac{\hat{y}_i - y_i}{1 - h_{ii}}, \tag{2}$$

$\hat{y}_i^{(-i)}$ represents the output of the classifier for the i^{th} training pattern in the i^{th} fold of the leave-one-out procedure and h_{ii} is an element of the principal diagonal of the hat matrix $\boldsymbol{H} = \boldsymbol{X}\left[\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right]^{-1}\boldsymbol{X}^T$. The ridge parameter can be optimised more efficiently in canonical form (Weisberg, 1985) via eigen-decomposition of the data covariance matrix $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}^T\boldsymbol{\Lambda}\boldsymbol{V}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues. The normal equations and hat matrix can then be written as

$$[\boldsymbol{\Lambda} + \lambda\boldsymbol{I}]\,\boldsymbol{\alpha} = \boldsymbol{V}^T\boldsymbol{X}^T\boldsymbol{y} \quad \text{where} \quad \boldsymbol{\alpha} = \boldsymbol{V}^T\boldsymbol{\beta} \qquad \text{and} \qquad \boldsymbol{H} = \boldsymbol{V}\,[\boldsymbol{\Lambda} + \lambda\boldsymbol{I}]^{-1}\,\boldsymbol{V}^T \tag{3}$$

As only a diagonal rather than a full matrix need now be inverted following a change in λ, the computational expense of optimising the ridge parameter is greatly reduced. For problems with more features than training patterns, $d > \ell$, the kernel ridge regression classifier (Saunders et al., 1998) with a linear kernel is more efficient and exactly equivalent. The ridge parameter for KRR can also be optimised efficiently via an eigen-decomposition of the kernel matrix (Saadi et al., 2007).

2.1 Non-Causal Feature Selection

The feature selection methods most frequently used in practical applications aim to determine a small subset of features that are predictive of the target variable, without any consideration of causal relationships. For a survey of conventional feature selection methods, see Guyon and Eliseeff (2003). In this study, we adopt an embedded feature selection method, known as BLogReg (Cawley and Talbot, 2006), based on logistic regression with Bayesian regularisation using a Laplace prior. As the usual regularisation parameter is integrated out analytically in this approach using an uninformative hyper-prior, the number of features is determined automatically, without the need for additional cross-validation. Rather than make predictions with BLogReg directly, it is used to select features for a ridge regression model so that the comparison of feature selection techniques is not obscured by the differences due to the classifier.

2.2 Finding the Markov Blanket

The most basic form of causal feature selection aims to determine the Markov blanket, $MB(T)$, the set of features such that the target, T, is conditionally independent of all other features, conditioned on the features comprising $MB(T)$. Under the faithfulness assumption (Pearl, 1988), the Markov blanket consists of the set of features representing the direct causes (parents) and direct consequences (children) of the target, and also any other causes directly affecting the consequences of the target (spouses). In this study, we use the HITON algorithm (Aliferis et al., 2003) to infer the Markov blanket of the *unmanipulated* distribution throughout. If information is available regarding which features have been manipulated, it is in principle possible to infer the Markov blanket of the *manipulated* distribution, however this was not investigated (due to the ignorance of the investigator at the time!). Once the Markov blanket of the unmanipulated distribution has been determined, the manipulated children of the target can be deleted (provided they are not also a spouse via an unmanipulated child) as the direct causal link has been broken and similarly spouses related only via manipulated children can also be deleted, forming the Markov blanket of the manipulated distribution.

2.3 Discerning Causes and Effects

More advanced causal inference methods attempt to construct a directed graph representing the causal relationships between variables. This can be used to identify direct causes and direct effects of the target variable, forming alternative feature sets for ridge regression classifiers. In this study, the PC and MMHC algorithms of the Causal Explorer package (for further details, see Aliferis et al., 2003) were used throughout. As these algorithms are computationally expensive, they are applied to the subset of features already identified as belonging to the Markov blanket. As the PC algorithm can accommodate problems with continuous attributes, no discretisation of continuous features was necessary.

2.4 Use of Ensembles

As feature selection algorithms are unstable, i.e. a perturbation of the data is likely to result in a different subset of features being selected, an ensemble of 100 ridge regression classifiers is used in all experiments to minimise the distracting effects of this source of variability. For each component classifier, a different random partition of the available data is used to form training and test sets (in proportions of 9:1) and feature selection performed separately for each partition. The prediction is then made using the arithmetic mean of the 100 component classifiers. As the regularisation parameter is also tuned separately for each of the component classifiers, this approach also addresses over-fitting the model selection criterion so some degree (Cawley and Talbot, 2007; Hall and Robinson, 2009). It should be noted that as feature selection is performed separately for each component classifier, the number of features used by the ensemble is generally much larger than the number used by any individual member of the ensemble.

3. Results

In this section, we compare the performance of different causal and non-causal feature selection procedures for ridge regression classifiers on two illustrative benchmarks and each of the four challenge datasets, before discussing the absolute performance of the base ridge regression classifier. The results are given for a number of different models, in most cases labelled according to the feature selection process used:

- **True MB:** Ensembles trained using the true Markov blanket of the target variable for each variant of the benchmark. For REGED and MARTI, manipulated features are only retained if they are parents or spouses of the target. For CINA and SIDO, where the feature set is comprised of real variables and probes, the true Markov blanket is unknown, and hence the union of the set of all real variables and probes belonging to the true Markov blanket is used instead, for further details, see Guyon et al. (2008).

- **Inferred MB:** Features identified by HITON_MB as forming the Markov blanket of the *unmanipulated* distribution are retained.

- **None:** No feature selection is performed, relying purely on regularisation to prevent over-fitting the training data.

- **Non-causal:** Feature selection performed using the BLogReg algorithm (Cawley and Talbot, 2006), providing an example of the performance of traditional embedded feature selection methods.

- **Causes & effects:** Members of the Markov Blanket of the unmanipulated distribution, identified by HITON_MB, that are determined using the MMHC or PC algorithms to be direct causes and effects of the target variable. It should be noted that the pre-filtering to remove features not belonging to the Markov blanket of target for the unmanipulated distribution probably made it very difficult for the causal discovery algorithm to correctly orient the edges of the causal graph, and this perhaps explains the poor performance observed.

- **Causes only:** Members of the Markov Blanket of the unmanipulated distribution, identified by HITON_MB, that are determined using the MMHC or PC algorithms to be direct causes of the target variable.

- **Winner:** This is the scores achieved by the winning entries for each benchmark in the WCCI-2008 Causation and Prediction Challenge. For REGED and CINA, these are described in Chang and Lin (2008); for SIDO and MARTI, these are described here and in the supplementary material[1]. Note that the entries for REGED and CINA are not based on ridge regression ensembles and so are not directly comparable to the other methods.

- **Best TSCORE:** Results for models achieving the best TSCORE, whether or not they were the winning entries, for each variant of each benchmark during the WCCI-2008 Causation and Prediction Challenge.

- **Yin *et al.*:** Feature selection based on the method by Yin et al. (2008), which was identified as the most successful algorithm for local structure determination.

1. http://theoval.cmp.uea.ac.uk/~gcc/projects/causal

3.1 LUCAS — LUng CAncer Simple Dataset

The LUCAS dataset represents a synthetic medical diagnosis problem, where the task is to identify patients with lung cancer from a set of explanatory variables of putative causal relevance. As the data are generated artificially using a set of simple Bayesian network models (see Guyon et al., 2008, Figure 1 a-c), the true nature of the underlying causal relationships is known, and so this benchmark is useful in illustrating the value of different approaches in ideal conditions. The results obtained[2] on this benchmark are shown in Table 1.

Table 1: Results obtained for the LUCAS benchmark: FNUM – number of features used, FSCORE – area under the receiver operating characteristic (AUROC) statistic for the detection of causally related features, DSCORE – AUROC on the training set, TSCORE – AUROC on the test set, AUC – AUROC using 100-fold repeated hold-out validation.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
LUCAS0	None	11	1.0000	0.9139	0.9170	0.9079
	Inferred MB	6	1.0000	0.9102	0.9168	0.9082
	True MB	5	1.0000	0.9103	0.9167	0.9082
	Non-causal	11	0.8070	1.0000	0.9139	0.9079
	Causes & effects	4	0.9000	0.8911	0.8992	0.8910
	Causes only	2	0.7000	0.7782	0.7968	0.7832
LUCAS1	True MB	4	1.0000	0.9026	0.9041	—
	Inferred MB	6	1.0000	0.9102	0.9012	—
	None	11	1.0000	0.9139	0.9005	—
	Non-causal	11	1.0000	0.9139	0.9005	—
	Causes & effects	4	0.8571	0.8911	0.8808	—
	Causes only	2	0.7500	0.7782	0.7910	—
LUCAS2	True MB	2	1.0000	0.7782	0.7913	—
	Causes only	2	1.0000	0.7782	0.7913	—
	Causes & effects	4	0.9444	0.8911	0.7579	—
	Inferred MB	6	0.9444	0.9102	0.7410	—
	None	11	0.8333	0.9139	0.7348	—
	Non-causal	11	0.9444	0.9139	0.7342	—

Regularisation proves satisfactory in suppressing the influence of uninformative features in the absence of external manipulation (LUCAS0), and so feature selection does not improve predictive performance (although selection of the Markov blanket is only marginally inferior). In the presence of mild manipulation (LUCAS1), the benefit of selecting only the variables comprising the Markov blanket of the target becomes more apparent, achieving the best TSCORE as the manipulation of causally irrelevant variables is ignored. It is interesting to note, however, that the result obtained is only marginally better than that for a ridge regression model without any form of feature selection, showing that regularisation is effective in suppressing the influence of irrelevant variables. However other explanations are plausible. For instance, it could be that including redundant variables is better than deleting important variables (Guyon et al., 2008, §6.2); alternatively it may be the case in many applications that the most relevant features are simply those best correlated with the target. For LUCAS2, only the direct causes are rel-

2. Non-causal feature selection performed using BLogReg (tolerance = 1×10^{-9}), identification of the Markov blanket using HITON_MB ("g2" statistic, threshold = 0.05, maximum size of conditioning set = 4), identification of direct causes and effects using MMHC (with default parameter settings).

evant, and for this simple dataset the causal discovery algorithms (HITON_MB and MMHC) are effective in identifying them so the *causes only* model performs significantly better than the others.

3.2 LUCAP — LUng CAncer with Probes Dataset

The LUCAP benchmark extends the medical diagnosis problem introduced in LUCAS to include *probes*, artificial variables that are noisy functions of the existing variables (see Guyon et al., 2008, Figure 1 d-e). The results obtained[3] on this benchmark are shown in Table 2. The probes appear to obfuscate the task of discovering the true causal structure of the data, and the models with non-causal feature selection fare conspicuously better than those with causal feature selection on the manipulated datasets. The organizers suggest that selecting features that are direct causes of the target may be an attractive approach; however the causal discovery algorithm (MMHC) found 26 features that may be direct causes, over the 100 random partitions of the data, when in fact there are only two genuine direct causes. The *causes only* approach therefore performed very poorly. It is a rather discouraging result that the causal feature selection procedures perform so poorly, albeit perhaps in the hands of an inexpert user.

Table 2: Results obtained for the LUCAP benchmark, see caption of Table 1 for details.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
LUCAP0	Non-causal	42	0.5930	0.9749	0.9711	0.9681
	True MB	105	1.0000	0.9757	0.9692	0.9698
	None	143	0.7381	0.9768	0.9686	0.9695
	Inferred MB	77	0.8466	0.9726	0.9684	0.9674
	Causes & effects	45	0.7143	0.9703	0.9675	0.9664
	Causes only	26	0.6238	0.9486	0.9382	0.8089
LUCAP1	True MB	11	1.0000	0.9139	0.9126	—
	Non-causal	42	0.6832	0.9749	0.8564	—
	Causes & effects	45	0.5561	0.9703	0.8317	—
	Inferred MB	77	0.5344	0.9726	0.8121	—
	None	143	0.6109	0.9768	0.7744	—
	Causes only	26	0.5090	0.9486	0.6504	—
LUCAP2	True MB	11	1.0000	0.9139	0.9165	—
	Non-causal	42	0.6832	0.9749	0.6578	—
	Inferred MB	77	0.5344	0.9726	0.5634	—
	Causes & effects	45	0.5561	0.9703	0.5575	—
	None	143	0.6109	0.9768	0.5100	—
	Causes only	26	0.5090	0.9468	0.4344	—

3.3 REGED — REsimulated Gene Expression Dataset

The REGED dataset represents a re-simulated gene expression microarray classification problem, where the task is to diagnose lung cancer on the basis of gene expression profiles, clas-

3. BLogReg: tolerance = 1×10^{-9}, HITON_MB: “g2” statistic, threshold = 0.05, maximum size of conditioning set = 4, MMHC: default parameter settings.

sifying samples as malignant (adenocarcinoma) or benign (squamous). The results obtained[4] on this benchmark are shown in Table 3 and statistical significance diagrams (adapted from the critical difference diagrams introduced by Demšar (2006)), are shown in Figure 1:

- For all three datasets, True MB achieves a TSCORE that is statistically indistinguishable from the best obtained during the challenge (Best TSCORE), demonstrating that the linear ridge regression ensemble is a competitive base classifier for this benchmark.
- For all three datasets, the TSCORE performance obtained using the non-causal feature selection procedure (BLogReg) was statistically indistinguishable from that obtained using the best overall causal feature selection procedure (Yin *et al.*). This is perhaps because the BLogReg algorithm was originally developed with this particular application (Cawley, 2006) in mind, although the approach is generally applicable and without specific adaption to microarray classification.
- The total number of features used by the inferred Markov Blanket ensemble for REGED0 (78) is much larger than the average number of features used by the individual component classifiers (24.85), providing an indication of the instability of the causal feature selection methods. Note that the average size of the inferred Markov blanket for each component is however close to the true value (21). The average number of features used by individual component classifiers are shown in Table 8, demonstrating that a degree of instability is to be expected when using both causal and non-causal feature selection methods.
- None of the causal feature selection algorithms, with the exception of reference entries, proved statistically superior to non-causal selection procedures for any of the three REGED datasets, a rather disappointing and challenging result.

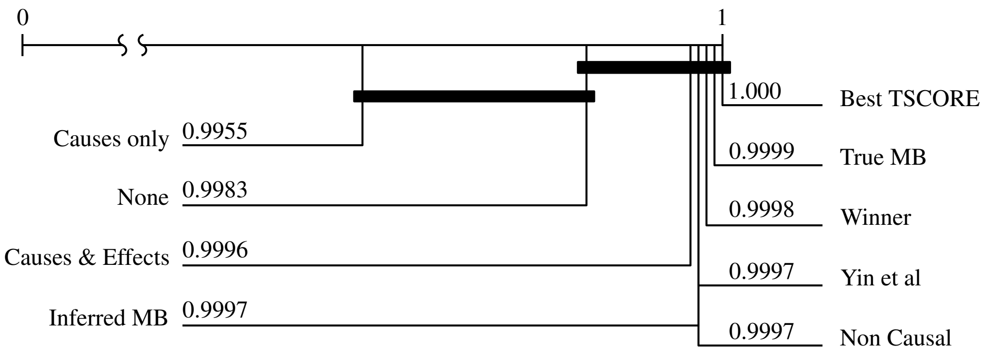
No use was made of the information regarding the manipulated variables for the methods introduced in this study; it is possible that better results might be obtained by taking advantage of this information, especially for causal feature selection approaches.

3.4 SIDO — SImple Drug Operation

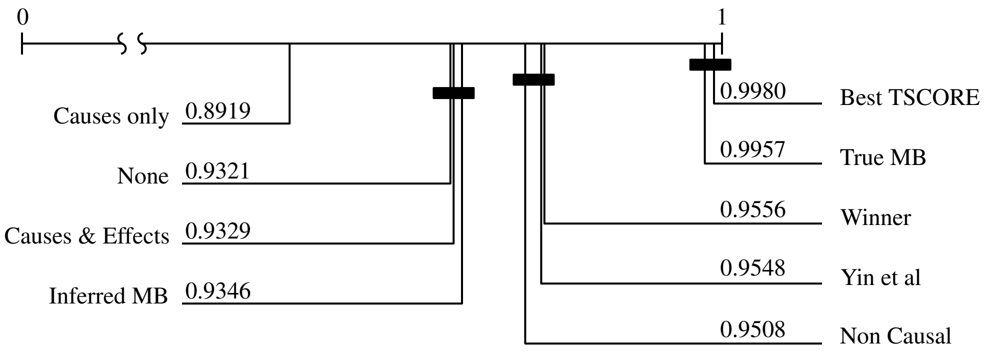
The SIDO benchmark represents a problem in pharmacology, where the task is to identify small molecules that are active against the AIDS HIV virus on the basis of a large number of binary molecular descriptors. The results obtained[5] on this benchmark are shown in Table 4 and the statistical significance diagram in Figure 2:

- Again, for all three datasets, True MB achieves a TSCORE that is statistically indistinguishable from the best obtained during the challenge (Best TSCORE), demonstrating that the linear ridge regression ensemble is a competitive base classifier for this benchmark.
- For both manipulated datasets, the TSCORE achieved using no feature selection at all is statistically superior to the best overall causal feature selection method (Yin *et al.*). For the unmanipulated dataset, the differences in performance were statistically insignificant. This demonstrates that regularisation alone can be highly effective in suppressing the deleterious influence of uninformative features.

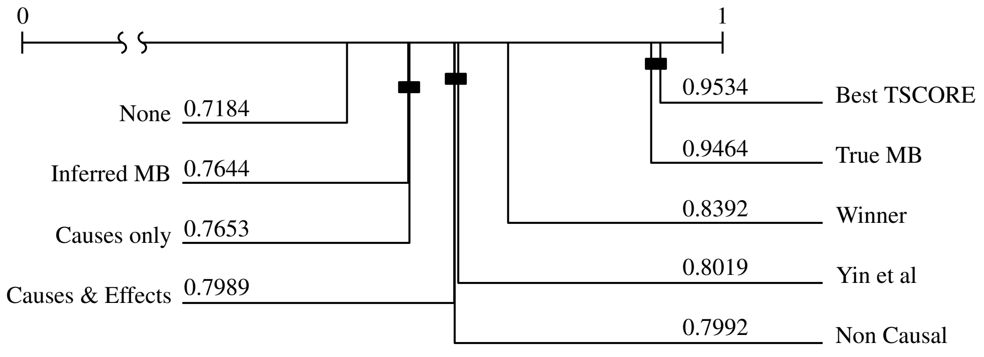
4. BLogReg: tolerance = 1×10^{-6}, HITON_MB: “z” statistic, threshold = 0.05, maximum size of conditioning set = 2, PC: ’z’ statistic, threshold = 0.05, $k = 16$.
5. BLogReg: tolerance = 1×10^{-6}, HITON_MB: “g2” statistic, threshold = 0.05, maximum size of conditioning set = 3, PC: “g2” statistic, threshold = 0.05, $k = 8$.



(a)



(b)



(c)

Figure 1: Statistical significance diagrams for (a) REGED0, (b) REGED1 and (c) REGED2. The axis represents the TSCORE statistic, and the heavy bars denote groups of classifiers with statistically indistinguishable performance. The statistical significance of differences in TSCORE are determined using the two sample z-test at the 95% level of significance (a critical value of $z = 1.64$).

- The differences in TSCORE between the best causal feature selection procedure (Yin *et al.*) and non-causal feature selection using BLogReg were statistically insignificant for all three datasets.
- None of the causal feature selection algorithms investigated, with the exception of reference entries, proved statistically superior to an ensemble trained using all of the available features for any of the three SIDO datasets, again a rather disappointing result.

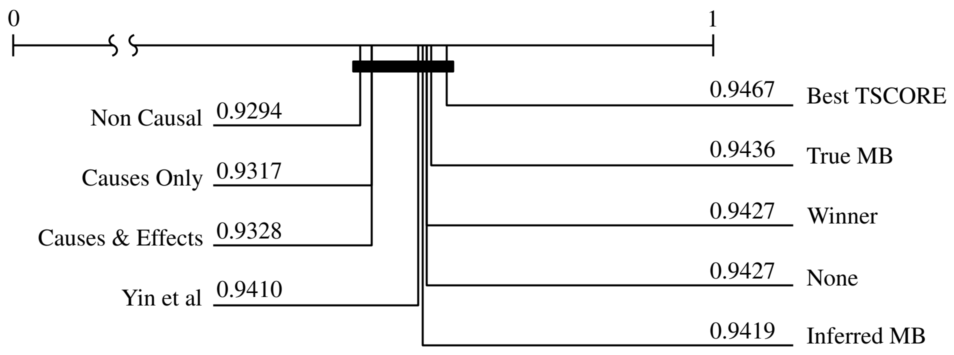
3.5 CINA — Census Is Not Adult

The CINA benchmark describes an econometrics problem, where the task is to discover the socio-economic factors affecting income (the positive class representing individuals with annual

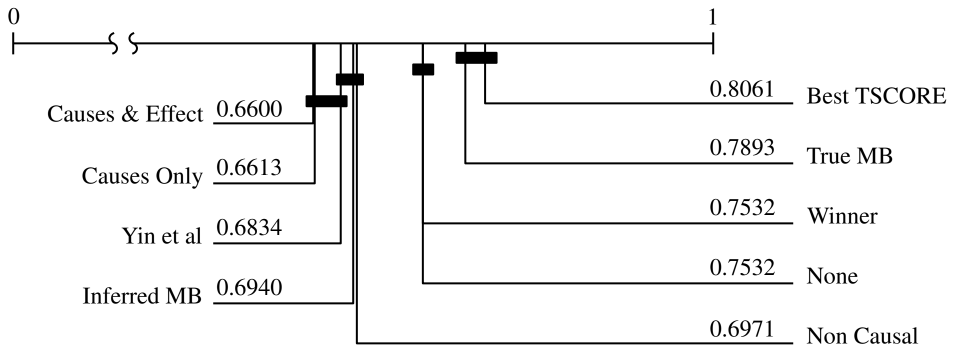
Table 3: Results obtained for the REGED benchmark, see caption of Table 1 for details.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
REGED0	Best TSCORE*	122	0.8352	1.0000	1.0000 ± 0.0002	—
	True MB	21	1.0000	0.9999	0.9999 ± 0.0008	0.9997
	Winner‡	16	0.8526	1.0000	0.9998 ± 0.0009	—
	Yin *et al.*	15	0.8571	1.0000	0.9997 ± 0.0010	0.9998
	Non-causal	26	0.8070	1.0000	0.9997 ± 0.0009	0.9997
	Inferred MB	78	0.8988	0.9999	0.9997 ± 0.0012	0.9995
	Causes & effects	13	0.8095	0.9999	0.9996 ± 0.0011	0.9996
	None	999	0.9204	1.0000	0.9983 ± 0.0017	0.9962
	Causes only	9	0.7143	0.9984	0.9955 ± 0.0018	0.8961
REGED1	Best TSCORE*	122	0.7946	1.0000	0.9980 ± 0.0015	—
	True MB	14	1.0000	0.9926	0.9957 ± 0.0020	—
	Winner‡	16	0.8566	1.0000	0.9556 ± 0.0040	—
	Yin *et al.*	14	0.8185	0.9999	0.9548 ± 0.0036	—
	Non-causal	26	0.7798	1.0000	0.9508 ± 0.0036	—
	Inferred MB	78	0.8438	0.9999	0.9346 ± 0.0044	—
	Causes & effects	13	0.7822	0.9999	0.9329 ± 0.0037	—
	None	999	0.9078	1.0000	0.9321 ± 0.0036	—
	Causes only	9	0.7124	0.9984	0.8919 ± 0.0042	—
REGED2	Best TSCORE†	2	1.0000	0.9611	0.9534 ± 0.0042	—
	True MB	2	1.0000	0.9557	0.9464 ± 0.0041	—
	Winner‡	8	0.9970	0.9995	0.8392 ± 0.0052	—
	Yin *et al.*	11	0.9975	0.9997	0.8019 ± 0.0054	—
	Non-causal	26	0.9980	1.0000	0.7992 ± 0.0056	—
	Causes & effects	13	0.9970	0.9999	0.7989 ± 0.0057	—
	Causes only	9	0.9970	0.9984	0.7653 ± 0.0054	—
	Inferred MB	78	0.9975	0.9999	0.7644 ± 0.0057	—
	None	999	0.9950	1.0000	0.7184 ± 0.0059	—

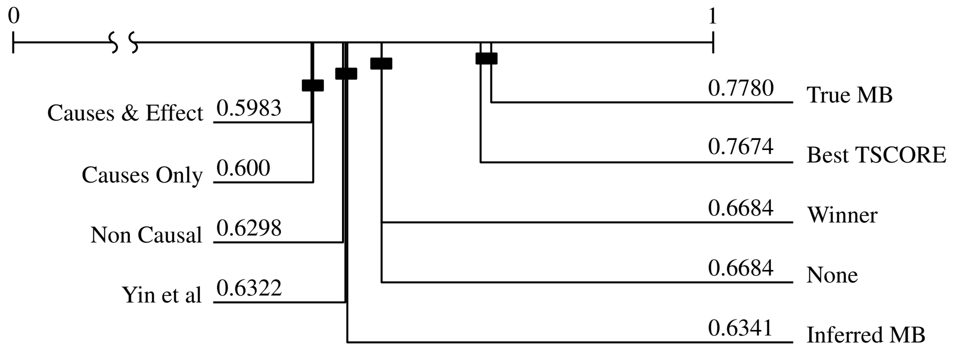
*Reference "SNB(CMA), IID assumption", †Reference "True model with parents", ‡Yin-Wen Chang "final submission".



(a)



(b)



(c)

Figure 2: Statistical significance diagrams for (a) SIDO0, (b) SIDO1 and (c) SIDO2. The axis represents the TSCORE statistic, and the heavy bars denote groups of classifiers with statistically indistinguishable performance.

Table 4: Results obtained for the SIDO benchmark, see caption of Table 1 for details.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
SIDO0	Best TSCORE†	181	0.4940	0.9584	0.9467 ± 0.0073	—
	True MB	4301	0.9995*	0.9830	0.9436 ± 0.0072	0.9471
	Winner§	4928	0.5890	0.9840	0.9427 ± 0.0070	—
	None	4928	0.5890	0.9840	0.9427 ± 0.0070	0.9472
	Inferred MB	837	0.5834	0.9563	0.9419 ± 0.0075	0.9356
	Yin *et al.*	16	0.5019	0.9475	0.9410 ± 0.0074	0.9442
	Causes & effects	58	0.5067	0.9459	0.9328 ± 0.0085	0.8798
	Causes only	58	0.5067	0.9454	0.9317 ± 0.0089	0.8733
	Non-causal	138	0.5160	0.9482	0.9294 ± 0.0080	0.9226
SIDO1	True MB	1643	0.9997*	0.9098	0.8061 ± 0.0132	—
	Best TSCORE‡	1024	0.8114	0.9021	0.7893 ± 0.0135	—
	Winner§	4928	0.5314	0.9840	0.7532 ± 0.0137	—
	None	4928	0.5314	0.9840	0.7532 ± 0.0137	—
	Non-causal	138	0.4909	0.9482	0.6971 ± 0.0138	—
	Inferred MB	873	0.5351	0.9563	0.6940 ± 0.0138	—
	Yin *et al.*	16	0.5035	0.9475	0.6834 ± 0.0133	—
	Causes only	58	0.4989	0.9454	0.6613 ± 0.0138	—
	Causes & effects	58	0.4989	0.9459	0.6600 ± 0.0137	—
SIDO2	True MB	1643	0.9997*	0.9089	0.7780 ± 0.0130	—
	Best TSCORE‡	512	0.8114	0.8693	0.7674 ± 0.0129	—
	Winner§	4928	0.5314	0.9840	0.6684 ± 0.0130	—
	None	4928	0.5314	0.9840	0.6684 ± 0.0130	—
	Inferred MB	873	0.5351	0.9563	0.6341 ± 0.0124	—
	Yin *et al.*	16	0.5035	0.9475	0.6322 ± 0.0131	—
	Non-causal	138	0.4909	0.9482	0.6298 ± 0.0039	—
	Causes only	58	0.4989	0.9545	0.6000 ± 0.0129	—
	Causes & effects	58	0.4989	0.9459	0.5983 ± 0.0129	—

*Some features beneath the Markov blanket assigned a weight of zero and was not included. †Gavin Cawley "Final #009", ‡Reference "MB_LR_S", §Gavin Cawley "final models".

income in excess of $50K). The results obtained[6] on this benchmark are shown in Table 5 and the statistical significance of differences in TSCORE are depicted in Figure 3:

- True MB achieves a TSCORE that is statistically indistinguishable from the best obtained during the challenge (Best TSCORE), on both manipulated datasets, but not in CINA0. This suggests that linear ridge regression ensembles may not provide a genuinely competitive base classifier for this benchmark, especially as the difference is quite large for CINA0. Note that BLogReg was used as the base classifier for the final challenge submission as the mean AUC scores for the individual component classifiers was lower than that for linear ridge regression.

- For both manipulated datasets, the TSCORE achieved using no feature selection at all is statistically superior to the best overall causal feature selection method (Yin *et al.*). For

6. BLogReg: tolerance = 1×10^{-6}, HITON_MB: "z" statistic, threshold = 0.05, maximum size of conditioning set = 5, PC: "z" statistic, threshold = 0.05, $k = 4$.

the unmanipulated dataset, the difference in performance is statistically insignificant. In this case, it seems that, while regularisation is not that effective in suppressing the influence of uninformative features, the instability of feature selection procedure means that better performance is only available given prior knowledge of the causal relationships.

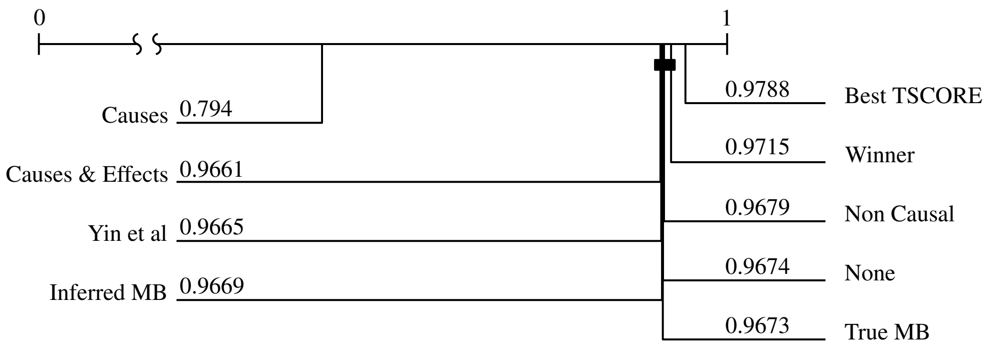
- The differences in TSCORE between the best causal feature selection procedure (Yin *et al.*) and non-causal feature selection using BLogReg were statistically insignificant for all three datasets.

- None of the causal feature selection algorithms investigated, with the exception of reference entries, proved statistically superior to an ensemble trained using all of the available features for any of the three CINA datasets, again a rather disappointing result.

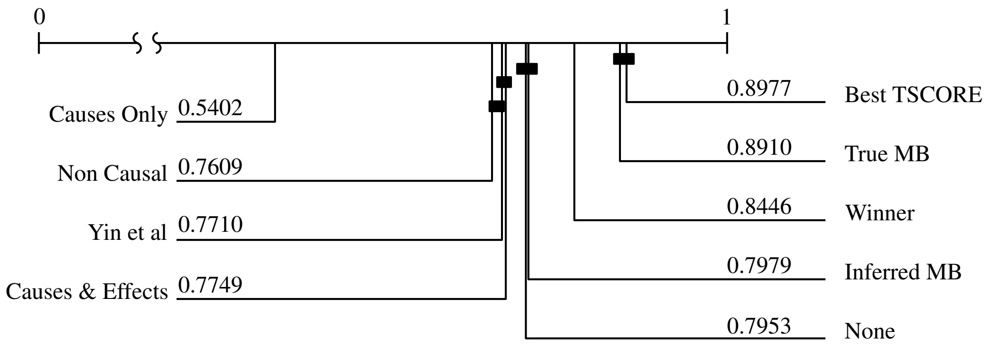
Table 5: Results obtained for the CINA benchmark, see caption of Table 1 for details.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
CINA0	Best TSCORE*	90	0.8913	0.9794	0.9788 ± 0.0029	—
	Winner‡	64	0.6000	0.9721	0.9715 ± 0.0032	—
	Non-causal	67	0.5708	0.9682	0.9679 ± 0.0035	0.9660
	None	132	0.7908	0.9677	0.9674 ± 0.0035	0.9664
	True MB	115	1.0000	0.9674	0.9673 ± 0.0035	0.9663
	Inferred MB	70	0.7708	0.9669	0.9669 ± 0.0035	0.9660
	Yin *et al.*	22	0.5957	0.9657	0.9665 ± 0.0034	0.9657
	Causes & effects	42	0.6826	0.9654	0.9661 ± 0.0035	0.9653
	Causes only	4	0.5174	0.7923	0.7911 ± 0.0046	0.5351
CINA1	Best TSCORE*	90	0.4542	0.9794	0.8977 ± 0.0043	—
	True MB	44	1.0000	0.8915	0.8910 ± 0.0040	—
	Winner‡	64	0.7053	0.9721	0.8446 ± 0.0047	—
	Inferred MB	70	0.5261	0.9669	0.7979 ± 0.0052	—
	None	132	0.5865	0.9677	0.7953 ± 0.0050	—
	Causes & effects	42	0.5477	0.9654	0.7749 ± 0.0050	—
	Yin *et al.*	24	0.5823	0.9652	0.7710 ± 0.0048	—
	Non-causal	67	0.6436	0.9682	0.7609 ± 0.0053	—
	Causes only	4	0.5114	0.7923	0.5402 ± 0.0056	—
CINA2	True MB	44	1.0000	0.8915	0.8920 ± 0.0043	—
	Best TSCORE†	32	1.0000	0.8909	0.8910 ± 0.0042	—
	Winner‡	4	0.7053	0.8137	0.8157 ± 0.0052	—
	None	132	0.5865	0.9677	0.5502 ± 0.0043	—
	Inferred MB	70	0.5261	0.9669	0.5469 ± 0.0041	—
	Non-causal	67	0.6436	0.9682	0.5464 ± 0.0039	—
	Causes & effects	42	0.5477	0.9654	0.5394 ± 0.0038	—
	Yin *et al.*	18	0.5794	0.9636	0.5373 ± 0.0041	—
	Causes only	4	0.5114	0.7923	0.4825 ± 0.0035	—

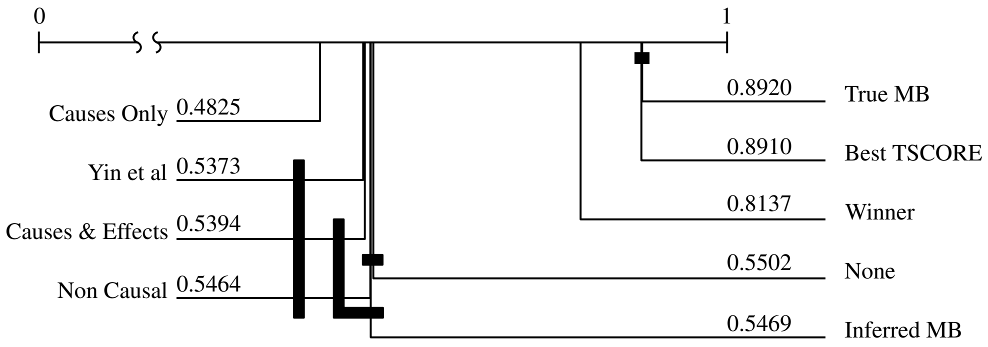
*Reference "SNB(CMA), IID assumption". †Reference "CINA Test", ‡ Yin-Wen Chang "final submission".



(a)



(b)



(c)

Figure 3: Statistical significance diagrams for (a) CINA0, (b) CINA1 and (c) CINA2. The axis represents the TSCORE statistic, and the heavy bars denote groups of classifiers with statistically indistinguishable performance.

3.6 MARTI — Measurement ARTIfact

Like REGED, the MARTI benchmark represents a re-simulated microarray classification task, the aim of which is to identify genes that may be responsible for lung cancer. However, in this case additive zero-mean correlated noise has been added to the data to simulate measurement artifacts introduced by an instrument used to collect the training data that is substantially inferior to a more accurate instrument used to gather the test data. Figure 4 shows an example of the correlated noise corrupting a training sample from the MARTI benchmark. The correlated noise is likely to confuse both causal and non-causal feature selection algorithms, and therefore MARTI differs from the other challenge datasets in that non-trivial pre-processing is required. We adopt a kernel ridge regression approach to try to estimate the noise for each training pattern as a function of the x- and y-co-ordinates of the spot on the microarray image. Let $\boldsymbol{X}$ represent the $d \times 2$ matrix, where each row, $\boldsymbol{x}_i$, gives the x- and y-co-ordinates of a spot on the microarray image, and $\boldsymbol{Y}$ represents the $d \times \ell$ matrix containing the expression levels for every gene, where each row, $\boldsymbol{y}_i$, represents a spot and each column represents a sample. We assume that the noise contaminating the expression levels can be approximated by a linear model in a feature space induced by a radial basis function kernel, with the expression levels themselves modelled by a Gaussian noise process,

$$\boldsymbol{y}_i = \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{W} + \boldsymbol{\varepsilon_i}, \qquad \text{where} \qquad \boldsymbol{\epsilon}_i \sim \mathrm{N}\left(\boldsymbol{0}, \sigma_i^2 \boldsymbol{I}\right), \tag{4}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ represents the image of the data in the kernel induced feature space. Note that a heteroscedastic noise model is used (e.g. Cawley et al., 2004) as considerable variation is evident in the range of expression of different genes. The model (4) is equivalent to a multi-output weighted kernel ridge regression model (Saunders et al., 1998), with the weights given by the inverse noise variance for each spot, $\boldsymbol{\sigma}^{-1} = \left(\sigma_1^{-2}, \sigma_2^{-2}, \ldots, \sigma_d^{-2}\right)$. The iterative training algorithm alternates updates of the model parameters with re-estimation of the noise variance terms using the model residuals. The usual regularisation and kernel parameters were tuned via numerical minimisation of the cross-validation error. Estimates of the true expression profiles can then be obtained by simply subtracting from $\boldsymbol{Y}$ the estimate of the correlated noise given by the fitted model. The results obtained[7] on this benchmark are shown in Table 6, the corresponding statistical significance diagram is shown in Figure 5:

- The pre-processing steps described above proved quite satisfactory, as demonstrated by the similarity of results obtained on the REGED and MARTI benchmarks, shown in Ta-

7. BLogReg: tolerance = 1×10^{-6}, HITON_MB: "z" statistic, threshold = 0.05, maximum size of conditioning set = 5, PC: "z" statistic, threshold = 0.05, k = 16.

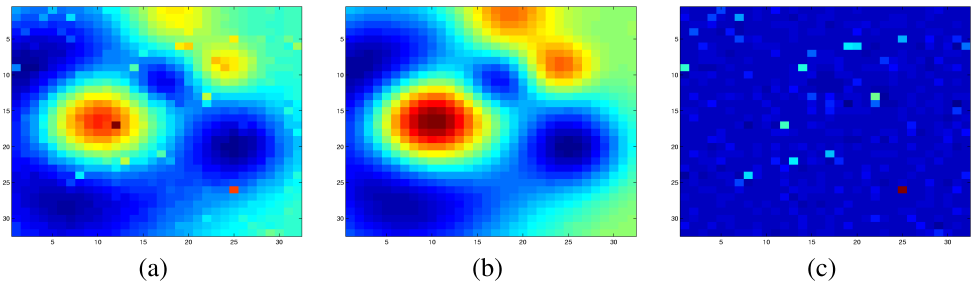


Figure 4: Example pattern from the training set of the MARTI benchmark (a) raw microarray image (b) estimate of correlated noise and (c) filtered expression levels.

bles 3 and 6 respectively, however no use was made of the calibrant features or knowledge of manipulated features so the results are likely to be somewhat sub-optimal.

- The TSCORE for a linear ridge regression ensemble using knowledge of the true Markov blanket exceeds that of the best TSCORE achieved by any challenges submission, by a statistically significant margin on the manipulated datasets. This suggest that linear ridge regression ensembles are competitive as a base classifier for this application.

- For both manipulated datasets, the TSCORE achieved using no feature selection at all is statistically superior to the best overall causal feature selection method (Yin *et al.*). For the unmanipulated dataset, the difference in performance is statistically insignificant. In this case, it seems that while regularisation is not that effective in suppressing the influence of uninformative features, the instability of feature selection procedure means that better performance is only available given prior knowledge of the causal relationships.

- The TSCORE achieved using non-causal feature selection was statistically indistinguishable from that achieved by the best all-round causal feature selection procedure (Yin *et al.*) on the unmanipulated data (MARTI0), was statistically superior on one manipulated dataset (MARTI1) and statistically inferior on the other (MARTI2), suggesting that causal feature selection does not improve overall on non-causal feature selection.

3.7 Final Challenge Submission

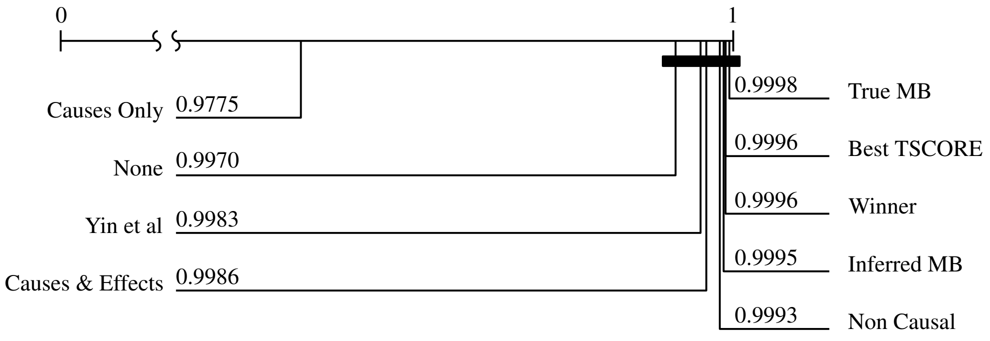
Table 7 shows the results for the final challenge submission. BLogReg was used as the base classifier for the CINA benchmark, as this gave slightly better performance under the 100-fold repeated hold-out procedure used for validation during the development phase of the challenge. The full set of models for the SIDO datasets was incomplete by the challenge deadline; the best models proved to be simple ridge regression models with no feature selection (note that there were four features in the training set with zero variance, hence only 4928 features were actually used by the classifier). The rankings indicate that the base classifiers were good choices for the benchmarks considered, and so the comparison of feature selection methods provides a good indication of their relative merits. Further details of the final challenge submission are available in the supplementary material.

4. Recommendations

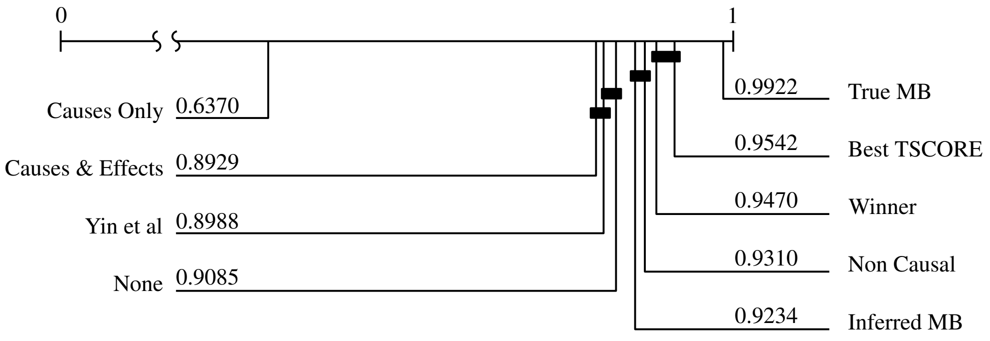
The results of the investigation presented in the previous section suggest that further research is required in order for causal feature selection methods to approach more closely the superior performance that experimental "ground truth" evidence and qualitative arguments suggest are available. We are however in a position to make some recommendations for use in practical applications:

- Use regularisation: Regularisation is known to be a viable alternative to feature selection in applications with unmanipulated data, where predictive performance is the primary objective rather than discovering a compact set of informative features (Miller, 2002). It has also been argued that when faced with covariate shift it may be better to include bad features rather than delete good features (Guyon et al., 2008, §6.2), in which case using a larger feature set with regularisation to avoid over-fitting seems a sensible strategy.

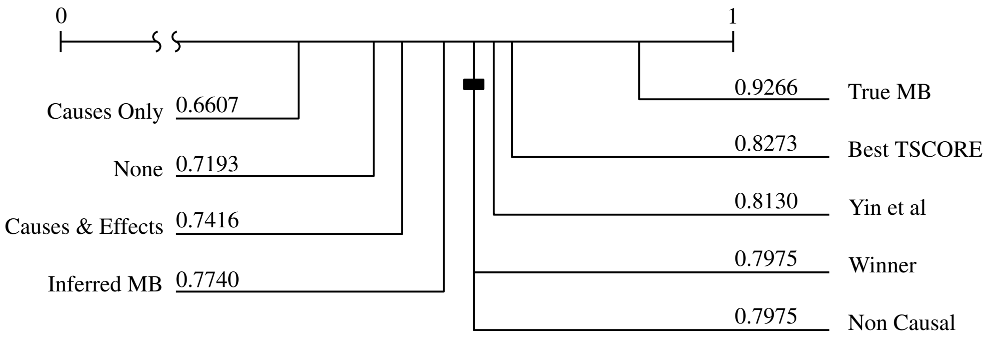
- Use Bagging: A comparison of the size of the true Markov blanket of the unmanipulated distribution with the number of determined to belong to the Markov blanket of individ-



(a)



(b)



(c)

Figure 5: Statistical significance diagrams for (a) MARTI0, (b) MARTI1 and (c) MARTI2. The axis represents the TSCORE statistic, and the heavy bars denote groups of classifiers with statistically indistinguishable performance.

ual component classifiers and the number of features used by the ensemble as a whole, suggests that identification of the Markov blanket using `HITON_MB` is unstable (i.e. the composition of the Markov blanket depends substantially on the sample of data from which it was inferred). Model selection, including the tuning of the regularisation parameter is also subject to over-fitting the selection criterion (Cawley and Talbot, 2007), and bagging will help to alleviate this also (Hall and Robinson, 2009).

- Investigate alternative base classifiers: In this study, we investigated only two base classifiers, linear ridge regression and BLogReg (for CINA). It may be that the benefits of causal feature selection may be obscured by the use of a base classifier that is unable to take advantage of non-linear relationships between features.

- In orienting the edges in the causal graph, it would be better to pre-filter the features to include not only the Markov blanket of the target, but also the parents and children of all features within the Markov blanket (c.f. Yin et al., 2008).

Table 6: Results obtained for the MARTI benchmark, see caption of Table 1 for details.

Dataset	Selection	FNUM	FSCORE	DSCORE	TSCORE	AUC
MARTI0	True MB	21	1.0000	0.9997	0.9998 ± 0.0010	0.9991
	Best TSCORE*	148	0.9078	1.0000	0.9996 ± 0.0010	—
	Winner§	128	0.8697	1.0000	0.9996 ± 0.0012	—
	Inferred MB	131	0.8862	1.0000	0.9995 ± 0.0011	0.9994
	Non-causal	44	0.8029	0.9998	0.9993 ± 0.0014	0.9986
	Causes & effects	15	0.7849	0.9987	0.9986 ± 0.0016	0.9978
	Yin *et al.*	11	0.6896	0.9982	0.9983 ± 0.0018	0.9973
	None	1024	0.7980	1.0000	0.9970 ± 0.0019	0.9950
	Causes only	3	0.5714	0.9821	0.9775 ± 0.0031	0.9346
MARTI1	True MB	14	1.0000	0.9889	0.9922 ± 0.0024	—
	Best TSCORE†	8	1.0000	0.8992	0.9542 ± 0.0041	—
	Winner§	32	0.8064	1.0000	0.9470 ± 0.0039	—
	Non-causal	44	0.7752	0.9998	0.9310 ± 0.0039	—
	Inferred MB	131	0.8265	1.0000	0.9234 ± 0.0045	—
	None	1024	0.7923	1.0000	0.9085 ± 0.0047	—
	Yin *et al.*	11	0.6399	0.9982	0.8988 ± 0.0046	—
	Causes & effects	15	0.7820	0.9987	0.8929 ± 0.0049	—
	Causes only	3	0.5347	0.9821	0.6370 ± 0.0059	—
MARTI2	True MB	2	1.0000	0.9277	0.9266 ± 0.0049	—
	Best TSCORE‡	2	1.0000	0.8099	0.8273 ± 0.0060	—
	Yin *et al.*	11	0.9980	0.9982	0.8130 ± 0.0053	—
	Winner§	64	0.9956	0.9998	0.7975 ± 0.0059	—
	Non-causal	44	0.9976	0.9998	0.7975 ± 0.0059	—
	Inferred MB	131	0.9966	1.0000	0.7740 ± 0.0060	—
	Causes & effects	15	0.9956	0.9987	0.7416 ± 0.0063	—
	None	1024	0.9951	1.0000	0.7193 ± 0.0062	—
	Causes only	3	0.7485	0.9821	0.6607 ± 0.0062	—

*Gavin Cawley "marti001 part006", †Reference "MB_NB_F_S", ‡Reference "FMBLR", §Gavin Cawley "final models".

Table 7: Summary of results for the final challenge submission. Top Ts gives the best Tscore amongst all valid final submissions, Max Ts gives the optimal Tscore, given knowledge of the true causal relationships, estimated using reference submissions, see caption of Table 1 for further details.

Dataset	**Causal Discovery**		**Target Prediction**				**Rank**
	Fnum	**Fscore**	**Dscore**	**Tscore**	**Top Ts**	**Max Ts**	
CINA0	128	0.5166	0.9737	0.9743	0.9765	0.9788	
CINA1	128	0.5860	0.9737	0.8691	0.8691	0.8977	3
CINA2	64	0.5860	0.9734	0.7031	0.8157	0.8910	
MARTI0	128	0.8697	1.0000	0.9996	0.9996	0.9996	
MARTI1	32	0.8064	1.0000	0.9470	0.9470	0.9542	1
MARTI2	64	0.9956	0.9998	0.7975	0.7975	0.8273	
REGED0	128	0.9410	0.9999	0.9997	0.9998	1.0000	
REGED1	32	0.8393	0.9970	0.9787	0.9888	0.9980	2
REGED2	8	0.9985	0.9996	0.8045	0.8600	0.9534	
SIDO0	4928	0.5890	0.9840	0.9427	0.9443	0.9467	
SIDO1	4928	0.5314	0.9840	0.7532	0.7532	0.7893	1
SIDO2	4928	0.5314	0.9840	0.6684	0.6684	0.7674	

- Like conventional feature selection procedures, causal feature discovery methods appear to exhibit significant instability. An empirical characterisation of this instability would be an interesting area for further research.

5. Summary

In this paper, we have evaluated causal and non-causal feature selection procedures for ridge regression under covariate-shift. The reference submissions generated with knowledge of the true causal relationships clearly demonstrate that causal feature selection is very effective in mitigating against covariate-shift. However the models with causal feature selection procedures investigated here generally failed to out-perform models with non-causal feature selection (or indeed without a feature selection step), except on the most basic toy benchmark (LUCAS). This is a surprising and disappointing result for datasets designed for causal inference. It should be noted that the causal feature selection procedures are also computationally expensive, for instance identification of the Markov blanket for the SIDO dataset using HITON_MB took on average 76 hours, 57 minutes 8 seconds, and orientation of causal links using the PC algorithm took on average 50 hours, 21 minutes and 26 seconds. This means that the SIDO experiments consumed approximately 18 processor-months, without providing any improvement in predictive accuracy! These results demonstrate that causal inference is a challenging task, where further theoretical and algorithmic advances are likely to bring substantial practical benefits and where a more detailed empirical study is clearly warranted.

Acknowledgments

I would like to thank the co-organizers for their efforts in staging a very interesting and, for myself at least, educational challenge. I would also like to thank Gareth Janacek, the editors and the anonymous reviewers for their helpful and constructive comments, and Nicola Talbot for her help in preparing the manuscript.

References

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON: A novel Markov blanket algorithm for optimal variable selection. In *Proc. AMIA Annual Symposium*, pages 21–25, 2003.

D. M. Allen. The relationship between variable selection and prediction. *Technometrics*, 16: 125–127, 1974.

G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proc. IJCNN-06*, pages 1661–1668, July 16–21 2006.

G. C. Cawley and N. L. C. Talbot. Gene selection in cancer classification using sparse logistic regression with Bayesian regularization. *Bioinformatics*, 22(19):2348–2355, October 1 2006.

G. C. Cawley and N. L. C. Talbot. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, April 2007.

G. C. Cawley, N. L. C. Talbot, R. J. Foxall, S. R. Dorling, and D. P. Mandic. Heteroscedastic kernel ridge regression. *Neurocomputing*, 57:105–124, March 2004.

Y.-W. Chang and C.-J. Lin. Feature ranking using linear SVM. *JMLR: Workshop and Conference Proceedings*, 3, WCCI-2008 Workshop on Causality:53–54, 2008.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, January 1992.

I. Guyon and A. Eliseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov. Design and analysis of the causation and prediction challenge. *JMLR: Workshop and Conference Proceedings*, 3, WCCI-2008 Workshop on Causality:1–33, 2008.

P. Hall and A. P. Robinson. Reducing the variability of crossvalidation for smoothing parameter choice. *Biometrika*, 96(1):175–186, March 2009.

A. Miller. *Subset selection in regression*, volume 95 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, second edition, 2002.

J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset shift in machine learning*. Neural Information Processing Series. MIT Press, 2009.

K. Saadi, G. C. Cawley, and N. L. C. Talbot. Optimally regularised kernel Fisher discriminant classification. *Neural Networks*, 20(7):832–841, September 2007.

C. Saunders, A. Gammermann, and V. Vovk. Ridge regression in dual variables. In J. Shavlik, editor, *Proc. ICML-98*. Morgan Kaufmann, 1998.

S. Weisberg. *Applied linear regression.* John Wiley and Sons, New York, 2nd edition, 1985.

J. Yin, Y. Zhou, C. Wang, P. He, Zhengm C., and Z. Geng. Partial orientation and local structure learning of causal networks for prediction. *JMLR: Workshop and Conference Proceedings*, 3, WCCI-2008 Workshop on Causality:93–105, 2008.

Table 8: Mean number of features used, and hold-out set AUROC score, over the 100 models comprising each of the ensembles used to make predictions.

Benchmark	**Selection**	**Features**	**AUROC**
LUCAS	None	11.00	0.9079
	Non-causal	10.99	0.9079
	Markov blanket	5.01	0.9082
	Causes & effects	4.00	0.8910
	Causes only	2.00	0.7832
LUCAP	None	143	0.9695
	Non-causal	6.03	0.9426
	Markov blanket	47.83	0.9674
	Causes & effects	39.91	0.9664
	Causes only	2.06	0.8089
CINA	None	132.00	0.9664
	Non-causal	29.44	0.9660
	Markov blanket	55.30	0.9660
	Causes & effects	21.21	0.9653
	Causes only	1.02	0.5351
REGED	None	999.00	0.9962
	Non-causal	14.69	0.9997
	Markov blanket	24.85	0.9995
	Causes & effects	11.11	0.9996
	Causes only	2.39	0.8961
SIDO	None	4932.00	0.9472
	Non-causal	28.96	0.9226
	Markov blanket	136.27	0.9348
	Causes & effects	10.07	0.8798
	Causes only	9.95	0.8733
MARTI	None	1024.00	0.9950
	Non-causal	15.19	0.9986
	Markov blanket	26.86	0.9994
	Causes & effects	8.60	0.9978
	Causes only	1.56	0.9346

Appendix I

Causation and Prediction Challenge Fact Sheets

Causation and Prediction Challenge Fact Sheet 1

Title: Feature selection, redundancy elimination, and gradient boosted trees

Author: Alexander Borisov

Address: INTEL Corporation, Advanced Analytics team

Email: alexander.borisov@intel.com

Acronym of your best entry: ACE+GBT

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Alexander_Borisov.html

References

[1] Borisov, A., Torkkola, K., Tuv E. (2006) "Best Subset Feature Selection for Massive Mixed – Type Problems". 7th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL-2006, Lecture Notes in Computer Science Series, Vol. 4224, 1048-1056, Springer 2006.

[2] Tuv E., Borisov A., Runger G., Torkkola K. "Best Subset Feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination". Submitted to Journal of Machine learning Research, 2008.

Method

No preprocessing was done.

Feature selection method contains 2 steps. For unbalanced datasets, all classifiers (RF, GBT) use stratified sampling to compensate, i.e. for each tree in ensemble 60% samples of rare class and same quantity of frequent class are selected as input.

1. **Feature selection** using ensemble classifiers (ACE FS). Contrast variables that are permutation of original features are added. Importance of each variable in RF ensemble is compared versus importance of probes using t-test over several ensembles. Variables that are more important in statistical sense then most of probes are selected as important. Variables are ordered according to sum of Gini index reduction in tree splits.

2. **Variable masking** is estimated on important variables with GBT ensemble using surrogate splits (if more important variable has surrogate on less important one, the second variable is masked by the first). Again, statistically significant masking pairs are selected, then subset of mutually non-masked variables with high importance is selected.

3. **Effect of found variables** is removed using RF ensemble.

Steps 1–3 are repeated until no more important variables remain.

Variables are sorted by cumulative variable importance (computed as usual for ensemble of trees, i.e. importance of feature is sum of split weights on this feature) in ensembles constructed on step 3. Then top 1, 2, 4,... and so on variables are used to build GBT model. For more than

100 features we used embedded feature selection in GBT that reduces the running time. The idea is that with redundant feature elimination probes will be recognized as redundant, and will have zero or very small importance.

The following parameters of GBT were selected empirically for all datasets:

800 iterations, tree depth = 8, shrinkage = 0.01

For FS, #of trees in series = 50, #series = 20, importance and masking quantile = 0.75, tree depth = 6.

Results

Table I.1: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	171	ACE+GBT	512/999**	0.6969	0.9996 ± 0.0009	0.9998	1		
REGED1	171	ACE+GBT	512/999**	0.6838	0.9095 ± 0.0038	0.9888	0.998	0.8331	8
REGED2	171	ACE+GBT	8/999**	0.9107	0.5902 ± 0.0059	0.86	0.9534		
SIDO0	171	ACE+GBT	256/4932**	0.4653	0.9337 ± 0.0076	0.9443	0.9467		
SIDO1	171	ACE+GBT	2048/4932**	0.468	0.6908 ± 0.0136	0.7532	0.7893	0.7333	8
SIDO2	171	ACE+GBT	4932/4932**	0.468	0.5756 ± 0.0130	0.6684	0.7674		
CINA0	171	ACE+GBT	64/132**	0.6312	0.9755 ± 0.0029	0.9765	0.9788		
CINA1	171	ACE+GBT	64/132**	0.6085	0.8236 ± 0.0048	0.8691	0.8977	0.8328	5
CINA2	171	ACE+GBT	64/132**	0.6085	0.6993 ± 0.0043	0.8157	0.891		
MARTI0	171	ACE+GBT	512/1024**	0.4841	0.8872 ± 0.0050	0.9996	0.9996		
MARTI1	171	ACE+GBT	32/1024**	0.5188	0.7005 ± 0.0061	0.947	0.9542	0.7638	7
MARTI2	171	ACE+GBT	128/1024**	0.5998	0.7036 ± 0.0063	0.7975	0.8273		

Quantitative advantages

Method is fast (~a minute for one FS iteration on largest dataset)

Time complexity is proportional to $(\text{Fsel} + \text{Fimpvar}) * N * \log N * \text{Ntrees} * \text{Nensembles} * \text{Niter} + \text{Niter} * \text{Fimpvar}^2$,

Niter – #of iteration of ACE FS algorithm always < 10, usually 3–4

Nensembles = 20 (number of ensembles for t-test)

Ntrees = 50 (number of trees in RF or ensemble)

N – number of samples,

Fsel = number of selected important variables per tree split (sqrt(total number features) or less)

Fimvar -– total number of selected important variable.

Works with any variable types, mixed values, requires no preprocessing.

Qualitative advantages

Requires no investigation of causal structure.

It is not a push-button application. ACE is a part of internally developed at Intel machine learning toolset called IDEAL not available for external usage.

Keywords:

- **Preprocessing or feature construction:** no.
- **Causal discovery:** indirect trough redundant feature elimination strategy in ACE method. Probes should be more likely to be redundant and go at the end of the feature sorted list.
- **Feature selection:** embedded feature selection using tree ensembles.
- **Classifier:** RF, GBT (tree ensembles).
- **Hyper-parameter selection:** used defaults that work well on most data sets.
- **Other:** ensemble method.

Causation and Prediction Challenge Fact Sheet 2

Title: Regularized and Averaged Selective Naïve Bayes Classifier

Author: Marc Boullé

Address: France Telecom R&D, 2, avenue Pierre Marzin, 22307 Lannion cedex – France

Email: `marc.boulle@francetelecom.com`

Acronym of your best entry: SNB(CMA), IID assumption

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/MB.html`

References

[1] M. Boullé. Compression-Based Averaging of Selective Naïve Bayes Classifiers. Journal of Machine Learning Research, 8:1659–1685, 2007.

[2] M. Boullé. MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning, 65(1):131–165, 2006.

Method

IID assumption

The method is based on the IID assumption and ignores causal discovery. Although its results make sense only on the initial datasets, it was also applied on the manipulated datasets to challenge the causal methods.

Noise filtering for MARTI

The data samples of MARTI were preprocessed to remove the correlated noise as follows:

- The 2-dimensional nature of the patterns was reconstructed using the variable indices
- The low frequency noise was removed by convolving the image thus obtained with a 2-d Gaussian filter to obtain the "background", then subtracting this background from the image. Specifically, we used the kernel ker=[1 4 6 4 1]'*[1 4 6 4 1]; ker=ker./sum(sum(ker)); without tuning its width. Better results might be obtained with other kernels or bay adjusting the width.
- To alleviate border effects, the image was first extrapolated by tiling the borders with average values of near border variables.
- To alleviate the problem of high intensity outliers, we detected points whose value was more that one standard deviation away from the mean of their neighbors and replaced them by that mean before computing the background.
- Finally, a bias value was added to the resulting filtered image such that the average of the calibrants is the same as that is test data (namely 1).

Compression-based averaging of selective naïve Bayes classifiers

Our method is based on the naïve Bayes assumption, and incorporates optimal preprocessing, feature selection and model averaging as follows:

- All the input features are preprocessed using the Bayes optimal MODL discretization method, which results in a reliable and accurate estimation of the univariate class conditional probabilities.
- Feature selection is performed using a Bayesian approach to find a trade-off between the number of selected features and the performance of the selective naïve Bayes classifier: this provides a regularized feature selection criterion. The feature selection search is performed using alternate forward selection and backward elimination searches on randomly ordered feature sets: this provides a fast search heuristic, with super-linear time complexity with respect to the number of instances and features.
- The method exploits a variant of feature selection: feature "soft" selection. Whereas feature "hard" selection gives a "Boolean" weight to the features according to whether they selected or not, the method gives a continuous weight between 0 and 1 to each feature. This weighing schema of the features comes from a new classifier averaging method, derived from Bayesian Model Averaging, with a logarithmic smoothing of the posterior distribution of the models.

Advantages

- Bayesian regularization technique (for preprocessing and feature selection): all the available data is used for training, with no need for validation or cross-validation
- fully automatic
- highly scalable (train and deploy)
- accurate and reliable
- easy interpretation
- compute the posterior probabilities

Limitations

- the naïve Bayes assumption might be harmful is no subset of variables in the initial representation is compliant with the conditional independence assumption: this can be leveraged by feature construction to extend the representation space
- no causal discovery

Results

For each of the four datasets, one single model was trained and applied on the initial test set (0) and the two manipulated test sets (1 and 2).

The results are very good on the initial test sets, which conform to the IID assumption: our method gets the best Tscore on REGED0 and CINA0, and is within 1% of the best performance for the two other datasets.

Surprisingly, the results are good on some tests sets 1, with the best Tscore on REGED1 and CINA1.

This might be explained by two features of our method:

- the optimal preprocessing is highly reliable: any input noise variable is almost surely detected as irrelevant and discarded
- the model averaging accounts for the uncertainty on model selection: whereas one single maximum a posteriori (MAP) model might select a wrong subset of variables with respect to causation, averaging a large number of models leverages the effect of irrelevant features

Not surprisingly, the results are very poor on the test sets 2, which are heavily manipulated. Our method based on the IID assumption is clearly outperformed by the causal methods.

Table I.2: Results table. The star following the feature number indicates that the feature set was sorted. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**
REGED0	321	SNB(CMA), IID assumption	122/999*	0.8352	1.0000 ± 0.0002	1	1	
REGED1	321	SNB(CMA), IID assumption	122/999*	0.7946	0.9980 ± 0.0015	0.998	0.998	0.8462
REGED2	321	SNB(CMA), IID assumption	122/999*	0.991	0.5407 ± 0.0061	0.86	0.9534	
SIDO0	321	SNB(CMA), IID assumption	1592/4932*	0.6831	0.9297 ± 0.00700	0.9443	0.9467	
SIDO1	321	SNB(CMA), IID assumption	1592/4932*	0.3922	0.6337 ± 0.0132	0.7532	0.7893	0.7104
SIDO2	321	SNB(CMA), IID assumption	1592/4932*	0.3922	0.5678 ± 0.0129	0.6684	0.7674	
CINA0	321	SNB(CMA), IID assumption	90/132*	0.8913	0.9788 ± 0.0029	0.9788	0.9788	
CINA1	321	SNB(CMA), IID assumption	90/132*	0.4542	0.8977 ± 0.0043	0.8977	0.8977	0.8694
CINA2	321	SNB(CMA), IID assumption	90/132*	0.4542	0.7318 ± 0.0043	0.8157	0.891	
MARTI0	386	SNB(CMA), IID assumption (F)	22/1024*	0.7097	0.9848 ± 0.0031	0.9996	0.9996	
MARTI1	386	SNB(CMA), IID assumption (F)	22/1024*	0.6716	0.8891 ± 0.0043	0.947	0.9542	0.8869
MARTI2	386	SNB(CMA), IID assumption (F)	22/1024*	0.9936	0.7868 ± 0.0058	0.7975	0.8273	

Code

Our implementation was done in C++.

The software is available as a shareware on http://perso.rd.francetelecom.fr/boulle/.

Keywords:

- **Preprocessing or feature construction:** Bayes optimal discretization
- **Causal discovery:** none
- **Feature selection:** Bayesian regularization, fast forward backward feature selection
- **Classifier:** naïve Bayes, compression-based model averaging
- **Hyper-parameter selection:** none, automatic

Causation and Prediction Challenge Fact Sheet 3

Title: A Strategy for Making Predictions Under Manipulation

Author: Laura Brown and Ioannis Tsamardinos

Contact:

Laura Brown,
Eskind Biomedical Library 4th floor, 2209 Garland Ave. Nashville, TN 37232 USA.
laura.e.brown@vanderbilt.edu
Ioannis Tsamardinos,
FORTH-ICS, N. Plastira 100, Vassilika Vouton GR-700 13 Heraklion Crete, GREECE.
tsamard@ics.forth.gr

Acronym of your best entry: final test

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/LEBYT.html

Complete paper:
A Strategy for Making Predictions Under Manipulation. Laura E. Brown and Ioannis Tsamardinos; JMLR W&CP 3:35–52, 2008.

References

Bach's -– Bach, F.R. and Jordan, M.I. NIPS, 2002

FCI — Spirtes, P. et al. 1993

HITON -– Aliferis, C.F. et al. AMIA, 2003

MMHC — Tsamardinos, I. et al. Machine Learning, 2006

MMPC, MMMB -– Tsamardinos, I. et al. SIGKDD, 2003

RFE — Guyon et al. Machine Learning, 2002

Regions of Interest -– Tsamardinos et al., Tech Report DSL-03-02, DBMI, Vanderbilt University

Method

Preprocessing

The preprocessing was tailored to 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; binary variables were all set to values of zero and one. For the MARTI data set, the calibrant variables were used to fit a spline across the training array estimating the correlated noise model. The estimated noise was then subtracted from the training samples.

Causal discovery

We addressed the following problems in turn (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.

Once the initial data sets have been pre-processed, the next step of our procedure was to identify the Markov Blanket (MB) from the non-manipulated data sets, i.e., the parents, children, and spouses of the target. Several variable selection techniques, mostly causally-based, were applied to this problem in order to both identify the MB and also attempt to gain insight into the predictive variables in each domain. The published methods used included MMPC (for identifying the parents and children of a target variable, PC(T)), MMMB (for identifying the Markov Blanket of a target variable), HITON-MB (for identifying the Markov Blanket of a target variable), and RFE (variable selection method to identify predictive variables). All of the above causally-based methods assume that if a variable belongs in the neighbor's set of the target, it will have a detectable pairwise association with the target. RFE is able to additionally identify variables that participate in strong multivariate associations, even if they have no detectable pairwise association (e.g., parity functions). A new technique under development (recently submitted for publication), called Feature Space Markov Blanket (FSMB) combines kernel-based methods with causally-based methods to identify the neighbor's set in feature space, where multivariate associations may become pairwise associations. Any additional multivariate associations identified by FSMB were added to the Markov Blanket and participated in subsequent analysis. At this point, we know that our Markov Blanket set contains all variables need for calculation of the Causal Markov Blanket in any *manipulated* distribution (plus false positives depending on the type of manipulations).

In the second step, starting from the above Markov Blanket we identified the skeleton structure of the Bayesian Network around the target variable recursively using the MMPC algorithm, up to three edges away from the target. This region of interest makes it practical to apply causal algorithms that cannot scale up to the sizes of all the networks in the challenge. There are theoretical reasons why a network region of depth 3 allows most inferences about the orientation of the edges to be made. The idea of region learning was first described in Tsamardinos et al. 2003 (DSL-03-02). Further theoretical and experimental results are about to be submitted.

In the third step of our analysis we tried to orient the edges and discover whether an edge appears in the network due to a hidden confounder. The orientations of the edges and the confounded edges are necessary to identify the Causal Markov Blanket of the target, i.e., the Markov Blanket in the manipulated distribution. For the case of continuous or mixed data, an adaptation of Bach's algorithm was used. For the case of binary data, MMHC was used to find the top scoring network. The final network was converted into a PDAG to find the orientation of the compelled edges. To obtain suspected hidden confounders we used the FCI algorithm and developed our own extension of the Y-structures' identification algorithm (see Mani, et al. UAI 2006) for the purposes of the challenge. Our extension is based on tests of independence rather than scoring and is able to handle confounders of the top variables in the Y-structure. We suspect this way we can identify Y-structures in more general conditions that those described in Mani.

For the non-manipulated data set, the Markov Blanket was selected as the variables to include in the variable list. The members were sorted first by parents, children, and spouses. For the manipulated data set where manipulations were known, the variable list consisted of the Causal Markov Blanket. This we defined to be the *effective* parents, the non-manipulated children, and the *effective* spouses of the target. The effective parents are the parent variables of the target that are still predictive of the target in the manipulated distribution. They are the direct *causes* of the target (i.e., parents found not to be confounded) plus the parents of the target that are not manipulated. The effective spouses are the effective parents of the non-manipulated children. For the manipulated data set where the manipulations were unknown, the variable list consisted of the causes of the target node, i.e., parents found not to be confounded. Weighting the evidence of the orientation of an edge and whether it is due to a hidden confounder or not by the above methods was done based on methods under development and submission for publication. For some edges the above methods failed to provide evidence whether they are causal or not (i.e., confounded) or about their direction. Thus, some guesswork was necessary that gave rise to different variable subsets that we have tried.

Classification and Model Selection

Once the variable list was determined for each problem and data set, a final classification model was trained using only the variables of the feature list. The models trained for this task were SVMs. An n-fold cross validation design was used to select the optimal parameters (type of kernel, kernel parameters, and C value). 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.

Results

Table I.3: Results table. The star following the feature number indicates that the feature set was sorted. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants. This entry obtained best average score for REGED among all valid last entries.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>
REGED0	1491	final test	15/999*	0.8571	0.9998 ± 0.0010	0.9998	1	
REGED1	1491	final test	9/999*	0.7851	0.9673 ± 0.0036	0.9888	0.998	0.9423
REGED2	1491	final test	3/999*	1	0.8600 ± 0.0053	0.86	0.9534	
SIDO0	1491	final test	13/4932*	0.5015	0.9230 ± 0.0069	0.9443	0.9467	
SIDO1	1491	final test	4/4932*	0.5003	0.6073 ± 0.0027	0.7532	0.7893	0.6909
SIDO2	1491	final test	4/4932*	0.5003	0.5426 ± 0.0027	0.6684	0.7674	
CINA0	1491	final test	101/132*	0.8496	0.9721 ± 0.0031	0.9765	0.9788	
CINA1	1491	final test	5/132*	0.4716	0.5113 ± 0.0053	0.8691	0.8977	0.6015
CINA2	1491	final test	5/132*	0.4716	0.3210 ± 0.0025	0.8157	0.891	
MARTI0	1491	final test	24/1024*	0.5869	0.9681 ± 0.0037	0.9996	0.9996	
MARTI1	1491	final test	17/1024*	0.5643	0.7837 ± 0.0056	0.947	0.9542	0.8083
MARTI2	1491	final test	3/1024*	0.4985	0.6730 ± 0.0060	0.7975	0.8273	

The methods described above generally resulted in a compact variable list representing either the Markov Blanket or Causal Markov Blanket. The results on CINA were very low and are indicative to the inappropriateness of the statistical tests used in MMPC and MMMB when mixed data was used. The MMPC and MMMB algorithms have statistical tests provided for when the data is entirely binary or continuous (with a binary target); the mixed data set did not therefore match well to these methods.

The methods described above were implemented in Matlab. The MMPC, MMMB, and MMHC methods are available from the Causal Explorer library, `www.dsl-lab.org` (please note, we were in part the developers of these methods and may have slightly extended or modified the code from the precise implementation available in Causal Explorer). The SVMs were created using the LibSVM software (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`). Our method combined many different approaches and is not currently available as a push-button application although we are working on automating this process.

Code

Keywords:

- **Preprocessing or feature construction:** normalization
- **Causal discovery:** Bayesian Network
- **Feature selection:** filter
- **Classifier:** SVM
- **Hyper-parameter selection:** K-fold cross-validation

Causation and Prediction Challenge Fact Sheet 4

Title: Causation, Prediction, Feature Selection and Regularization

Author: Gavin Cawley

Address: School of Computing Sciences, UEA, Norwich, U.K.

Email: `gcc@cmp.uea.ac.uk`

Acronym of your best entry: final models

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Gavin_Cawley.html`

Method

Preprocessing

All continuous features are standardized to have zero mean and unit variance. For MARTI, the correlated noise was reduced by fitting a multi-output kernel ridge regression model, with a Gaussian RBF kernel, to the training that predicts the data as a function of the x and y co-ordinates. No special use was made of the calibration points, so the method was probably sub-optimal.

Causal discovery

The CausalExplorer package was used to detect feature sets representing the Markov blanket, direct causes + effects and direct causes only. This made use of the HITON_MB, PC and MMHC algorithms.

Feature selection

Sparse logistic regression with Bayesian regularization using a Laplace prior (BLogReg) was used for non-causal feature selection for comparison purposes. Models using the full feature set were also used to determine if regularization alone were sufficient. Also for the final submission using the multi-column format, some predictions are made by models using the feature weightings from other ridge regression models, so there is also a crude form of RFE used in some cases.

Classification

Ridge regression was used for MARTI, REGED and SIDO, in cases where there were more features than patterns, kernel ridge regression with a linear kernel was used for computational efficiency. For CINA, the BLogReg algorithm was used as this seemed to produce better results under cross-validation.

Model selection/hyper-parameter tuning

Virtual leave-one-out cross-validation using Allen's PRESS statistic was used for hyper-parameter selection throughout.

Performance evaluation

The model skill for the un-manipulated datasets was performed via 100-fold repeated hold out experiments. The predictions submitted for the challenge represent the mean of the resulting ensemble of 100 models. The number of features used by individual models is generally much smaller than that reported on the challenge website as not all features are used by all 100 models.

Fuller description of the methods used in "finalsubmission"

CINA

Sparse logistic regression with Bayesian regularisation using a Laplace prior was used as the base classifier for all CINA datasets. An ensemble of models from 100-fold repeated hold-out was used to make predictions. All features were standardised to have zero mean and unit variance. HITON_MB ($k = 2$, 'z' statistic, threshold=0.05) was used to pre-select the Markov blanket independently in each trial of the hold-out procedure for CINA0 and CINA1. HITON_MB ($k = 5$, 'z' statistic, threshold=0.05) was used for CINA2. Not all of the features are used by all of the members of the ensemble, so the feature set contains some features that are barely used (if at all).

SIDO

An optimally regularised ridge regression model with no feature selection was used for all datasets. An ensemble of models from 100-fold repeated hold-out was used to make predictions.

REGED

Optimally regularised kernel ridge regression used as the base classifier for all datasets, all features standardised, again an ensemble of 100 models is used to make predictions.

REGED0 HITON_MB ($k = 2$, 'z' statistic, threshold=0.05) used to identify the Markov blanket in each fold.

REGED1 Features known to be manipulated are discarded, HITON_MB ($k = 4$, 'z' statistic, threshold=0.05) used to identify the Markov blanket in each fold. The PC algorithm in Causal-Explorer ($k = 16$, 'z' statistic, threshold=0.05) was then used to find the direct causes and direct effects from the features comprising the Markov blanket.

REGED2 This is a hybrid model created to interpolate between more formal models:

PART #022 All features used by all 100 models for REGED0 were identified. The PC ($k = 16$, 'z' statistic, threshold=0.05) algorithm was then used to identify the direct causes using the entire training set. This identified two features (the reason for training additional models without ensembling was to populate the first few columns of the multi-column prediction matrix).

PART #021 All features used by all 100 models for REGED0 were identified. The PC ($k = 16$, 'z' statistic, threshold=0.05) algorithm was then used to identify the direct causes and direct effects using the entire training set rather than using an ensemble approach. This identified twelve features.

The features found by PART #022 were sorted in decreasing order of the magnitude of the weights of the model found in PART #022. Then the additional features found in PART #021 were added, sorted in order of the magnitude of the weights of the PART #021 model. The first eight features on this list were used to train a single model using the full training set. Roughly, this model contains the direct causes and a selection of the better correlated direct causes.

MARTI

The pre-processing step described in the paper (based on iteratively re-weighted kernel ridge regression) was used to remove the noise. All features were then standardised. Optimally regularised kernel ridge regression used as the base classifier. Again, some models were constructed to interpolate between more formal feature selection methods. A feature list was constructed as before from the following parts:

- PART #009 HITON_MB ($k = 2$,'z' statistic, threshold=2) used to find the Markov blanket, PC ($k = 16$, 'z' statistic, threshold=0.05) used to determine the direct causes (3 features)
- PART #009 HITON_MB ($k = 2$,'z' statistic, threshold=2) used to find the Markov blanket, PC ($k = 16$, 'z' statistic, threshold=0.05) used to determine the direct causes and effects (15 features).
- PART #011 BlogReg for non-causal feature selection (44 features).
- PART #007 HITON_MB ($k = 5$,'z' statistic, threshold=2) used to find the Markov blanket (131 features).
- PART #003 Full model trained on all features.

The features were ranked by PART and then by the magnitude of the corresponding weight of the model in the PART where first encountered. All of these PARTS used an ensemble of 100 models, and so the feature sets are relatively large.

MARTI0 Model trained on the first 128 elements of the feature list. This is likely to contain all direct causes and effects, some highly correlated features and most (if not all) of the true Markov blanket.

MARTI1 Model trained on the first 32 elements of the feature list. This will be all direct causes and all direct effects, plus some highly correlated features.

MARTI2 The BLogReg algorithm was used for non-causal feature selection. An ensemble of 100 models was used for predictions.

All submissions use a single classifier with nested subsets, but not all subsets correspond to simple atomic feature selection policies. I added the interpolating models to fill in the gaps as this could not decrease my chances of winning, even if they didn't help. However, I also performed more formal experiments so that some more solid conclusions could be drawn about the value of causal and non-causal feature selection methods. (I am convinced I should learn more about them!)

Table I.4: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	1373	final models	128/999**	0.941	0.9997 ± 0.0012	0.9998	1		
REGED1	1373	final models	32/999**	0.8393	0.9787 ± 0.0036	0.9888	0.998	0.9276	2
REGED2	1373	final models	8/999**	0.9985	0.8045 ± 0.0056	0.86	0.9534		
SIDO0	1373	final models	4928/4932**	0.589	0.9427 ± 0.0070	0.9443	0.9467		
SIDO1	1373	final models	4928/4932**	0.5314	0.7532 ± 0.0137	0.7532	0.7893	0.7881	1
SIDO2	1373	final models	4928/4932**	0.5314	0.6684 ± 0.0130	0.6684	0.7674		
CINA0	1373	final models	128/132**	0.5166	0.9743 ± 0.0031	0.9765	0.9788		
CINA1	1373	final models	128/132**	0.586	0.8691 ± 0.0046	0.8691	0.8977	0.8488	3
CINA2	1373	final models	64/132**	0.586	0.7031 ± 0.0047	0.8157	0.891		
MARTI0	1373	final models	128/1024**	0.8697	0.9996 ± 0.0012	0.9996	0.9996		
MARTI1	1373	final models	32/1024**	0.8064	0.9470 ± 0.0039	0.947	0.9542	0.9147	1
MARTI2	1373	final models	64/1024**	0.9956	0.7975 ± 0.0059	0.7975	0.8273		

Results

Much of the MATLAB code used is available from my website, BLogReg is available from `http://theoval.cmp.uea.ac.uk/cbl/blogreg/` and the KRR model is implemented in the GKM toolbox, `http://theoval.cmp.uea.ac.uk/~gcc/projects/gkm/`. Scripts were written to perform the repeated hold-out validation etc and to distribute the work across the parallel HPC facility.

Keywords:

- **Preprocessing or feature construction:** standardization, regression.
- **Causal discovery:** Bayesian Network, Information Theoretic Method.
- **Feature selection:** Embedded feature selection, feature ranking, RFE.
- **Classifier:** kernel-method, least-square, ridge regression, L1 norm regularization, L2 norm regularization, logistic regression, ensemble method.
- **Hyper-parameter selection:** cross-validation.
- **Other:** ensemble method.

Answers to questions asked by the reviewers

How was the information about manipulations used in REGED1?

The final submission for REGED1 discarded all features known to be manipulated.

Some algorithms, e.g. MMHC, require discrete data. How was the discretization performed?

I didn't do any discretization of continuous variables, but used the PC algorithm instead for problems with continuous features.

Algorithms like PC do not typically scale to datasets with more than 100 variables.

Yes, the simulations using the PC algorithm did take quite a long time! However, I used HITON_MB to find an estimate of the Markov blanket and then used the PC algorithm to direct the edges.

Similarly, it may be extremely computationally expensive to apply MMHC to some datasets with >1000–5000 variables. Did the author perform any pre-filtering to apply these algorithms?

My plan for SIDO was to find the Markov Blanket first (using HITON_MB) and then use MMHC to direct the edges in the Markov blanket, but I didn't finish the experiments in time for the close of the challenge. I hope to have completed them for Table 2 of the compete paper to give a more complete comparison of feature selection methods.

Causation and Prediction Challenge Fact Sheet 5

Title: SVM-Based Feature Selection for Causation and Prediction Challenge

Author: Yin-Wen Chang

Address: Department of Computer Science, No. 1, Sec. 4, Roosevelt Road, Taipei, 106, Taiwan

Email: b92059@csie.ntu.edu.tw

Acronym of your best entry: final submission

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Yin-Wen_Chang.html

Complete paper

Feature Ranking Using Linear SVM. Yin-Wen Chang and Chih-Jen Lin; JMLR W&CP 3:53-64, 2008.

Method

Preprocessing

We scale the numerical data sets and conduct instance-wise normalization on the binary data sets as preprocessing. Gaussian filter is used to eliminate the low frequency noise in the MARTI data sets.

Feature selection

- We experiment with various SVM-based feature selection methods and have several interesting findings. Feature ranking via linear SVM models seems to be useful for these data sets. Checking AUC with/without removing each feature gives similar rankings.
- During the development period, we experiment with various methods including feature ranking based on F-score, linear SVM weights, AUC/ACC change of removing a feature. After comparing the cross-validation AUC on training sets and the performance on toy examples, we use the feature ranking based on linear SVM weight in our final submission. We rank features according to the absolute value of weight corresponding to each feature.
- The models for all versions of each task are the same since we tried to obtain a general and simple model for the problem.
- In addition to cross-validation on training sets and the performance of toy examples, the quartile information is used since only one method is in the first quartile for all datasets.
- Discovering that nested subsets would results in better performance when the selected features are the same, we used nested subsets of features from the slist we submitted. The reason might be that the feature rank we give in slist is good enough.

- We did not use any knowledge derived from the test set to make the submissions.

Classification

We use L2-loss linear SVM to train the classifier.

Model selection/hyper-parameter tuning

Grid search is used to select the parameter of the SVM classifier.

Results

Table I.5: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1452	final submission	16/999**	0.8526	0.9998 ± 0.0009	0.9998	1		
REGED1	1452	final submission	16/999**	0.8566	0.9556 ± 0.0040	0.9888	0.998	0.9316	1
REGED2	1452	final submission	8/999**	0.997	0.8392 ± 0.0052	0.86	0.9534		
SIDO0	1452	final submission	1024/4932**	0.6516	0.9432 ± 0.0074	0.9443	0.9467		
SIDO1	1452	final submission	4096/4932**	0.5685	0.7523 ± 0.0137	0.7532	0.7893	0.773	2
SIDO2	1452	final submission	2048/4932**	0.5685	0.6235 ± 0.0129	0.6684	0.7674		
CINA0	1452	final submission	64/132**	0.6	0.9715 ± 0.0032	0.9765	0.9788		
CINA1	1452	final submission	64/132**	0.7053	0.8446 ± 0.0047	0.8691	0.8977	0.8773	1
CINA2	1452	final submission	4/132**	0.7053	0.8157 ± 0.0052	0.8157	0.891		
MARTI0	1452	final submission	256/1024**	0.8073	0.9914 ± 0.0025	0.9996	0.9996		
MARTI1	1452	final submission	256/1024**	0.7279	0.9209 ± 0.0045	0.947	0.9542	0.891	3
MARTI2	1452	final submission	2/1024**	0.9897	0.7606 ± 0.0062	0.7975	0.8273		

- **quantitative advantages:** simplicity
- **qualitative advantages:** SVM feature selection method comparison.

Our implementation consists of python and matlab codes, and the LIBLINEAR software is used to train and predict.

Keywords:

- **Preprocessing or feature construction:** scaling.
- **Feature selection:** filter, feature ranking.
- **Classifier:** SVM.
- **Hyper-parameter selection:** grid-search.

Answers to the organizer's questions

What else did you try besides the method you submitted last? What do you think was a critical element of success compared to other things you tried?

We have tried several approaches. We used a kernel function to measure association between variables, but it takes too long time for a large data set. We also used several methods for predictions, such as Naïve Bayes, Boosting, SVM, Lasso et al. But they may be good for some of data sets but bad for others. Finally we select the L1 penalized logistic regression approach which performed averagely well for all of these data sets. We focused on causal discovery and prediction models, especially we tried to minimize the number of features (ulist) selected for prediction. We should take advantage of a slist of features to improve TScore by chance.

In what do the models for the versions 0, 1, and 2 of the various tasks differ?

The structure learning is all the same to version 0, 1 and 2. But the selection of variables from the learned graph is different in the cases 0, 1 and 2 since they were differently manipulated. Nothing more is different among these three tasks.

Did you rely on the quartile information available on the web site for model selection or did you use another scheme?

The quartile is useful auxiliary information for us to consider whether it is necessary to improve our prediction. But they are not determinate. We check whether our model behaves better or worse by observing the main output indicators. We didn't use any other scheme.

In the result table you submitted, did you use nested subsets of features from the slist you submitted?

We did not use any nested subsets of features from slist, and we used the ulist only.

Did you use any knowledge derived from the test set to make your submissions, including simple statistics and visual examination of the data?

We did not use any knowledge from the test data.

Causation and Prediction Challenge Fact Sheet 6

Title: Boosting Probabilistic Network for causality prediction

Author: Louis Duclos-Gosselin

Address: 205 Gosselin street, St-Agapit, Québec, g0s 1z0, Canada

Email: `louis.gosselin@hotmail.com`

Acronym of your best entry: Bayes Method

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Louis_Duclos-Gosselin.html

Reference

These recent years Bayesian Analysis became a subject of great interest for many practitioners and researchers. The 2008 causality challenge permits me to test one of my best Bayes methods. In fact, I use a special case of boosting probabilistic networks (Bayes Network) controlled with genetic algorithm and simulated annealing. More precisely, the construction of the network works this way. First, I use conventional probabilistic network architecture in conjunction with genetic algorithm and simulated annealing for controlling those elements: learning algorithms (M.A.P. and L.M.), number of neurons, number of links in the network, number of layer, type of kernel, transfer and activation function and the predictors to be in the models. Second, I use the idea of boosting (weighted re sampling) to construct an ensemble of probabilistic network. Third, during the process, Bayes Analysis (prior) helped to produce posterior probability. Finally, the joint distribution between the predictors and the joint distribution between the predictors and the target variable was used.

Method

- **Preprocessing:** Informational theory, entropy.
- **Causal discovery:** Probabilistic networks.
- **Feature selection:** Genetic algorithm and simulated annealing.
- **Classification:** Boosting Probabilistic networks (learned with M.A.P. and L.M.)
- **Model selection/hyper-parameter tuning:** Genetic algorithm, simulated annealing, bayes analysis.

Results

The strength of this method is the use of boosting probabilistic networks controlled with simulated annealing and genetic algorithm. In addition, the uses of bayes analysis add something interesting.

Table I.6: Results table. The star following the feature number indicates that the feature set was sorted. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	361	Bayes Method	66/999*	0.6973	0.9311 ± 0.0040	1	1		
REGED1	361	Bayes Method	10/999*	0.6761	0.8406 ± 0.0054	0.998	0.998	0.7582	14
REGED2	361	Bayes Method	66/999*	0.7317	0.5030 ± 0.0016	0.86	0.9534		
SIDO0	361	Bayes Method	6/4932*	0.5007	0.8956 ± 0.0082	0.9443	0.9467		
SIDO1	361	Bayes Method	25/4932*	0.4994	0.5244 ± 0.0081	0.7532	0.7893	0.6254	10
SIDO2	361	Bayes Method	6/4932*	0.5005	0.4562 ± 0.0059	0.6684	0.7674		
CINA0	361	Bayes Method	106/132*	0.644	0.9337 ± 0.0030	0.9788	0.9788		
CINA1	361	Bayes Method	109/132*	0.6689	0.7419 ± 0.0052	0.8977	0.8977	0.7453	11
CINA2	361	Bayes Method	109/132*	0.6689	0.5602 ± 0.0052	0.8157	0.891		
MARTI0	361	Bayes Method	10/1024*	0.495	0.9196 ± 0.0041	0.9996	0.9996		
MARTI1	361	Bayes Method	2/1024*	0.499	0.6658 ± 0.0060	0.947	0.9542	0.7539	9
MARTI2	361	Bayes Method	2/1024*	0.499	0.6764 ± 0.0062	0.7975	0.8273		

Quantitative advantages

This method is really long to compute, but it has the advantage to explore all the possibility and it uses the full power of bayes analysis.

Qualitative advantages

This method provide a lot of new elements. In fact, the idea of using boosting with probabilistic network is pretty new. In addition, the use of bayes analysis, simulated annealing and genetic algorithm to control all the processes is really special. All this make that method really unique. In brief, this method should be explore by researcher.

This method can be easily implanted in many system with a SAS code, C code or C++ code.

Keywords:

Bayesian Analysis, Bayes Network, Boosting, Causality Prediction, Genetic Programming, Probabilistic Network, Simulated Annealing

- **Preprocessing or feature construction:** Entropy, information theory.
- **Causal discovery:** Bayesian Network, Probabilistic network, boosting.
- **Feature selection:** Genetic algorithm, simulated annealing.
- **Classifier:** boosting probabilistic network.
- **Hyper-parameter selection:** Genetic algorithm, simulated annealing, bayes analysis.
- **Other:** ensemble method.

Causation and Prediction Challenge Fact Sheet 7

Title: Dimensionality reduction through unsupervised learning

Author: Nistor Grozavu (Nist in challenge)

Address: LIPN, Institut Galilée, 99 Av. J.B. Clément, F-93430 Villetaneuse

Email: `nistor_grozavu@yahoo.com`

Acronym of your best entry: Som (by Nist)

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Nistor_Grozavu.html

Method

Profile of my methods:

- **Preprocessing:** Data normalization;
- **Causal discovery:** Self Organizing Maps (SOM – Kohonen Map) adapted for supervised learning; Statistical Test (Cattell Scree Test) using acceleration stop criteria.
- **Feature selection:** Statistical Test (Cattell Scree Test) using acceleration stop criteria for each cluster.
- **Classification:** SOM + CAH (or K-means)

Results

Table I.7: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	1447	Som	4/999	0.498	0.5000 ± 0.0000	1	1		
REGED1	1447	Som	4/999	0.498	0.5000 ± 0.0000	0.998	0.998	0.5	15
REGED2	1447	Som	4/999	0.498	0.5000 ± 0.0000	0.86	0.9534		
SIDO0	137	fd	$16/4932^{**}$	0.5068	0.505 ± 0.0056	0.9443	0.9467		
SIDO1	137	fd	$128/4932^{**}$	0.493	0.5161 ± 0.0068	0.7532	0.7893	0.5098	16
SIDO2	137	fd	$4/4932^{**}$	0.499	0.5076 ± 0.0057	0.6684	0.7674		
MARTI0	1447	Som	71/1024	0.4889	0.5000 ± 0.0000	0.9996	0.9996		
MARTI1	1447	Som	71/1024	0.5011	0.5000 ± 0.0000	0.947	0.9542	0.5	12
MARTI2	1447	Som	71/1024	0.7158	0.5000 ± 0.0000	0.7975	0.8273		

Quantitative advantages

Compact feature subset (71 for MARTI), rapid in cost time.

Qualitative advantages

Elements of novelty: SOM provide a nice visualization; Using Cattell Statistical Test for each cluster we can give a good cluster characterization.

Implementation

I implemented the model in Matlab and I used Statistical Toolbox and SOM Toolbox to facilitate the implementation.

Keywords:

- **Preprocessing or feature construction:** normalization.
- **Causal discovery:** Supervised SOM, Scoring.
- **Feature selection:** statistical test, weighting.
- **Classifier:** neural networks, CAH.

Causation and Prediction Challenge Fact Sheet 8

Title: Markov blanket of the target and Norm1 linear SVM

Author: Cristian Grozea

Address: Fraunhofer Institute FIRST, Kekulestrasse 7, 12489 Berlin, Germany.

Email: `cristian.grozea@first.fraunhofer.de`

Acronym of your best entry: darum

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Cristian_Grozea.html`

Method

- Cina and the toy problems: norm1 linear svm
- Other pbs: norm1 linear svm on the features in the Markov blanket of the target

The features have been ranked by the strength of the corresponding weight in the final classifier. The same model has been applied to all three subproblems. On the problems where the Markov blanket has been used as a feature selection method, the results without these selection were initially bad on the colored quartiles. Hold-out test set has also been used to measure the performance of the training. Nested subsets of features have not been used.

For MARTI (where I have also used at training spatial filters) I have looked at the first few entries in the test set in order to understand the phrases:
The test sets have no added noise. This situation simulates a case where we would be using different instruments at "training time" and "test time", e.g. we would use DNA microarrays to collect training data and PCR for testing.

Erratum

The graphs from `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Cristian_Grozea.html` confirm what I suspected, that is that I sent the wrong features indexes for Marti. What I did was keeping only the estimated Markov blanket features, then running my existing code that reported the index of the features ordered by their absolute weight in the final classifier, but not taking into account the original index of the features. I have to admit that I didn't pay much attention to this as it was seemingly not important for the ranking. For the next problems I preferred to "kill" the unwanted features by zeroing them, such that I wouldn't have to change the code and still get the right indexes. The reason for writing this is to avoid the impression that you could do well with the wrong features.

Implementation

Matlab, CVX and Causal explorer have been used.

Keywords:

- **Preprocessing or feature construction:** causal.
- **Causal discovery:** Markov blanket.
- **Classifier:** SVM, L1 norm regularization.
- **Hyper-parameter selection:** sweep, hold-out test.

Causation and Prediction Challenge Fact Sheet 9

Title: An Energy-based Model for Feature Selection

Author: H. Jair Escalante, Luis Enrique

Address: Erro #1, Tonantzintla, Puebla, 72840, México

Email: hugo.jair@gmail.com, hugojair@ccc.inaoep.mx

Acronym of your best entry: DRF-LM-PSMS Final Run 2

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/H_Jair_Escalante.html

Method

We propose an energy-based (random-field like) model for the selection of predictive features. The user specifies k, the size of the subset of features they want to obtain. Then a random-field with k nodes is defined; each node representing a feature. Each of the k features depends on the other $k-1$ features and on the target variable Y. In Figure I.1 the graphical model of the proposed approach is shown for a value of $k = 6$.

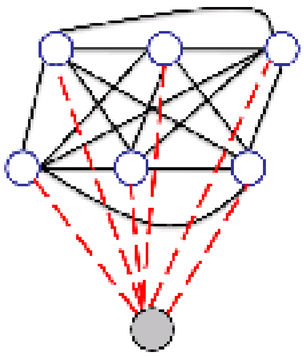


Figure I.1: Graphical model of the proposed EBM for a value of $k = 6$

An energy value is assigned to each combination of k-features, according an energy function. This function assigns low values to *good* configurations of features taking into account the following information:

1. The rank position of each individual feature though the ranking lists returned by eight ranking-based feature selection methods (those in the CLOP package [2]). Different ranking lists are merged considering the position of features along the lists.
2. The predictive power of each individual feature, measured by the CV-balanced error rate obtained by an arbitrary classifier using a single feature for predicting Y.
3. The combined predictive power of the k-features, measured as above using the k-features for predicting Y.

4. Global Markov blanket (MB) information: those features appearing in the MB are weighted higher. The global MB is calculated using all the features in the training data with the Causal Explorer [1].
5. Local MB information: those features appearing in the local MB receive an extra weight. The local MB is calculated using only the k-features in the training data with the Causal Explorer [1].

The feature selection problem reduces to find the configuration of k-features that minimizes the energy function. This configuration will be that offering the best tradeoff among the considered information (1–5) . A simple iterative procedure called iterative conditioned modes (ICM) is used for minimization of the energy function. For those entries containing PSMS in their name we applied particle swarm model selection at the end of the feature selection process. This method is used for searching for the best classifier and hyperparameters for each subset of features k. Therefore, different classifiers were considered for different subset sizes.

I tried several combinations of the sources of information we considered (1–5). The key elements of the proposed approach were the rank of individual features according several feature selection methods (1) and the predictive power of individual features (2). There is not a significant difference (neither positive nor negative) of using only (1–2) or including causal information (1–5). This result is interesting because by simply combining the ranked lists of features from different methods and taking into account the individual predictive power of features we can obtain competitive results. For SIDO the *PC_HITON* algorithm could not be applied because it was running too slow. For CINA this algorithm was not able to infer the MB.

Preprocessing

No preprocessing was applied to data.

Causal discovery

For some experiments I used the PC_HITON implementation ([1]) from the Causal Explorer for obtaining the Markov Blanket of the target variable.

Feature selection

I used the following feature selection methods from the Challenge Learning Object Package (CLOP) (See [2] for a description of these methods):
s2n,gs,relief, svcrfe, aucfs, f-test, t-test, Pearson

Classification

Kernel ridge regression and Naïve Bayes (CLOP implementations) were considered for classification. The latter method was used (extensively) during the optimization process and the former for computing initial and final predictions.

Model selection/hyper-parameter tuning

- For most of the entries, default parameters were considered for the methods above described.
- For a few runs it was used PSMS (a population-based search strategy for model selection) for the selection of a classifier at the end of the feature selection process.

Results

Table I.8: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1485	DRF-LM-PSMS Final Run 2	32/999**	0.8778	0.9996 ± 0.0010	0.9998	1		
REGED1	1485	DRF-LM-PSMS Final Run 2	128/999**	0.7996	0.9448 ± 0.0039	0.9888	0.998	0.8985	5
REGED2	1485	DRF-LM-PSMS Final Run 2	64/999**	0.7638	0.7512 ± 0.0060	0.86	0.9534		
SIDO0	1485	DRF-LM-PSMS Final Run 2	1024/4932**	0.8442	0.9352 ± 0.0075	0.9443	0.9467		
SIDO1	1485	DRF-LM-PSMS Final Run 2	4932/4932**	0.4675	0.6913 ± 0.0134	0.7532	0.7893	0.7474	7
SIDO2	1485	DRF-LM-PSMS Final Run 2	4932/4932**	0.4675	0.6157 ± 0.0128	0.6684	0.7674		
CINA0	1485	DRF-LM-PSMS Final Run 2	132/132**	0.955	0.9670 ± 0.0035	0.9765	0.9788		
CINA1	1485	DRF-LM-PSMS Final Run 2	132/132**	0.4982	0.7873 ± 0.0049	0.8691	0.8977	0.7675	9
CINA2	1485	DRF-LM-PSMS Final Run 2	128/132**	0.4982	0.5481 ± 0.0044	0.8157	0.891		
MARTI0	1485	DRF-LM-PSMS Final Run 2	1024/1024**	0.5446	0.9673 ± 0.0036	0.9996	0.9996		
MARTI1	1485	DRF-LM-PSMS Final Run 2	512/1024**	0.4711	0.8636 ± 0.0054	0.947	0.9542	0.8691	5
MARTI2	1485	DRF-LM-PSMS Final Run 2	8/1024**	0.7055	0.7764 ± 0.0061	0.7975	0.8273		

Quantitative advantages

- It is computationally efficient: by considering the two sources of information that worked well we will obtain competitive results very fast and in a simpler way, that does not requires specialized knowledge.
- The method can be applied to any data set without an ad hoc modification; particularly, the things that worked well (1–2) can be used directly in any binary classification data set.
- The method is easy to implement: even when taking into account all of the sources of information it is not complicated to implement it. Furthermore, the energy-based modeling

framework allows us introducing other sources of information, not considered here, with little effort.

- The method may (or may not) take into account causal information into the feature selection process. Causal information could be very useful for improving the feature selection process.
- It can return subsets of features of size k; the user is able to set this parameter (k). Furthermore, we can return a ranked list of features according their importance.

Qualitative advantages

- The energy-based model we propose is a new way to approach the feature selection problem. Since it is based on the energy-minimization framework it is a very general approach that can be easily modified. The proposed model can, even, be considered a template under which several sources of information and different form of potentials can be tested. This will motivate further research in several directions, particularly on the appropriate ad-hoc definition of potentials and on learning the energy function from data.
- The fusion of the ranking lists of diverse feature selection methods proved to be very useful for feature selection. Information fusion has been proved to be very effective in a number of fields, most notably in machine learning (boosting, bagging) and information retrieval (multi-modal retrieval of video and images). The results obtained by merging different lists give evidence that the fusion of the outputs of diverse feature selection methods has practical advantages that motivate further research.
- Domain knowledge and further information (both causal and non-causal) can be easily introduced into our model, this is also related to the generality of energy-based modeling.
- For this implementation we have used the simpler potentials one can use for this problem and the simplest algorithm for energy minimization (Iterated Conditioned Modes, ICM). Therefore, better results are expected by defining more elaborate potentials and by using faster and better convergence optimization algorithms (e.g. the graph cuts algorithm). Furthermore, fixed classifiers and default hyperparameters have been used in most of the experiments.

Implementation

The method has been implemented in Matlab, it requires the CLOP toolbox and the causal explorer (if causal information is considered). The implementation is very simple and it can be considered a push-button application that can be applied to any domain without a significant modification.

Keywords:

- **Causal discovery:** Markov blanket information, HITON algorithm.
- **Feature selection:** Feature ranking, combination of feature selection methods, s2n, gs, relief, svcrfe, aucfs, f-test, t-test, Pearson.
- **Classifier:** Ridge regression, Naïve Bayes classifier.

- **Hyper-parameter selection:** PSMS.
- **Other:** Energy-based models, random field modeling, ICM.

References

[1] C.F. Aliferis, I. Tsamardinos, A. Statnikov, *HITON: A Novel Markov Blanket Algorithm for Optimal Variable Selection*, In FLAIRS 2003.

[2] A. Safari and I. Guyon, *Quickstart guide for CLOP*. Technical report, Graz University of Technology and Clopinet, May 2006. http://www.ymer.org/research/files/clop/QuickStartV1.0.pdf.

Causation and Prediction Challenge Fact Sheet 10

Title: Translate Binary Variable to Continuous Variable

Author: Jinzhu Jia

Address: School of Mathematical Sciences, Peking University, Beijing, P.R.China, 100871

Email: `jinjinjia@gmail.com`

Acronym of your best entry: Final

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Jinzhu_Jia.html`

Method

- EM Algorithm
- Causal discovery
- Elastic net
- SVM

Quantitative advantages

Simple and fast. After translating the binary variable to continuous variable, we can use modern model selection methods to deal with this problem, such as Lasso, Elastic, etc. All of these method are computationally fast.

Qualitative advantages

Very novel method. I construct the correlation between a binary variable and a continuous variable and then I transform a binary variable into a continuous variable without lose the information between the two variables.

Implementation

For the "Reged" data set, we think that the target variable Y comes from a hidden variable H with a normal distribution and "$Y = 1$" corresponds to "$H > a$" for some fixed real number a. Based on this assumption, we can use EM algorithm to construct the correlation between Y and each of the predictors.

After obtaining the correlation matrix, we run a ridge regression and get the regression coefficients of Y on X, then we construct Y. But the solution of ridge regression is not sparse, then we run elastic net to get a sparse coefficient. Those predictors with non-zero coefficients are our approximated Blank variables and we use these variables to do predictions.

We use the approximated "blanket" variable selected from elastic net and Y to construct a causal network, by the software of TETRAD and then get the parents and children variables of Y.

For data set Reged0, since it is not manipulated, we use all the parents and children variables to do predictions.

For data set Reged1, we use all the parents and those children variables which are not manipulated to do predictions.

For the data set Reged 2, we just use the parents nodes to do predictions, for the son nodes have been changed a lot and thus there is little information to do predictions.

When do predictions, we use three methods: 1.linear regression, 2. SVM 3. SVM regression, and then the three results are used to give a final result. If more than two of the methods give $Y = 1$, then $Y = 1$, or else, $Y = -1$.

This is not a push-button application. Since we have to use the software TETRAD to decide which variables are parents.

Results

Table I.9: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1487	Final	14/999	0.7847	0.9954 ± 0.0010	1	1		
REGED1	1487	Final	11/999	0.748	0.8158 ± 0.0056	0.998	0.998	0.8118	12
REGED2	1487	Final	10/999	0.996	0.6242 ± 0.0053	0.86	0.9534		
SIDO0	311	test	60/4932	0.507	0.8968 ± 0.0082	0.9443	0.9467		
SIDO1	311	test	4932/4932	0	0.5000 ± 0.0028	0.7532	0.7893	0.6123	11
SIDO2	311	test	4932/4932	0	0.4401 ± 0.0027	0.6684	0.7674		

Keywords:

- **Preprocessing or feature construction:** centering, standardization.
- **Causal discovery:** Bayesian Network, Probabilistic Graphical Models.
- **Feature selection:** Penalized regression, Lasso, Elastic net.
- **Classifier:** SVM, least-square, ridge regression, L1 norm regularization, L2 norm regularization, ensemble method.
- **Hyper-parameter selection:** cross-validation, K-fold.

Causation and Prediction Challenge Fact Sheet 11

Title: Univariate feature ranking and SVM classifier

Author: Jianming Jin

Address: HP Labs, China, 112 JianGuo Road, ChaoYang District, HP Building, Beijing, China, 100022

Email: `jian-ming.jin@hp.com`

Acronym of your best entry: HPLC

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Jianming_Jin.html`

Method

Preprocessing

We map the causation and prediction challenge to a document classification task. Here, a dataset is a group of documents, the feature dimension is the size of lexicon, each feature is corresponding to a word in the lexicon, and the value of a feature is the word appearance number in a document (TF).

Feature selection

Weighted feature vector is calculated by multiply the original feature vector with a weighting vector. The weighting value of each dimension is mainly determined by the TF distribution variance in positive training data and negative training data.

Classification

SVMLight is used for training and classification on the weighted feature vectors.

Model selection/hyper-parameter tuning

The model is trained on the provided training set, the training parameters are optimized according to the classification result on the provided testing set. The model with the best F1 value on the provided testing set is chosen as the final model.

There is no special optimization for a peculiar dataset, and there is no need for human participation during the whole process.

Results

Table I.10: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1088	exp 1	999/999	0.5	0.9932 ± 0.0022	1	1		
REGED1	1088	exp 1	999/999	0.5	0.9340 ± 0.0042	0.998	0.998	0.8868	8
REGED2	1088	exp 1	999/999	0.5	0.7331 ± 0.0059	0.86	0.9534		
SIDO0	1089	exp 1	4932/4932	0.5	0.9320 ± 0.0096	0.9443	0.9467		
SIDO1	1089	exp 1	4932/4932	0.5	0.7307 ± 0.0136	0.7532	0.7893	0.7662	3
SIDO2	1089	exp 1	4932/4932	0.5	0.6359 ± 0.0133	0.6684	0.7674		
CINA0	1088	exp 1	132/132	0.5	0.9566 ± 0.0034	0.9788	0.9788		
CINA1	1088	exp 1	132/132	0.5	0.6528 ± 0.0056	0.8977	0.8977	0.6883	13
CINA2	1088	exp 1	132/132	0.5	0.4556 ± 0.0035	0.8157	0.891		
MARTI0	1088	exp 1	1024/1024	0.5	0.8967 ± 0.0047	0.9996	0.9996		
MARTI1	1088	exp 1	1024/1024	0.5	0.7597 ± 0.0060	0.947	0.9542	0.7848	7
MARTI2	1088	exp 1	1024/1024	0.5	0.6979 ± 0.0063	0.7975	0.8273		

Implementation

The implementation is a Java package, which using SVMLight for training and classification. It's a function module without user interface.

Keywords:

- **Preprocessing or feature construction:** none.
- **Causal discovery:** none.
- **Feature selection:** filter, weighting.
- **Classifier:** SVM.
- **Hyper-parameter selection:** grid-search.

Causation and Prediction Challenge Fact Sheet 12

Title: Collider scores

Author: Ernest Mwebaze and John Quinn

Address: Faculty of Computing & Information Technology, Makerere University, Kampala, Uganda.

Email: emwebaze@cit.mak.ac.ug, jquinn@cit.mak.ac.ug

Acronym of your best entry: submission

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/E_MwebazeJ_Quinn.html

Method

Preprocessing

None, raw data used directly.

Causal discovery

HITON_PC used to estimate neighbouring variables. For manipulated datasets we further narrow down the feature set by computing two scores to select strong causes:

1. To score a variable A as a cause of target variable T using supporting variable B_i, use ratio of partial correlation of $(A, B_i|T)$ and correlation of (A, B_i).

2. For the second score, calculate the difference of:

   (a) evidence that target T is a collider for causes A and B_i, looking for high correlation between (A, target) and (B_i, target) and low correlation between (A, B_i)

   (b) evidence that variable A is a collider for causes T and B_i, using the equivalent pattern of correlation.

Both scores are aggregated over the B_i's.

Feature selection

For unmanipulated datasets, use the features estimated to be neighbouring the targets. For manipulated datasets, choose the subset of features with highest mean scores above.

Classification

For REGED, k-nn classification. For SIDO and CINA, shallow decision trees with naive Bayes classifiers in the leaves (single trees only — no ensemble methods).

Results

Estimation of neighbouring variables uses the HITON_PC implementation in the Matlab "Causal Explorer" library. All other code (for calculating scores, learning and classification etc) written in Python using the Numpy libraries.

The scores are simple to implement and quick to calculate (on the order of seconds for all datasets).

The utility of the scores is dependent on the success of estimating variables which are neighbours to the target. The inclusion of other variables, particularly outside the Markov blanket, can confound the result.

Table I.11: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	1444	submission	14/999**	0.8088	0.9933 ± 0.0014	0.9998	1		
REGED1	1444	submission	1/999**	0.7122	0.9528 ± 0.0028	0.9888	0.998	0.8559	7
REGED2	1444	submission	14/999**	0.9935	0.6216 ± 0.0019	0.86	0.9534		
SIDO0	1444	submission	10/4932**	0.5003	0.9325 ± 0.0074	0.9443	0.9467		
SIDO1	1444	submission	6/4932**	0.5009	0.6660 ± 0.0133	0.7532	0.7893	0.7509	6
SIDO2	1444	submission	6/4932**	0.5009	0.6541 ± 0.0131	0.6684	0.7674		
CINA0	1444	submission	8/132**	0.7575	0.9430 ± 0.0033	0.9765	0.9788		
CINA1	1444	submission	46/132**	0.5885	0.7381 ± 0.0047	0.8691	0.8977	0.7832	8
CINA2	1444	submission	8/132**	0.5235	0.6685 ± 0.0042	0.8157	0.891		

Keywords:

- **Preprocessing or feature construction:** none.
- **Causal discovery:** Structural Equation Models, heuristic.
- **Feature selection:** feature ranking.
- **Classifier:** nearest neighbors, tree classifier, naive Bayes.
- **Hyper-parameter selection:** cross-validation.

Causation and Prediction Challenge Fact Sheet 13

Title: Random Sets Approach and its Applications

Author: Vladimir Nikulin

Address: Suncorp, Brisbane, Australia

Email: v.nikulin@uq.edu.au

Acronym of your best entry: *"vn14"* and *"vn14a"* (for SIDO)

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Vladimir_Nikulin.html

Complete paper

Random Sets Approach and its Applications. Vladimir Nikulin; JMLR W&CP 3:65-76, 2008.

Introduction

It is a well known fact that for various reasons it may not be possible to theoretically analyze a particular algorithm or to compute its performance in contrast to another. The results of the proper experimental evaluation are very important as these may provide the evidence that a method outperforms alternative approaches.

Feature selection represents a very essential component of data mining, as it will help reduce overfitting and make prediction more accurate. Causal discovery may be regarded as a next step with aim to uncover causal relations between features and target variable. In many cases it is theoretically impossible to solve full graphical structure of all relations between features and target variable but it may be possible to uncover and approximate some essential relations. This knowledge will help to understand data better and will give some hints which methods will be more efficient.

Method

Random sets approach has heuristic nature and has been inspired by the growing speed of computations. For example, we can consider large number of classifiers where any single classifier (base classifier or model) is based on the subset of relatively small number of randomly selected features or random sets of features. Using cross-validation we can rank all random sets according to the selected criterion, and use this ranking for further feature selection.

In the case of SIDO-set, Random Forest model proved to be the most suitable. Note that RF model appears to be very relevant to the subject of this paper. However, approach of RF is far from the same comparing with RS approach. We used RF model with 1000 trees where 70 randomly selected features were used for any splitter. The splitting process was stopped if size of the current node was smaller than 10 (anyway, no more than 8 splitting levels were used). The SIDO-set is binary, and this property simplified implementation of the RF-algorithm essentially. Next, we computed for any particular feature number of repeats in the above RF-

Table I.12: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants. These entries used unsorted lists of features.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	1340	vn14	400/999	0.7576	0.9989 ± 0.0016	0.9998	1		
REGED1	1340	vn14	400/999	0.7316	0.9522 ± 0.0038	0.9888	0.998	0.9094	4
REGED2	1340	vn14	400/999	0.8004	0.7772 ± 0.0059	0.86	0.9534		
SIDO0	1479	vn14a	527/4932	0.5502	0.9429 ± 0.0075	0.9443	0.9467		
SIDO1	1479	vn14a	527/4932	0.5339	0.7192 ± 0.0138	0.7532	0.7893	0.7588	5
SIDO2	1479	vn14a	203/4932	0.5225	0.6143 ± 0.0132	0.6684	0.7674		
CINA0	1340	vn14	50/132	0.7174	0.9764 ± 0.0031	0.9765	0.9788		
CINA1	1340	vn14	30/132	0.5	0.8617 ± 0.0047	0.8691	0.8977	0.8504	2
CINA2	1340	vn14	30/132	0.5	0.7132 ± 0.0043	0.8157	0.891		
MARTI0	1340	vn14	217/1024	0.5863	0.9889 ± 0.0025	0.9996	0.9996		
MARTI1	1340	vn14	400/1024	0.5554	0.8953 ± 0.0048	0.947	0.9542	0.8736	4
MARTI2	1340	vn14	611/1024	0.7021	0.7364 ± 0.0062	0.7975	0.8273		

Table I.13: List of the base models, which were used during WCCI-2008 data-mining competition.

Data	**Model**	**Software**
LUCAS	neural+doubleboost	MATLAB-CLOP
LUCAP	neural+doubleboost	MATLAB-CLOP
REGED	SVM-RBF (special software)	C
SIDO	binaryRF (special software)	C
CINA	adaBoost	R
MARTI	svc+standardize	MATLAB-CLOP

object. These repeats were used for further feature selection. For example, we used in the final submission 1030 features for SIDO0, 517 features for SIDO1 and only 203 features for SIDO2.

Preprocessing

The case of MARTI-set appears to be the most complicated because of the 25 given calibrants: the training set was perturbed by a zero-mean correlated noise model. The test sets have no added noise. We used linear regression model in order to filter noise from the training set. As a target variables we used remaining 999 features.

Another application of random sets was motivated by the huge imbalanced data, which represent significant problem because the corresponding classifier has tendency to ignore patterns with smaller representation in the training set. We propose to consider large number of balanced training subsets where representatives from both patterns are selected randomly.

Keywords:

Causal relations, graphical models, random forest, boosting, svm, CLOP, cross validation.

Causation and Prediction Challenge Fact Sheet 14

Title: Optimally Compressive Regression

Author: Florin Popescu

Address: Fraunhofer-Institut FIRST, Kekulestrasse 7, 12489 Berlin Germany

Email: `florin.popescu@first.fraunhofer.de`

Acronym of your best entry: optimally_compressive_regression

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Florin_Popescu.html`

Reference

Florin Popescu. Identification of Sparse Multivariate Autoregressive Models. EUSIPCO 2008 (in press).

Method

Models (Gaussian probability distributions) are derived each class using auto-regression and minimum description length (MDL) regularization, where details of MDL have been derived by the author to allow mixtures of binary and non-binary valued data. Initially the method was meant for classification of time series data, but is applicable also to stationary data sets by essentially building large, sparse covariance matrices. MDL sparsifies automatically and in itself is a conduit for causality inference: the best explanation (e.g. causal chain) is the one that compresses the data the most.

Preprocessing

The features were scaled such that their quantization level is 1.

Causal discovery

Linear regressions were done to make a list of predictability of each feature (how compressible it is given knowledge of all other features: a *full* regression). MDL gives sparse results so each feature thereby has a set of predictor features.

Feature selection

The union of the best $X\%$ predicted variables and all necessary predictors ordered by predictability.

Model selection/hyper-parameter tuning

The model (for each class) was a strictly upper triangular auto-regression (AR) matrix between selected features (with bias and scaling). This is called *causal* regression because it enforces a causal chain. The feature set was *sorted* by the resulting predictability and the causal regression re-computed until the predictability is strictly increasing. The MDL regression sparsifies the AR matrix and therefore may further reduce the feature set – it also produces a directed *acyclic* graph of causal factors (the causal chain). The method "works" with the parameter X set to 0 (naïve method: meaning only predictors are used) but it was set at a higher level. By the MDL principle it is the final MDL score that counts: hyperparameters such as X only serve to make the necessary (non-convex) MDL optimization faster or more likely to reach the global minimum, they do not embody a statistical principle *per se*. There is no cross-validation.

Classification

Once the class models are obtained, the classification is trivial: each new example corresponds to a likelihood (or probability) determined by the model, and the label is the class of the highest likelihood model.

Results

Table I.14: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants. These entries used unsorted lists of features.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
CINA0	1495	optimally compressive regression	25/132*	0.6087	0.7769 ± 0.0051	0.9788	0.9788		
CINA1	1495	optimally compressive regression	25/132*	0.7639	0.7322 ± 0.0052	0.8977	0.8977	0.7488	10
CINA2	1495	optimally compressive regression	25/132*	0.7639	0.7372 ± 0.0052	0.8157	0.891		

- **quantitative advantages:** The feature set is automatically generated and for the naïve method was very compact (11 features selected for CINA0)
- **qualitative advantages:** The method, as explained, computes posterior probabilities as well as a directed causal chain, is theoretically motivated, can be fully automatic and is (to the author's knowledge) fully novel.

The method was implemented in Matlab and C. The linear regressions were performed using standard methods (albeit organized such that large regressions can be done within memory limitations). The MDL optimization is computationally expensive but is *not* an exhaustive feature subset modeling technique, rather it is programmed using iterative heuristics for sparsification of features and structural models.

Keywords:

- **Preprocessing or feature construction:** scaling.
- **Causal discovery:** Structural Equation Models, Probabilistic Graphical Models, Information Theoretic Method.
- **Feature selection:** filter, feature ranking.
- **Classifier:** likelihood-based.
- **Hyper-parameter selection:** none.
- **Other:** minimum description length.

Causation and Prediction Challenge Fact Sheet 15

Title: Markov blanket and kernel ridge regression

Author: Marius Popescu

Address: University of Bucharest Department of Computer Science Academiei 14, 70109 Bucharest, Romania

Email: `popescunmarius@gmail.com`

Acronym of your best entry: MB_Kcomb1

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Marius_Popescu.html`

Method

I approached the challenge from the position of someone with experience in machine learning, but a completely newcomer in causality. As learning method I used Kernel Ridge Regression. For prediction (and training) I used only features from the Markov Blanket (MB) of the target variable, but I also tried to exploit the structure of the MB. The structure of MB was exploited by defining two separate kernels: one over the parents (direct causes) and another one over the children (direct effects) and spouses. The kernel used in Ridge regression was a linear combination of these two kernels.

Obtaining MB and its structure

To obtain the Markov blanket and its structure I relied on "Causal Explorer". To obtain the variables of the MB I used HITON_MB method. To obtain the structure I used TPDA method over the variables in the MB and target variable. Because TPDA can leave some edges undirected and because MB is not a general Bayesian network (it has a special structure around the target node) I used the following heuristic to direct all the edges: all nodes for which TPDA find a directed link from target node to it (1 in the adjacency matrix) are considered children, all others nodes for which TPDA find connection (1 or 2 in the adjacency matrix) to target node are considered parents node, remaining nodes are considered spouses. This heuristics prefers to introduce false parents than to miss some parents (we consider direct causes very important). The details can be seen in the following MATLAB script (`p` is the index of parents, `c` the index of children and `s` the index of spouses):

```
load reged0_train
Y = load('reged0_train.targets');
Y = 0.5*(Y+1);
XY= [X,Y];
mb=Causal_Explorer('HITON_MB', XY, 1000, [], 'z', 0.05, 3)
XY2 = XY(:, [mb,1000]);
A=Causal_Explorer('TPDA', XY2, [], 'z', 0.05, 1, 1)

ic = find(A(end, 1:end-1) == 1);
```

```
ip = find(A(1:end-1, end) ~= 0);
is = setdiff(1:length(mb), union(ip,ic));
p = mb(ip);
c = mb(ic);
s = mb(is);

save mb mb A ic ip is p c s;
```

Training and Prediction

The kernel used in the Kernel Ridge Regression was a linear combination of two quadratic (normalized) kernels:

```
K = (eta * K1) + ((1 - eta) * K2)
```

K1 being a quadratic normalized kernel over the set of parents:

```
K1 = (X(:, p) * X(:, p)' + 0.5) .^ 2;
XN1 = sqrt(diag(K1));
K1 = K1 ./ (XN1 * XN1');
```

And K2 a quadratic normalized kernel over the set of children and spouses:

```
K2 = (X(:, [c,s]) * X(:, [c,s])' + 0.5) .^ 2;
XN2 = sqrt(diag(K2));
K2 = K2 ./ (XN2 * XN2');
```

The parameter `eta` of the linear combination is chosen taking into account how “good” is each kernel of the combination. The “goodness” is measured by “kernel alignment”, more precisely the alignment of the kernel with the ideal kernel `YY'`. Again the details (including the setting of parameters) can be seen in training MATLAB script:

```
lambda = 10 ^ (-6);

load lim;
load reged0_train
X = scale(X, ll, ul);
Y = load('reged0_train.targets');

load mb;

n = size(X, 1);

K1 = (X(:, p) * X(:, p)' + 0.5) .^ 2;
XN1 = sqrt(diag(K1));
K1 = K1 ./ (XN1 * XN1');

K2 = (X(:, [c,s]) * X(:, [c,s])' + 0.5) .^ 2;
XN2 = sqrt(diag(K2));
K2 = K2 ./ (XN2 * XN2');

align1 = (Y' * K1 * Y) / (n * norm(K1, 'fro'))
```

```
align2 = (Y' * K2 * Y) / (n * norm(K2, 'fro'))
eta = align1 / (align1 + align2);

K = (eta * K1) + ((1 - eta) * K2);

w = inv(K+(n*lambda)*eye(n)) * Y;

Yh = K * w;
%Yh = sign(Yh);
%Yh(find(Yh == 0)) = 1;

%err = length(find(Yh ~= Y)) / n

fid = fopen('reged0_feat.ulist','w');
fprintf(fid,'%g ',[p,c,s]);
fprintf(fid,'\n');
fclose(fid);

fid = fopen('reged0_train.predict','w');
fprintf(fid,'%g\n',Yh);
fclose(fid);

save model w eta XN1 XN2;
```

Treating manipulation

When the list of manipulated variables is available (REGED1) from the MB are removed the children of the target that are manipulated. Also are removed all the spouses that remain without children. The script is:

```
load mb;

tbelm = [20, 27, 36, 70, 82, 83, 85, 91, 118, 125, 139, 143,
160, 169, 176, 185, 191, 204, 219, 224, 229, 239, 243, 251,
252, 269, 281, 282, 295, 297, 301, 319, 320, 321, 342, 350,
357, 359, 361, 378, 387, 407, 409, 412, 429, 430, 469, 472,
499, 501, 507, 512, 540, 545, 552, 561, 566, 572, 580, 586,
593, 618, 622, 637, 651, 663, 674, 681, 683, 686, 690, 702,
727, 754, 762, 764, 773, 786, 805, 815, 835, 861, 872, 873,
877, 880, 889, 904, 935, 936, 939, 942, 949, 962, 977, 985,
989, 991, 992, 994];

se = intersect(tbelm,s);
ce = intersect(tbelm,c);

[tmp, ise] = ismember(se, mb);
[tmp, ice] = ismember(ce, mb);

ice2 = [];
```

```
for x = ise
   ice2 = union(ice2, find(A(ic,x) ~= 0)');
end

ice2 = ic(ice2);
ice = union(ice, ice2);

ictmp = setdiff(ic, ice);

ise2 = [];

for x = is
   if isempty(find(A(ictmp,x) ~= 0))
     ise2 = [ise2, x];
   end
end

ise = union(ise, ise2);

icn = setdiff(ic, ice);
isn = setdiff(is, ise);

cn = mb(icn);
sn = mb(isn);

save newmb p cn sn;
```

In the case of REGED2 (when all variable excepting parents are manipulated) that mean that will remain only one kernel (instead of a combination of two kernels), the quadratic kernel over the set of parents.

Results

Table I.15: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	77	MB_Kcomb1	24/999	0.9012	0.9931 ± 0.0017	1	1		
REGED1	77	MB_Kcomb1	11/999	0.7842	0.9888 ± 0.0026	0.998	0.998	0.9032	5
REGED2	77	MB_Kcomb1	6/999	0.7475	0.7278 ± 0.0060	0.86	0.9534		

Keywords:

- **Preprocessing or feature construction:** centering, scaling, standardization, PCA.

- **Causal discovery:** Bayesian Network, Structural Equation Models, Probabilistic Graphical Models, Markov Decision Processes, Propensity Scoring, Information Theoretic Method.
- **Feature selection:** filter, wrapper, embedded feature selection, feature ranking, etc.
- **Classifier:** neural networks, nearest neighbors, tree classifier, RF, SVM, kernel-method, least-square, ridge regression, L1 norm regularization, L2 norm regularization, logistic regression, ensemble method, bagging, boosting, Bayesian, transduction.
- **Hyper-parameter selection:** grid-search, pattern search, evidence, bound optimization, cross-validation, K-fold.
- **Other:** ensemble method, transduction.

Causation and Prediction Challenge Fact Sheet 16

Title: Markov Blanket Filtering using Mixture Models

Author: Mehreen Saeed

Address: FAST National University of Computer & Emerging Sciences, Lahore Campus, Pakistan.

Email: mehreen.saeed@nu.edu.pk

Acronym of your best entry: Final Entry 2

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Mehreen_Saeed.html

Complete paper

Bernoulli Mixture Models for Markov Blanket Filtering and Classification. Mehreen Saeed; JMLR W&CP 3:77–91, 2008.

Method

Preprocessing

Normalize and Standardize

Causal discovery

Markov blanket filtering with Bernoulli mixtures in case of SIDO. In case of CINA this method was only applied to binary features but this was not our last submitted entry.

Feature selection

Subset feature selection using forward selection algorithm. The heuristic used to guide the search was the accuracy obtained from the Naïve Bayes' classifier.

Classification

Naïve Bayes' classifier, Bernoulli mixtures + ensemble of neural nets in case of SIDO and ensemble of neural nets in case of CINA,

Model selection/hyper-parameter tuning

Cross validation.

Results

Table I.16: Results table. The two stars next to the feature number indicate that the submission included a sorted list of features and multiple results for nested subsets of features. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
SIDO0	1413	final entry 2	8/4932**	0.4971	0.9391 ± 0.0079	0.9443	0.9467		
SIDO1	1413	final entry 2	4096/4932**	0.7242	0.7246 ± 0.0137	0.7532	0.7893	0.7618	3
SIDO2	1413	final entry 2	512/4932**	0.7242	0.6216 ± 0.0132	0.6684	0.7674		
CINA0	1413	final entry 2	32/132**	0.5069	0.9751 ± 0.0031	0.9765	0.9788		
CINA1	1413	final entry 2	16/132**	0.7858	0.8248 ± 0.0045	0.8691	0.8977	0.8289	6
CINA2	1413	final entry 2	16/132**	0.7858	0.6867 ± 0.0041	0.8157	0.891		

Quantitative advantages

- FOR SIDO: Feature transformation to new probability space significantly reduces the dimensionality of data, use of Bernoulli mixtures for Markov blanket filtering drastically reduces the computational cost.
- FOR CINA: Subset feature selection using forward selection algorithm in CINA is extremely simple and intuitive.

Qualitative advantages

Methods based upon mixture densities which model the data very effectively.

Implementation

Code was implemented by adding modules to CLOP's library. C++ code was written for mixture models with an interface to Matlab.

Keywords:

- **Preprocessing or feature construction:** probability space provided by mixture models, normalization, standardization
- **Causal discovery:** Markov blanket filtering using Bernoulli mixtures
- **Feature selection:** hill climbing search.
- **Classifier:** neural networks, ensemble method, Naïve bayes', mixture models.
- **Hyper-parameter selection:** 2-fold cross validation.
- **Other:** None.

Other methods Tried

SVM, Bernoulli mixtures combined with SVM and simple neural networks.

What do you think was a critical element of success compared to other things you tried?

Quartile information gave a big clue as to which method was performing better. For some methods CV accuracy was a little misleading on versions 1 and 2 of datasets as it doesn't tell us much about a dataset where the features have been manipulated by external sources.

In what do the models for the versions 0, 1, and 2 of the various tasks differ?

SIDO: Models 1 and 2 were created with the same method and models, i.e., Markov blanket filtering using Bernoulli mixtures was used to find a feature list. Model 0 was created with subset feature selection using forward algorithm. Classification was done using Naïve Bayes'.

CINA: In case of CINA all versions were created with the same model.

Did you rely on the quartile information available on the web site for model selection or did you use another scheme?

Yes, we did use quartile information for model selection. Within a quartile we used 2 fold cross validation accuracy.

In the result table you submitted, did you use nested subsets of features from the slist you submitted?

Yes.

Did you use any knowledge derived from the test set to make your submissions, including simple statistics and visual examination of the data?

No.

Causation and Prediction Challenge Fact Sheet 17

Title: Ensemble Machine Learning Method

Author: Ching-Wei Wang

Address: Department of Computing and Informatics University of Lincoln Brayford Pool Lincoln LN6 7TS United Kingdom

Email: `cweiwang@lincoln.ac.uk`

Acronym of your best entry: c, 10, 15

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Ching-Wei_Wang.html`

References

1. Wang, C.-W. (2006) New Ensemble Machine Learning Method for Classification and Prediction on Gene Expression Data, Proceedings of the 28th international conference of the IEEE Engineering in Medicine and Biology Society. pp. 3478–3481. ISBN 1424400333.

2. Fayyad, U. M. & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning, 13th International Joint Conference of Artificial Intelligence, 1022–7.

3. Weka, `http://www.cs.waikato.ac.nz/ml/weka/`

Implementation

The learning algorithm is previously implemented in Java and can be referred to [1]. The feature selection algorithms can be found in the weka machine learning package. Portion of binary discretization for testing data is implemented in c#.

Results

Table I.17: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1240	c	999/999	0	0.9938 ± 0.0013	1	1		
REGED1	1240	c	999/999	0	0.9022 ± 0.0044	0.998	0.998	0.8512	10
REGED2	1240	c	999/999	0	0.6577 ± 0.0057	0.86	0.9534		
SIDO0	455	10	4932/4932	0	0.6259 ± 0.0117	0.9443	0.9467		
SIDO1	455	10	4932/4932	0	0.5023 ± 0.0021	0.7532	0.7893	0.5402	14
SIDO2	455	10	4932/4932	0	0.4925 ± 0.0006	0.6684	0.7674		
CINA0	599	15	132/132	0	0.9246 ± 0.0034	0.9788	0.9788		
CINA1	599	15	132/132	0	0.6778 ± 0.0052	0.8977	0.8977	0.7249	12
CINA2	599	15	132/132	0	0.5722 ± 0.0050	0.8157	0.891		
MARTI0	599	15	1024/1024	0	0.6828 ± 0.0056	0.9996	0.9996		
MARTI1	599	15	1024/1024	0	0.6294 ± 0.0058	0.947	0.9542	0.6527	10
MARTI2	599	15	1024/1024	0	0.6459 ± 0.0060	0.7975	0.8273		

Keywords:

- **Preprocessing or feature construction:** binary discretization.
- **Causal discovery:** Information Gain, Decision Tree.
- **Feature selection:** filter.
- **Classifier:** ensemble method, boosting, cw-boost.
- **Hyper-parameter selection:** 10-fold cross-validation.
- **Other:** ensemble method.

Method

Model selection/hyperparameter selection

10-fold cross validation.

	Preprocessing	Feature select	Classification
Reged0, 1, 2	Binary discretization[2] to change feature values and filter out the features, which can not be discretized. Produce 10 features to train. Features[*] include {82, 250, 320, 408, 452, 592, 738, 824, 929, 938}. The features are selected using reged0_train.data.		cw-Boost[1], containing one hundred C4.5 decision trees
Cina0	none	Use raw data	cw-Boost[1], containing seventy C4.5 decision trees (100 iterations may perform better)
Cina1, 2	Feature selection	[3]: -E weka.attributeSelection. CfsSubsetEva –S weka.attributeSelection. BestFirst –D 1 –N 5	
Marti0	Apply Feature selection 1 times, producing 295 features to train.	[3]: -E weka.attributeSelection. CfsSubsetEva –S weka.attributeSelection. GeneticSearch –Z 20 –G 20	
Marti1, 2	Apply Feature selection 3 times, producing 14 features to train.		
Sido0, 1, 2	The data is too big for the equipment (CPU 2.4GHz and 1G RAM only). Systematically divide the data to 10 sub-files, and apply feature selection to the first two files. Train the two files and combine the prediction results. The performance is poor since the data preprocessing is poor here due to limitation of PC memory.	AttributeSelection [3]: -E weka.attributeSelection. CfsSubsetEva –S weka.attributeSelection. BestFirst –D 1 –N 5	

[*]feature index: start from 0

Causation and Prediction Challenge Fact Sheet 18

Title: Partial Orientation and Local Structural Learning of DAGs for Prediction

Author:

Jianxin Yin	`jianxinyin@gmail.com`
Zhi Geng	`zgeng@math.pku.edu.cn`
You Zhou	`zhouyou@pku.edu.cn`
Changzhang Wang	`changzhang@pku.edu.cn`
Ping He	`sunhp@pku.edu.cn`
Cheng Zheng	`zzhengccheng@pku.edu.cn`
Zheyu Wang	`wangzy853@sohu.com`
Simeng Han	`hansimeng@pku.edu.cn`
Lingzhou Xue	`michaelxlz@gmail.com`
Shaopeng Wang	`wangshop@gmail.com`
Zhenguo Wu	`wuzhenguo@gmail.com`
Wei Yan	`yanwei1982@pku.edu.cn`
Manabu Kuroki	`mkuroki@sigmath.es.osaka-u.ac.jp`
Zhihong Cai	`cai@pbh.med.kyoto-u.ac.jp`

Address: School of Mathematical Sciences, Peking University, Beijing 100871, China

Acronym of your best entry: final submission

Performance graphs generated by the organizers: `http://clopinet.com/isabelle/Projects/WCCI2008/Reports/J_YinZ_Geng_Gr.html`

Complete paper:

Partial orientation and local structural learning of causal networks for prediction. Jianxin Yin, You Zhou, Changzhang Wang, Ping He, Cheng Zheng, and Zhi Geng; JMLR W&CP 3:93–105, 2008.

Method

Preprocessing

For the case with noise (e.g., MARTI), we filter the noise using a two steps process. At the first step, we correct the global noise pattern. We find a regression model for each of 999 gene expression features, in which 25 calibrate features, are treated as explanatory variables and the gene expression as the response variable. At the second step, we locally filter the noise of each gene expression feature with neighbor features. Given a new micro-array, we first correct it with the global regression models, and then filter its noises with local models.

For a data set with very high dimensional space (e.g., SIDO), we first screen features using sure independence screening method to reduce the dimensionality to a tractable size (e.g., 1000 features for SIDO). This screen step is not necessary for other data sets and even for a higher dimensional data set if the CPU time or memory for the following computations is not a problem.

Causal discovery

We propose an approach for local structural learning and partial orientation of the edges connected to the target. In the approach, we first find the parent-children set $PC(T)$ of the target T, and then we find the $PC(X)$ for each feature $X \in PC(T)$. We find local v-structures and try to orient the edges connected to the target T as much as possible. If all of the edges connected to the target T are oriented, we obtain all causal relationships to the target T that are necessary for prediction. If some edges have not oriented yet, we can repeat the process to find $PC(X)$ for other features $X \in PC(PC(T))$ until all edges connected to T are oriented or we have checked all features or we have tried the maximum number of steps that we set for a very large graph. Theoretically we can show that the proposed approach is correct, that is, it can correctly find at each step local v-structures of the global DAG.

Feature selection

For the data set without manipulation (numbered 0), all the variables in the Markov blanket (MB) are used to predict the target T. For the data set with a known manipulated variable set (numbered 1), we drop the manipulated variables in the set of children and drop the spouses of T whose children common with T have been all dropped, and we use all parent variables and unmanipulated children and the parents of unmanipulated children in the MB of T. For the data set with an unknown manipulated variable set (numbered 2), only the parent variables of the target are used. When the variable sets that are used for prediction are sensitive to significance levels and other parameters, we may use a union of these sets and then predict the target with a shrinkage method to remove the redundant variables.

Classification

We use the L1 penalized logistic regression model to fit the prediction model. We use the estimated conditional probability of the target variable for each individual in the test set for its classification.

Model selection/hyper-parameter tuning

Given the variable set obtained at the causal discovery, we use a penalized approach which implements both estimation and selection. In the penalized approach, we use a 5-fold of cross validation (CV) method only with the training data set to select the hyper-parameter λ in the solution path. We use the CV curve to diagnose the stableness of the selected model. Three main values that are recorded every time for comparison are the minimal value of CV error, the corresponding norm fraction and the ratio of the selected variable set to the candidate variable set.

Results

Quantitative Advantages

For causal discovery, the approach of partial orientation and local structural learning can greatly reduce computational complexity of structural learning. On the other hand, statistical test is more powerful for the local structural learning approach than the global learning. For the prediction, we use the L1 penalized generalized logistic model to shrink the parameters at the training stage, which can reduce mean squared error (MSE) of prediction.

Table I.18: Results table. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	Entry	Method	Fnum	Fscore	Tscore (Ts)	Top Ts	Max Ts	<Tscore>	Rank
REGED0	1475	final submission	15/999	0.8571	0.9997 ± 0.0010	0.9998	1		
REGED1	1475	final submission	14/999	0.8189	0.9517 ± 0.0033	0.9888	0.998	0.9133	3
REGED2	1475	final submission	11/999	0.9955	0.7885 ± 0.0056	0.86	0.9534		
SIDO0	1475	final submission	16/4932	0.5019	0.9443 ± 0.0075	0.9443	0.9467		
SIDO1	1475	final submission	16/4932	0.5035	0.6976 ± 0.0137	0.7532	0.7893	0.7609	4
SIDO2	1475	final submission	16/4932	0.5035	0.6408 ± 0.0132	0.6684	0.7674		
CINA0	1475	final submission	22/132	0.5957	0.9736 ± 0.0032	0.9765	0.9788		
CINA1	1475	final submission	24/132	0.5852	0.8577 ± 0.0047	0.8691	0.8977	0.833	4
CINA2	1475	final submission	18/132	0.5852	0.6676 ± 0.0044	0.8157	0.891		
MARTI0	1475	final submission	11/1024	0.689	0.9985 ± 0.0016	0.9996	0.9996		
MARTI1	1475	final submission	11/1024	0.6394	0.8911 ± 0.0050	0.947	0.9542	0.8955	2
MARTI2	1475	final submission	11/1024	0.9956	0.7969 ± 0.0060	0.7975	0.8273		

Qualitative Advantages

The approach of partial orientation and local structural learning is efficient for large causal networks if we are interested only in the prediction of the target. We can theoretically show that the approach can correctly obtain the edges connected to the target and their orientations. Although the Markov blanket is useful for prediction without manipulation, it cannot be used for prediction with manipulation, and more importantly it is not sufficient to orient the edges connected to the target. The two stage filtering is efficient for the case with noise and calibrates features. The sure independence screening is a useful preprocess for ultra-high dimensional feature space.

Keywords:

- **Preprocessing or feature construction:** Regression model, Global and local filtering.
- **Causal discovery:** Causal networks, Directed acyclic graphs, Local structural learning, Partial orientation.
- **Feature selection:** Parents and children, Feature ranking, Markov blanket.
- **Classifier:** L1 penalization, Logistic regression, Solution path.
- **Hyper-parameter selection:** Cross validation, K-fold.

Causation and Prediction Challenge Fact Sheet 19

Title: Causative Feature Selection by PC Algorithm and SVMs

Author: Wu Zhili

Address: OEW801, Department of Computer Science, HKBU, HK

Email: vincent@comp.hkbu.edu.hk

Acronym of your best entry: temp 7

Performance graphs generated by the organizers: http://clopinet.com/isabelle/Projects/WCCI2008/Reports/Wu_Zhili.html

Method

Preprocessing

We divide each column of feature by the maximum value.

Feature selection

We test the pcalg package in R, which as an implementation to PC algorithm can provide us a MB set. We also try the HITON_MB and HITON_PC in CausalityExplorer package. They can produce a similar causative feature set. For REGED data, the MB set we obtained leads to an improved Fscore. But for MARTI, the MB set returned doesn't help much. In early submission, feature selection based on weights calculated from linear SVMs is conducted preliminarily, like the submission for SIDO. We did not use any knowledge derived from the test set to make the submissions.

Classification

We use standard SVM to train the final classifier. RBF and high-degree polynomial kernels are used.

Model selection/hyper-parameter tuning

Cross validation with the criteria of balanced error rate is used, but not intensively conducted. Though we use separate weights for the unbalanced positive and negative classes, we haven't gain much performance improvement for these by model selection.

Results

- **Quantitative advantages:** easy to try based on existing packages.
- **Qualitative advantages:** explore the combination of MB selection with SVM.

Our implementation consists of Matlab, R, and also utilizes the LibSVM package.

Table I.19: Results table. The star following the feature number indicates that the feature set was sorted. Top Ts refers to the best score among all valid last entries made by participants. Max Ts refers to the best score reachable, as estimated by reference entries using the knowledge of true causal relationships not available to participants.

Dataset	**Entry**	**Method**	**Fnum**	**Fscore**	**Tscore (Ts)**	**Top Ts**	**Max Ts**	**<Tscore>**	**Rank**
REGED0	1051	temp 7	73/999*	0.8758	0.9820 ± 0.0012	1	1		
REGED1	1051	temp 7	73/999*	0.8213	0.8454 ± 0.0051	0.998	0.998	0.8117	13
REGED2	1051	temp 7	73/999*	0.9564	0.6077 ± 0.0056	0.86	0.9534		
SIDO0	529	wzl-03-27-v3	128/4932*	0.5021	0.6446 ± 0.0122	0.9443	0.9467		
SIDO1	529	wzl-03-27-v3	4932/4932*	0.5088	0.4994 ± 0.0002	0.7532	0.7893	0.5475	13
SIDO2	529	wzl-03-27-v3	4932/4932*	0.5088	0.4985 ± 0.0003	0.6684	0.7674		
CINA0	979	temp 2	66/132*	0.7504	0.9210 ± 0.0033	0.9788	0.9788		
CINA1	979	temp 2	66/132*	0.492	0.6233 ± 0.0048	0.8977	0.8977	0.6668	14
CINA2	979	temp 2	66/132*	0.492	0.4562 ± 0.0042	0.8157	0.891		
MARTI0	988	temp 3	19/1024*	0.4905	0.6541 ± 0.0058	0.9996	0.9996		
MARTI1	988	temp 3	19/1024*	0.4906	0.6411 ± 0.0060	0.947	0.9542	0.6488	11
MARTI2	988	temp 3	19/1024*	0.4907	0.6513 ± 0.0062	0.7975	0.8273		

Keywords:

- **Preprocessing or feature construction:** scaling.
- **Feature selection:** MB and PC algorithm, feature ranking.
- **Classifier:** SVM.
- **Hyper-parameter selection:** cross validation.

Appendix II

Technical Report Describing the Datasets of the Challenge

Datasets of the Causation and Prediction Challenge
The Causality Workbench Team

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet, California

Constantin Aliferis CONSTANTIN.ALIFERIS@NYUMC.ORG
New York University, New York

Greg Cooper GFC@PITT.EDU
University of Pittsburgh, Pennsylvania

André Elisseeff AEL@ZURICH.IBM.COM
IBM Research, Zürich

Jean-Philippe Pellet JEP@ZURICH.IBM.COM
IBM Research and ETH, Zürich

Peter Spirtes PS7Z@ANDREW.CMU.EDU
Carnegie Mellon University, Pennsylvania

Alexander Statnikov ALEXANDER.STATNIKOV@MED.NYU.EDU
New York University

Introduction

We prepared four datasets for the first challenge on causality we organized for the World Congress on Computational Intelligence, WCCI 2008. The focus of this challenge, entitled "**Causation and Prediction**", was on the **evaluation of causal modeling techniques**, aiming at predicting **the effect of "interventions"** performed by an external agent. Examples of that problem are found in the medical domain to predict the effect of a drug prior to administering it, or in econometrics to predict the effect of a new policy prior to issuing it. We concentrated on a given target variable to be predicted (*e.g.,* health status of a patient) from a number of candidate predictive variables or "features" (*e.g.,* risk factors in the medical domain). We limited ourselves to **binary target variables** (two-class classification problems), but **the input variables are either binary or continuous**. For each task, a **training set drawn from a "natural" distribution** is given and three test sets: **one test set from the same distribution as the training set** and **two test sets obtained after an external agent manipulated certain variables** (*i.e.,* set them to arbitrary values, not drawn from the natural distribution). The target variable itself is never manipulated and it is assumed that the external agent interventions do not alter the mechanisms by which one variable is determined by the value of others. The participants were asked to provide predictions of the target variable on test data and the list of variables (features) used to make predictions. The challenge platform remains open for post-challenge submissions (see `http://clopinet.com/causality`). The datasets were also used for the task **LOCANET**, which was part of the second causality challenge we organized for the Neural Information Processing Systems conference (NIPS 2008). The goal of LOCANET was to uncover the LOcal CAusal NETwork around the target. **This report was not available to the participants of the challenges.**

Table 1: **Datasets of the causation and prediction challenge.** The datasets of the challenge are indicated in boldface. Two other toy datasets (LUCAS and LUCAP) were used for illustration purpose.

Dataset	Description	Var. type	Var. #	Train	Test
LUCAS	Toy example (Bayes net)	binary	11	10000	20000
LUCAP	Toy example (Bayes net + probes)	binary	143	10000	20000
REGED	Genomics re-simulated	numeric	999	500	20000
SIDO	Pharmacology (real data + probes)	binary	4932	12678	10000
CINA	Marketing (real data + probes)	mixed	132	16033	10000
MARTI	Same as REGED + correlated noise	numeric	1024	500	20000

Brief description of the data

- **REGED** is a dataset generated by a simulator of gene expression data, which was trained on real DNA microarray data. The target variable is lung cancer subtype. Hence, the task is to discover genes, which trigger disease or are a consequence of disease. The manipulations simulate the effect of agents such as drugs and/or RNA silencing. For REGED1, the list of manipulated variable is provided, but not for REGED2. REGED has 999 features, of which the Markov blanket contains 2 direct causes, 13 direct effects and 6 spouses in REGED0, but only 2 direct causes, 6 direct effects and 4 spouses in REGED1, and 2 direct causes in REGED2.

- **SIDO** consists of real data, from a drug discovery problem. The variables represent molecular descriptors of pharmaceutical compounds, whose activity on the HIV virus must be determined (the target variable). Knowing which molecule feature is a cause of activity would be of great help to chemical engineers to design new compounds. To test the efficacy of causal discovery algorithm, artificial "distractor" variables (called "probes") were added, which are "non-causes" of the target. All the probes are manipulated in the test sets SIDO1 and SIDO2. The probes must be filtered out to get a good causal discovery score and good prediction performance on test data. SIDO has 4932 features, of which 1644 real features and 3288 probes.

- **CINA** is also a real dataset. The problem is to predict the revenue level of people from census data (marital status, years of study, gender, etc.). As a causal discovery problem, the task is to find causes, which might influence revenue. Similarly as for SIDO, artificial variables (probes) were added. CINA has 132 features, of which 44 real features and 88 probes; all probes are manipulated in CINA1 and 2.

- **MARTI** is a noisy version of REGED. Correlated noise was added to simulate measurement artifacts and introduce spurious relationships between variables. This dataset illustrates that without proper calibration/normalization of data, causal discovery algorithms may yield wrong causal structures. MARTI has 1024 features and the same causal graph as REGED. However, 25 calibrant variables were added to help taking out the noise.

Overall method

Preparing the data included the following steps:

- Adding artificial variables (probes) in real datasets.
- Preprocessing data to obtain features in the same numerical range (0 to 999 for continuous data and 0/1 for binary data).
- Randomizing the order of the patterns and the features to homogenize the data.
- Splitting the data into training and test set.

The classification results were evaluated with the Area under the ROC Curve (AUC) on test data. The target values on test examples were never revealed. The web site remains available to assess performances of new algorithms. No validation set was used to provide on-line feed-back to the participants during the challenge. Rather, the participants submitted results on test data and, during the development period, obtained a coarse information on the web site about their ranking (in which quartile their submission ranked).

Although the participants were strictly evaluated on prediction performance of the target variable, other metrics were computed to assess the correlation between correct causal structure discovery and correct target value predictions. To assess causal structure discovery, an index measuring the similarity of the feature set to the Markov boundary of the post-manipulation distribution was calculated (see the website of the challenge for details). In a follow up challenge (NIPS 2008), we asked the participants to return the depth 3 local structure, and we assessed its correctness with an edit distance to the true graph (see `http://www.causality.inf.ethz.ch/data/LOCANET.html`).

Real and artificial data

We use two types of data:

- **Re-simulated data:** We train a causal model with real data. The model is then used to generate artificial training and test data for the challenge. Truth values of causal relationships are known for the data generating model and used for scoring causal discovery results. REGED is an example of re-simulated dataset.
- **Real data with probe variables:** We use a dataset of real samples. Some of the variables may be causally related to the target and some may be predictive but non-causal. The nature of the causal relationships of the variables to the target is unknown (although domain knowledge may allow us to validate the discoveries to some extent). We have added to the set of real variables a number of distractor variables called "probes", which are generated by an artificial stochastic process, including explicit functions of some of the real variables, other artificial variables, and/or the target. All probes are non-causes of the target, some are completely unrelated to the target. The identity of the probes in concealed. The fact that truth values of causal relationships are known only for the probes affects the evaluation of causal discovery, which is less reliable than for artificial data.

We give in appendix details about the method we used to generate random probes.

Evaluation

For the first causality challenge "Causation and Prediction" organized for WCCI 2008, the participants were asked to return *prediction scores* or discriminant values v for the target variable on test examples, and a list of features used for computing the prediction scores, sorted in order of decreasing predictive power, or unsorted. The classification decision is made by setting a threshold μ on the discriminant value v: predict the positive class if $v > \mu$ and the negative class otherwise. The participants could optionally provide results for nested subsets of features, varying the subset size by powers of 2 (1, 2, 4, 8, etc.). Two scores were used:

- **Tscore:** The participants were ranked according to the area under the ROC curve (AUC) computed for test examples (referred to as Tscore), which is the area under the curve plotting sensitivity *vs.* ($1_{\,i}$ specificity) when the threshold μ is varied (or equivalently the area under the curve sensitivity *vs.* specificity). We call "sensitivity" the error rate of the positive class and "specificity" the error rate of the negative class. The AUC is a standard metric in classification. If results were provided for nested subsets of features, the best Tscore was retained. There are several ways of estimating error bars for the AUC. We use a simple heuristic, which gives us approximate error bars, and is fast and easy to implement: we find on the AUC curve the point corresponding to the largest balanced accuracy BAC = 0.5 (sensitivity + specificity). We then estimate the standard deviation of the BAC as: $\sigma = (1/2)\ \mathrm{sqrt}(p_+(1-p_+)/m_+ + p_-(1-p_-)/m_-)$, where m_+ is the number of examples of the positive class, m_- is the number of examples of the negative class, and p_+ and p_- are the probabilities of error on examples of the positive and negative class, approximated by their empirical estimates, the sensitivity and the specificity.

- **Fscore:** We also computed other statistics, which were not used to rank participants, but used in the analysis of the results. Those included the number of features used by the participants called "Fnum", and a statistic assessing the quality of causal discovery in the feature set selected called "Fscore". As with the Tscore, we provided quartile feed-back on Fnum and Fscore during the competition. For the Fscore, we used the AUC for the problem of separating features belonging to the Markov blanket of the test set distribution *vs.* other features. Details are provided on the web site of the challenge. As it turns out, this statistic correlates poorly with the Tscore. After experimenting with various scores, we found better alternatives.

- **New Fscore.** We ended up using as the new Fscore the Fmeasure for REGED and MARTI and the precision for SIDO and CINA, after experimenting with various alternative measures inspired by information retrieval. We use the following definitions: precision = $tp = (tp + fp)$, recall = $tp = (tp + fn)$ (also called sensitivity), and Fmeasure = 2 precision recall / (precision + recall). Our explorations indicate that precision, recall, and Fmeasure correlate well with Tscore for artificially generated datasets (REGED and MARTI). The *Fmeasure*, which captures the tradeoff between precision and recall, is a good measure of feature set quality for these datasets. However, recall correlates poorly with Tscore for SIDO and CINA, which are datasets of real variables with added artificial "probes". In these cases, we approximate the recall by the fraction of real variables recalled (present in the selected feature set), which can be very different from the true recall that is the fraction of relevant variables. For instance, if many real variables are irrelevant, a good causal discovery algorithm might eliminate them, thus obtaining a poor estimated recall. Hence, we can only use *precision* as of feature set quality for those datasets.

For the LOCANET task of the second causality challenge (NIPS 2008 Pot-luck Challenge), we assessed performance by comparing the local causal network (of depth 3) to the actual local causal network, using an edit distance. A confusion matrix C_{ij} was computed, recording the number of relatives confused for another type of relative, among the 14 types of relatives in depth 3 networks. A cost matrix A_{ij}, was then applied to account for the distance between relatives (computed with an edit distance as the number of substitutions, insertion, or deletion to go from one string to the other, using the string description described above). The score of the solution was computed as:

$$S = \sum_{ij} A_{ij} C_{ij}$$

Cost matrix (A_{ij})

Depth	Desired		1	1	2	2	2	2	3	3	3	3	3	3	3	3	X
Obtained	Relationship		P	C	Sp	GC	Si	GP	GGP	uud	N	PS	SC	IL	CP	GGC	Other
			u	d	du	dd	ud	uu	uuu	uud	udd	udu	ddu	duu	dud	ddd	
1	Parents	u	0	1	1	2	1	1	2	2	2	2	2	2	2	3	4
1	Children	d	1	0	1	1	1	2	3	2	2	2	2	2	2	2	4
2	Spouses	du	1	1	0	1	2	1	2	2	2	1	1	1	1	2	4
2	Gchildren	dd	2	1	1	0	1	2	3	2	1	2	1	2	1	1	4
2	Siblings	ud	1	1	2	1	0	1	2	1	1	1	2	2	1	2	4
2	Gparents	uu	1	2	1	2	1	0	1	1	2	1	2	1	2	3	4
3	Ggparents	uuu	2	3	2	3	2	1	0	1	2	1	2	1	2	3	4
3	Uncles/Aunts	uud	2	2	2	2	1	1	1	0	1	2	3	2	1	2	4
3	Nieces/Nephews	udd	2	2	2	1	1	2	2	1	0	1	2	3	2	1	4
3	Parents of siblings	udu	2	2	1	2	1	1	1	2	1	0	1	2	2	2	4
3	Spouses of children	ddu	2	2	1	1	2	2	2	3	2	1	0	1	2	1	4
3	Parents in law	duu	2	2	1	2	2	1	1	2	3	2	1	0	1	2	4
3	Children of spouses	dud	2	2	1	1	1	2	2	1	2	2	2	1	0	1	4
3	Ggchildren	ddd	3	2	2	1	2	3	3	2	1	2	1	2	1	0	4
X	Other		4	4	4	4	4	4	4	4	4	4	4	4	4	4	0

For artificially generated data (REGED and MARTI), the ground truth for the target local neighborhood was determined by the generative model. For real data with artificial "probe" variables (SIDO and CINA), we do not have ground truth for the relationships of the real variables to the target. The score was computed on the basis of the artificial variables only.

Data formats

All the data sets are in the same format and include 4 files in text format:
dataname.param: Parameters and statistics about the data
dataname_train.data: Training set (a sparse or a regular matrix, patterns in lines, features in columns).
dataname_test.data: Test set.
dataname_train.targets: Labels (truth values of the classes) for training examples.

The matrix data formats used are a space delimited file with a new-line character at the end of each line.

The results on each dataset should be formatted in 3 ASCII files:
dataname_train.predict: a discriminant value for training set output (a discriminant value is a score, which is large for examples of the positive class and small for examples of the negative class).
dataname_test.predict: a discriminant value for test set output.
dataname_feat.slist: a sorted list of features used.

or
dataname_feat.ulist: an unsorted list of features used.

Single predictions for each training or test examples could be provided, or multiple predictions corresponding to nested tested subsets of features could be given in the form of a data table (see the website of the challenge for details).

Dataset A: REGED

1) Topic

REGED stands for **RE**simulated **G**ene **E**xpression **D**ataset. The goal of **REGED** is to find genes, which could be responsible of lung cancer subtype. The data are "re-simulated", *i.e.,* generated by a model derived from real human lung-cancer microarray gene expression data. From the causal discovery point of view, it is important to separate genes whose activity cause lung cancer from those whose activity is a consequence of the disease.

We propose three tasks, REGED0, REGED1, and REGED2. All three datasets includes 999 features, the same 500 training examples, and different test sets of 20000 examples. The target variable is binary; it separates adenocarcinoma samples from squamous samples. The three tasks differ in the test data distribution, which results from various types of manipulations:

REGED0: No manipulation (distribution identical to the training data).

REGED1: The following variables are manipulated:
20, 27, 36, 70, 82, 83, 85, 91, 118, 125, 139, 143, 160, 169, 176, 185, 191, 204, 219, 224, 229, 239, 243, 251, 252, 269, 281, 282, 295, 297, 301, 319, 320, 321, 342, 350, 357, 359, 361, 378, 387, 407, 409, 412, 429, 430, 469, 472, 499, 501, 507, 512, 540, 545, 552, 561, 566, 572, 580, 586, 593, 618, 622, 637, 651, 663, 674, 681, 683, 686, 690, 702, 727, 754, 762, 764, 773, 786, 805, 815, 835, 861, 872, 873, 877, 880, 889, 904, 935, 936, 939, 942, 949, 962, 977, 985, 989, 991, 992, 994.

REGED2: Many variables are manipulated, including all the consequences of the target. When a manipulation is performed, the values of the manipulated variables are clamped to given values by an "external agent". All other variable values are obtained after the system stabilizes when it is let to evolve according to its own dynamics.

2) Sources

a. Original owners

Alexander Statnikov and Constantin F. Aliferis

References:

Aliferis, C.F. and Statnikov, A. (2007) High-Fidelity Resimulations from High-Throughput Data. *Technical Report DSL 07–03, Discovery Systems Laboratory, Vanderbilt University.*

b. Donor of database

This version of the database was prepared for the WCCI2008 by the Causality Workbench team.

c. Date prepared: Summer 2007.

d. Date released for the challenge: December 2007.

3) Past usage

Used for the two first challenges organized by the Causality Workbench Team: (1) the Causation and Prediction Challenge (WCCI 2008), (2) the NIPS 2008 Pot-Luck challenge, as part of the LOCANET task (see `http://clopinet.com/causality`).

4) Experimental design

1. Creation of a resimulated gene expression dataset that is modeled closely after the real microarray gene expression data

The ability to produce realistic simulated data is a critical component of evaluating discovery algorithms in a systematic manner. In machine learning, a standard practice is to use expert-derived networks (a prototypical example being the ALARM network [1]). Such networks do not correspond to biological systems, they are too small, and the distributions are highly discrete. In bioinformatics, researchers have simulated data from ad-hoc unvalidated generating models, or from validated but very small models (*e.g.*, `http://bioinformatics.upmc.edu/GE2/`, `http://www.phil.cmu.edu/projects/tetrad/`, [2]). In order to obtain large, realistic networks and data capturing the characteristics of human gene expression data, we applied a High-Fidelity Data Resimulation technique [3] that generates synthetic data from a causal process that is induced from the real data and guarantees that the synthetic data is non-distinguishable from the real data.

The High-Fidelity Data Resimulation method was applied to 1,000 randomly selected variables (999 oligonucleotide probes and one phenotype variable) from the 12,600 probes in the Affymetrix U95A array lung cancer gene expression data of [4]. Once a network (*REGED0*) was obtained by HITON-Bach algorithm, a set of 30,000 samples was generated from this network. The area under the ROC curve (AUC) for discriminating the real from the synthetic data (i.e., joint real and synthetic distributions of the 1,000 variables) indicated minor discrepancies between the two distributions.

2. Creation of additional re-simulated gene expression datasets with manipulations

The datasets with manipulations are generated from the original network (*REGED0*) by disconnecting some variables from their direct causes (see Figure 1). After the manipulations were performed in the network graph, the network was re-parameterized accordingly and data (30,000 samples) was generated from it as described in the previous section. We produced two datasets with manipulations:

- *REGED1*: We manipulated 1 direct cause of the target, 5 its mostly predictive direct effects, and 94 randomly selected variables that are scattered throughout the network but do not belong to the local neighborhood of target. In total, 100 variables were manipulated.
- *REGED2*: We manipulated 1 direct cause of the target, all (13) its direct effects, and 86 randomly selected variables that are scattered throughout the network but do not belong to the local neighborhood of target. In total, 100 variables were manipulated.

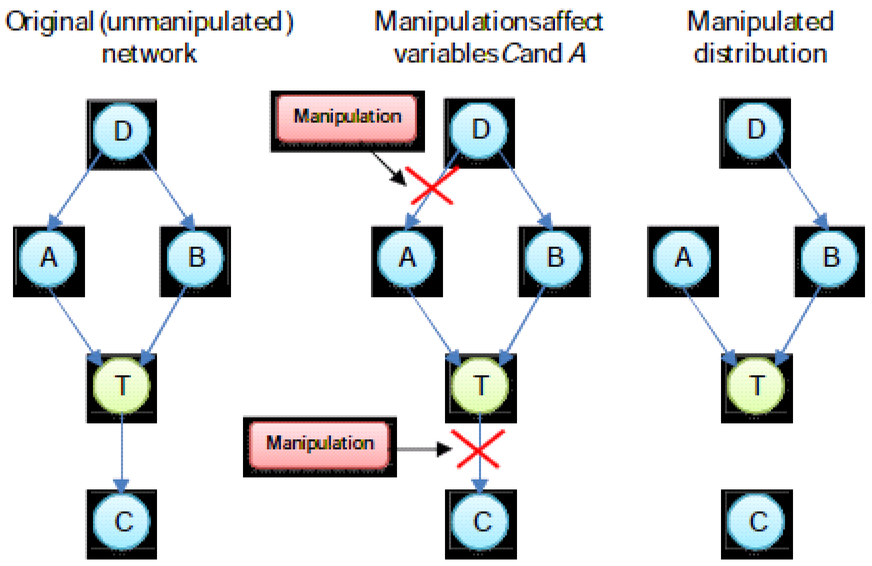

Original (unmanipulated) network
Manipulations affect variables C and A
Manipulated distribution
Manipulation
Manipulation
D
A
B
T
C

Figure 1: Example of manipulations. T is a target (response) variable. In the original network, both A (direct cause of T) and C (direct effect of T) are predictive of T. However, once C and A are manipulated, only A remains predictive of T in the manipulated network.

For *REGED1*, the Challenge provides a list of variables that were manipulated but does not disclose what these variables are (*e.g.,* direct causes, direct consequences, or neither). Since participants have access only to unmanipulated training data (*e.g., REGED0*), the optimal prediction for the target in *REGED1* can be achieved by using a Markov blanket of the target inferred in *REGED0* and excluding all manipulated direct effects and their direct causes.

For *REGED2*, the Challenge mentions that many variables including all direct effects of the target were manipulated but does not provide indices of these variables in the dataset (unlike *REGED1*). Since participants have access only to unmanipulated training data, the optimal solution for *REGED2* is to use only direct causes of the target variable.

3. Structure details:

- Local neighborhood of T contains 2 direct causes and 13 direct effects; there are also 6 spouses of T.
- Out of 999 non-target variables in the network, 789 are connected to T by an undirected path and 210 are not.
- Local structure in the natural distribution (all numbers below refer to column indices in the 2D-array data.)
  - Direct causes: {322, 931}
  - Direct effects: {84, 252, 345, 410, 426, 454, 572, 594, 595, 740, 818, 826, 940}
  - Local neighborhood: {84, 252, 322, 345, 410, 426, 454, 572, 594, 595, 740, 818, 826, 931, 940}

Note: the variable numbers are offset by 1 compared to the column number c in the data tables. The target is numbered 1. All other variables are numbered $c+1$.

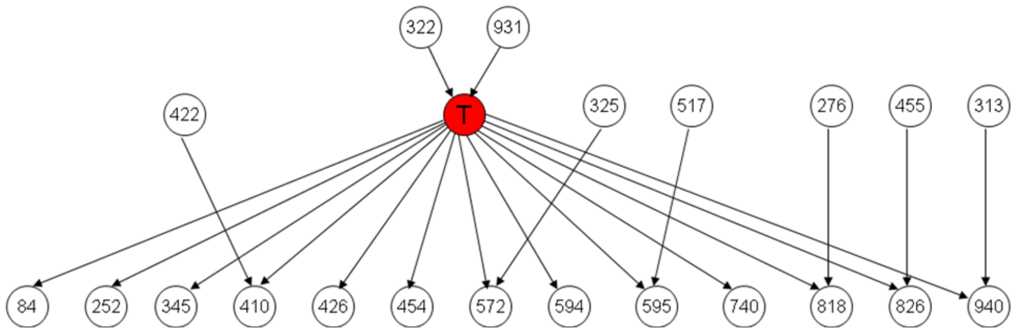


Figure 2: Local neighborhood of T in *REGED* network.

Reference List

1. Beinlich I, Suermondt HJ, Chavez R, Cooper G: **The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks.** *Proceedings of the Second European Conference on Artificial Intelligence in Medicine* 1989.

2. Basso K, Margolin AA, Stolovitzky G, Klein U, la-Favera R, Califano A: **Reverse engineering of regulatory networks in human B cells**. *Nat Genet* 2005, 37:382–390.

3. Aliferis CF, Statnikov A: **High-Fidelity Resimulation from High-Throughput Data.** *Technical Report DSL 07–03* 2007.

4. Bhattacharjee A, Richards WG, Staunton J, Li C, Monti S, Vasa P, Ladd C, Beheshti J, Bueno R, Gillette M, Loda M, Weber G, Mark EJ, Lander ES, Wong W, Johnson BE, Golub TR, Sugarbaker DJ, Meyerson M: **Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses.** *Proc Natl Acad Sci U S A* 2001, 98:13790–13795.

5) Number of examples and class distribution

REGED0	**Positive ex.**	**Negative ex.**	**Total**	**Check sum**
Training set	59	441	500	167114863.00
Test set	1853	18147	20000	6678340769.00
All	1912	18588	20500	6845455632.00

REGED1	**Positive ex.**	**Negative ex.**	**Total**	**Check sum**
Training set	59	441	500	167114863.00
Test set	1833	18167	20000	6657027579.00
All	1892	18608	20500	6824142442.00

REGED2	**Positive ex.**	**Negative ex.**	**Total**	**Check sum**
Training set	59	441	500	167114863.00
Test set	1781	18219	20000	6658529997.00
All	1840	18660	20500	6825644860.00

Note: the training set is the same for all three datasets.

6) Type of input variables and variable statistics

Artificial variables	**Random probes**	**Total**
999	0	999

All variables are **integer** quantized on 1000 levels. There are no **missing values**.

7) Results of baseline methods

Several quality assurance tests were performed with datasets in the Challenge. The following two goals were pursued in these tests:

1. Ensure sufficient strength of predictive signal.
2. Assess effectiveness and impact of manipulations on predictivity of the target in manipulated datasets.

We analyzed classification performance of different feature subsets. Namely, we analyzed difference in predictivity in manipulated data when using Markov blanket from manipulated network and Markov blanket from unmanipulated network. Also, we compared to using no variable selection combined with a powerful regularized classifier (*e.g.,* SVMs). In addition, we executed several state-of-the-art causal discovery algorithms to obtain baseline results for the Challenge datasets:

a. HITON-PC (semi-interleaved, without symmetry correction)

b. HITON-PC (semi-interleaved, without symmetry correction) with FDR prefiltering of variables

The reference submission uploaded to the website of the challenge. (reference_hpc) is for HITON-PC (max−k=3, alpha=0.05) & Linear SVMs (C=0.001) using original data for both algorithms. The test AUCs are:
REGED0 0.9998
REGED1 0.9800
REGED2 0.8447
The best AUC predictions of challenge participants are:
REGED0 1.000
REGED1 0.989
REGED2 0.839
The best score of participants on the LOCANET task is: 0.22. For reference, we also trained a linear SVM classifier using various feature subsets derived from the truth values of the causal relationships (Table A.1)

Table A.1: Results obtained with subsets of features derived from the true causal relationships. (a) Subsets chosen. (b) AUC obtained with a linear SVM.

Feature subsets

		1	2	3	4	5
REGED 0	1	Parents (N=2)	Children (N=13)	Parents + Children (N=15)	Markov Blanket (N=21)	All (N=999)
REGED 1	2	Parents (N=2)	Children (N=13)	Parents + Children (N=15)	Markov Blanket (N=21)	All (N=999)
REGED 2	3	Parents (N=2)	Children (N=13)	Parents + Children (N=15)	Markov Blanket (N=21)	All (N=999)
REGED 1	4	Parents (N=2)	unmanipulated Children (N=8)	Parents + unmanipulated Children (N=10)	Markov Blanket – manipulated Children and their Parents (N=14)	All (N=999)
REGED 2	5	Parents (N=2)	unmanipulated Children (N=0)	Parents + unmanipulated Children (N=2)	Markov Blanket – manipulated Children and their Parents (N=2)	All (N=999)

(a)

AUC results (subsets of features of the previous table)

		1	2	3	4	5
REGED 0	1	0.9411	0.9994	0.9998	0.9998	0.9946
REGED 1	2	0.9265	0.9154	0.9817	0.9841	0.9344
REGED 2	3	0.9343	0.4952	0.8426	0.8177	0.7254
REGED 1	4	0.9265	0.9788	0.9941	0.9956	0.9344
REGED 2	5	0.9343	0.5	0.9343	0.9343	0.7254

(b)

Dataset B: SIDO

1) Topic

SIDO stands for **SI**mple **D**rug **O**peration mechanisms. The SIDO dataset contains descriptors of molecules, which have been tested against the AIDS HIV virus. The target values indicate the molecular activity (+1 active, −1 inactive). The causal discovery task is to uncover causes of molecular activity among the molecule descriptors. This would help chemists in the design of new compounds, retaining activity, but having perhaps other desirable properties (less toxic, easier to administer). The molecular descriptors were generated programmatically from the three dimensional description of the molecule, with several programs used by pharmaceutical companies for QSAR studies (Quantitative Structure-Activity Relationship). For example, a descriptor may be the number of carbon molecules, the presence of an aliphatic cycle, the length of the longest saturated chain, etc. The dataset includes 4932 variables (other than the target), which are either molecule descriptors (all potential causes of the target) or "probes" (artificially generated variables, which are not causes of the target). The training set and the unmanipulated test set are similarly distributed. They are constructed such that some of the "probes" are effects (consequences) of the target and/or of other real variables, and some are unrelated to the target or other real variables. Hence, both in the training set and the unmanipulated test set, all the probes are non-causes of the target, yet some of them may be predictive of the target. In the manipulated test set, all the "probes" are "manipulated" in every sample by an "external agent" (*i.e.,* set to given values, not affected by the dynamics of the system) and can therefore not be relied upon to predict the target. The identity of the probes was concealed during the challenge. The probes were used to assess the effectiveness of the algorithms to dismiss non-causes of the target for making predictions in manipulated test data.

2) Sources

a. Original owners

The data was made available by the National Cancer Institute (NCI), via the DTP AIDS Antiviral Screen program at: http://dtp.nci.nih.gov/docs/aids/aids_data.html. The DTP AIDS Antiviral Screen has checked tens of thousands of compounds for evidence of anti-HIV activity. Available are screening results and chemical structural data on compounds that are not covered by a confidentiality agreement.

b. Donor of database

This version of the database was prepared for the WCCI2008 by the Causality Workbench team.

c. Date prepared: Fall 2007.

d. Date released for the challenge: December 2007.

3) Past usage

Another version of this dataset was used under the name HIVA for past challenges (the WCCI06 challenge on performance predictions, the NIPS08 model selection game and the IJCNN 07 "agnostic learning *vs.* prior knowledge" challenge. See http://clopinet.com/challenges/ for details). SIDO uses the same original data, but differently formatted and split. SIDO was used for the two first challenges organized by the Causality Workbench Team: (1) the Causation and Prediction Challenge (WCCI 2008), (2) the NIPS 2008 Pot-Luck challenge, as part of the LOCANET task (see http://clopinet.com/causality).

4) Experimental design

We describe first the raw data and the preprocessing we made (similar for HIVA and SIDO). We then describe the process used to add artificial variables (probes).

1. Preprocessing

The screening results of the May 2004 release containing the screening results for 43,850 compounds were used. The results of the screening tests are evaluated and placed in one of three categories:

- **CA** – Confirmed active
- **CM** – Confirmed moderately active
- **CI** – Confirmed inactive

We converted this into a two-class classification problem: Inactive (CI) *vs.* Active (CA or CM.)

Chemical structural data for 42,390 compounds was obtained from the web page. It was converted to structural features by three different methods, yielding three feature sets that were concatenated. We matched the compounds in the structural description files and those in the compound activity file, using the NSC id number. We ended up with 42678 examples.

First feature set (the same as the one used for HIVA):

The program ChemTK version 4.1.1, Sage Informatics LLC was used to generate features. (Appendix C).
Reference:
Miller, D.W. A Chemical Class-Based Approach to Predictive Model Generation. J. Chem. Inf. Comput. Sci. 2003, 43, 568–578
http://www.ncbi.nlm.nih.gov/pubmed/12653523

The **1617** features of the original HIVA dataset were included in SIDO; they are all binary:

- unbranched_fragments: 750 features
- pharmacophores: 495 features
- branched_fragments: 219 features
- internal_fingerprints: 132 features
- ring_systems: 21 features

Only binary features having a total number of ones larger than 100 (> 400 for unbranched fragments) and at least 2% of ones in the positive class were retained. In all cases, the default program settings were used to generate keys (except for the pharmacophores for which "max number of pharmacophore points" was set to 4 instead of 3; the pharmacophore keys for Hacc, Hdon, ExtRing, ExtArom, ExtAliph were generated, as well as those for Hacc, Hdon, Neg, Pos.) The keys were then converted to attributes.

We briefly describe the attributes/features:

Branched fragments: each fragment is constructed through an "assembly" of shortest-path unbranched fragments, where each of the latter is required to be bounded by two atoms belonging to one or more pre-defined "terminal-atom".

Unbranched fragments: unique non-branching fragments contained in the set of input molecules.

Ring systems: A ring system is defined as any number of single or fused rings connected by an unbroken chain of atoms. The simplest example would be either a single ring (*e.g.,* benzene) or a single fused system (*e.g.,* naphthalene).

Pharmacophores: ChemTK uses a type of pharmacophore that measures distance via bond connectivity rather than a typical three-dimensional distance. For instance, to describe a hydrogen-bond acceptor and hydrogen-bond donor separated by five connecting bonds, the corresponding key string would be "HAcc.HDon.5". The pharmacophores were generated from the following features:

Neg.	Explicit negative charge.
Pos.	Explicit positive charge.
HAcc.	Hydrogen-bond acceptor.
HDon.	Hydrogen-bond donor.
ExtRing.	Ring atom having a neighbor atom external to the ring.
ExtArom.	Aromatic ring atom having a neighbor atom external to the ring.
ExtAliph.	Aliphatic ring atom having a neighbor atom external to the ring.

Internal fingerprints: small, fixed catalog of pre-defined queries roughly similar to the MACCS key set developed by MDL.

Second feature set (new in SIDO)

Hans Bitter (Roche Palo Alto) kindly provided us with features computed with the Chemical Computing Group software (Appendix D). We retained **13** binary features:

FFType_ang	opr_leadlike
Q_VSA_FPOS	FFType_oop
lip_druglike	FFType_bond
FFType_atom	C2
reactive	C1
FFType_all	Q_VSA_FHYD
FFType_tor	

He also provided us with features computed with the program ISIS We retained **11** binary ISIS features:

MACCS(145)	MACCS(--6)
MACCS(125)	MACCS(120)
MACCS(--5)	MACCS(--4)
MACCS(162)	MACCS(-49)
MACCS(-10)	MACCS(136)
MACCS(166)	

MACCS stands for Molecuar ACCess System. The MACCS keys are a set of questions about a chemical structure. Here are some of the questions:

- Are there fewer than 3 oxygens?

- Is there a S-S bond?
- Is there a ring of size 4?
- Is at least one F, Cl, Br, or I present?

Third feature set (new in SIDO)

Georg Wichard (Institute of Molecular Pharmacology, Germany) kindly provided us with several feature sets. Those include:

- "Ghose-Crippen" Descriptors [**Prediction of Hydrophobic (Lipophilic) Properties of Small Organic Molecules Using Fragmental Methods: An Analysis of ALOGP and CLOGP Methods Arup K. Ghose,* Vellarkad N. Viswanadhan, and John J. Wendoloski**, *J. Phys. Chem. A* **1998**, *102*, 3762–3772]. We retained only **one** binary feature ALogP_Count[98] (I attached to C^3_{sp3}).
- FMP features (we retained **2** binary features: ES_Count_ssNH2 and ES_Count_ssPH)

2. Adding artificial variables (probes)

The motivation for adding artificial variables is that the truth values of the causal relationships between the real variables are not known. Compared to purely artificially generated data, using real variables allows us to work on realistic data distributions. The added artificial variables allow us to assess the performances of causal discovery algorithms.

The target variable is a real variable. Consequently, no artificial variable may be a cause of the target (direct or indirect). The artificial variables are constructed as functions (plus noise) of subsets of real variables (which may include the target) and other artificial variables. Some artificial variables are generated randomly (hence have no dependency with the real variables).

The method used for generating random probes is described in Appendix A. The specific parameters used for SIDO are found below:

```
    n=size(X,2); % Number of true variables
    nc=round(n/2); % Number of confounder probes
    ne=round(n); % Number of effect probes
    np=nc; % Number of truly random probes
    tpnc = 3; % Number of parent true variables for confounders
    ppnc = 2; % Number of parent, which are noise, for confounders
    tpne = 2; % Number of parent confounder variables for effects
    ppne = 2; % Number of parent, which are noise, for effects
    nlval=2; % non-linearity level 2
    noise=0.05; % random noise level (fraction of output range)
    top_num=50; % number of top ranking causes kept
    noise_Y=0.1;
    num_manip=0;
    [X, parents]=add_probes(X, Y, np, nc, ne, tpnc, ppnc, tpne, ppne,
nlval, noise, num_manip, top_num, noise_Y);
```

Note: the probes are first created unmanipulated for SIDO0 and then manipulated according to the methods described in Appendix A.

We summarize the statistics of the probes added:

```
== Total number of variables: 4932 ==
## Real variables (1644):
```

```
## Probes:
== Random (822):
  205 spouses of true var:
  205 spouses of target:
  412 independent of target: Warning, some probes assigned to be
spouses are unused
== Confounders (822): Warning, wierd set
  3.49+- 0.98 true variable parents,  2.50+- 0.50 parents unrelated to
target
== Effects (1644): Warning, wierd set
  2.51+- 0.50 confounder parents,  2.50+ 0.50 parents unrelated to
target
```

5) Number of examples and class distribution

SIDO0	Positive ex.	Negative ex.	Total	Check sum
Training set	452	12226	12678	6155080
Test set	351	9649	10000	4863127
All	803	21875	22678	11018207

SIDO1	Positive ex.	Negative ex.	Total	Check sum
Training set	452	12226	12678	6155080
Test set	335	9665	10000	4879177
All	787	21891	22678	11034257

SIDO2	Positive ex.	Negative ex.	Total	Check sum
Training set	452	12226	12678	6155080
Test set	365	9635	10000	4865938
All	817	21861	22678	11021018

Note: the training set is the same for all three datasets.

6) Type of input variables and variable statistics

Real variables	Random probes	Total
1644	**3288**	**4932**

All variables are **binary**. There are **no missing values**.

7) Results of baseline methods

Prior to releasing the data, we performed experiments with various causal discovery algorithms to assess the dataset and adjust the probe generation to a proper level of difficulty. We show below the experiments with the last version of the probe generation algorithm, which we ended up using. The final dataset that was released contains fewer variables because we removed at the last minute a few non-binary variables, which were accidentally left out and slightly reduced the number of probe. The systematic test were not re-run. However, several baseline results were uploaded to the web site of the challenge.

Select features from **natural distribution with probes**
Estimate classification performance in **manipulated distribution with probes**

Variable subset	Classification AUC	# of selected variables	# of selected real features	# of selected probes	# of selected probes that are children of the target
PC1	0.5771	259	75	184	119
PC1, real features only	0.6994	75	75	0	0
PC1, probes only	0.4553	184	0	184	119
PC2	0.6144	367	116	251	157
PC2, real features only	0.6296	116	116	0	0
PC2, probes only	0.4458	251	0	251	157
PC3	0.5989	25	12	13	8
PC3, real features only	0.4264	12	12	0	0
PC3, probes only	0.4166	13	0	13	8
PC4	0.5998	41	16	25	19
PC4, real features only	0.4621	16	16	0	0
PC4, probes only	0.4208	25	0	25	19
PC5	0.5421	10	4	6	3
PC5, real features only	0.5000	4	4	0	0
PC5, probes only	0.4135	6	0	6	3
PC6	0.6068	25	10	15	15
PC6, real features only	0.4074	10	10	0	0
PC6, probes only	0.4453	15	0	15	15
PC-FDR1	0.5867	230	69	161	96
PC-FDR1, real features only	0.6709	69	69	0	0
PC-FDR1, probes only	0.4480	161	0	161	96
PC-FDR2	0.5995	338	108	230	141
PC-FDR2, real features only	0.6107	108	108	0	0
PC-FDR2, probes only	0.4552	230	0	230	141
PC-FDR3	0.6038	28	12	16	11
PC-FDR3, real features only	0.5570	12	12	0	0
PC-FDR3, probes only	0.4238	16	0	16	11
PC-FDR4	0.6027	47	19	28	22
PC-FDR4, real features only	0.6448	19	19	0	0
PC-FDR4, probes only	0.4258	28	0	28	22
PC-FDR5	0.6240	12	9	3	2
PC-FDR5, real features only	0.5742	9	9	0	0
PC-FDR5, probes only	0.5000	3	0	3	2
PC-FDR6	0.6079	20	12	8	6
PC-FDR6, real features only	0.5407	12	12	0	0
PC-FDR6, probes only	0.4284	8	0	8	6

PC1: HITON-PC with alpha = 0.01, max−k =1
PC2: HITON-PC with alpha = 0.05, max−k =1
PC3: HITON-PC with alpha = 0.01, max−k =2
PC4: HITON-PC with alpha = 0.05, max−k =2
PC5: HITON-PC with alpha = 0.01, max−k =3
PC6: HITON-PC with alpha = 0.05, max−k =3
Same numbering scheme applies to PC-FDR methods. Benjamnin FDR prefiltering was applied prior to running HITON-PC. We used G2 test for all versions of HITON-PC.

The reference submission uploaded to the website of the challenge. (reference_hpc) is for HITON-PC (max−k=1, alpha=0.01) & Linear SVM (C=1), using original (binary) data for both algorithms. HITON-PC uses first 5000 samples in the training data:

SIDO0	0.9377
SIDO1	0.6825
SIDO2	0.6174

The best AUC results of the challenge participants are:

SIDO0	0.944
SIDO1	0.753
SIDO2	0.668

The best score on the LOCANET task is: 3.31

Dataset C: CINA

1) Topic

CINA (Census Is Not Adult) is derived from census data (the UCI machine-learning repository Adult database). The data consists of census records for a number of individuals. The causal discovery task is to uncover the socio-economic factors affecting high income (the target value indicates whether the income exceeds 50K). The 14 original attributes (features) including age, workclass, education, education, marital status, occupation, native country, etc. have been coded to eliminate categorical variables. Distractor features (artificially generated variables, which are not causes of the target) were added. In training data, some of these distractors are effects (consequences) of the target and/or of other real variables. Some are unrelated to the target or other real variables. Hence, some of the distractors may be correlated to the target in training data, although they do not cause it. The unmanipulated test data are distributed like the training data. Hence both causes and consequences of the target my be predictive in the unmanipulated test data. In contrast, in the manipulated test data, all the distractors are “manipulated” by an “external agent” (*i.e.,* set to given value, not affected by the dynamics of the system) and are therefore they cannot be relied upon to predict the target.

2) Sources

a. Original owners

This data was extracted from the census bureau database found at
http://www.census.gov/ftp/pub/DES/www/welcome.html.

Donor: Ronny Kohavi and Barry Becker,
Data Mining and Visualization
Silicon Graphics.
e-mail: ronnyk@sgi.com for questions.

The information below is excerpted from the UCI machine learning repository:

```
  Extraction was done by Barry Becker from the 1994 Census database. The
prediction task is to determine whether a person makes over 50K a year.
The attributes are:
age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov,
Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school,
Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th,
Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated,
Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales,
Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct,
Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv,
Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative,
Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada,
Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China,
Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam,
Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador,
Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland,
Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong,
Holand-Netherlands.
income: >50K, <=50K.

Split into train-test using MLC++ GenCVFiles (2/3, 1/3 random).
 48842 instances, mix of continuous and discrete    (train=32561,
test=16281)
 45222 if instances with unknown values are removed (train=30162,
test=15060)
 Duplicate or conflicting instances : 6
 Class probabilities for adult.all file
 Probability for the label '>50K'  : 23.93% / 24.78% (without unknowns)
 Probability for the label '<=50K' : 76.07% / 75.22% (without unknowns)

 Description of fnlwgt (final weight)
 The weights on the CPS files are controlled to independent estimates of
 the civilian noninstitutional population of the US.  These are prepared
 monthly for us by Population Division here at the Census Bureau.  We use
 3 sets of controls. People with similar demographic characteristics
 should have similar weights.
```

b. Donor of database

A first version of the database was prepared for the WCCI 2006 performance prediction challenge by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (isabelle@clopinet.com) under the name ADA. CINA resembles ADA, except that only binary variables were retained and the data were reshuffled.

The present version of CINA was prepared for the causation and prediction challenge by the Causality Workbench Team.

c. Date released for the challenge: January 2008.

3) Past usage

```
First cited in:
@inproceedings{kohavi-nbtree,
    author={Ron Kohavi},
    title={Scaling Up the Accuracy of Naive-Bayes Classifiers: a
           Decision-Tree Hybrid},
    booktitle={Proceedings of the Second International Conference on
               Knowledge Discovery and Data Mining},
    year = 1996}
 Error Accuracy reported as follows, after removal of unknowns from
    train/test sets):
    C4.5       : 84.46+-0.30
    Naive-Bayes: 83.88+-0.30
    NBTree     : 85.90+-0.28
 The following algorithms were later run with the following error rates,
    all after removal of unknowns and using the original train/test split.
    All these numbers are straight runs using MLC++ with default values.

    Algorithm               Error
 -- ----------------        -----
 1  C4.5                    15.54
 2  C4.5-auto               14.46
 3  C4.5 rules              14.94
 4  Voted ID3 (0.6)         15.64
 5  Voted ID3 (0.8)         16.47
 6  T2                      16.84
 7  1R                      19.54
 8  NBTree                  14.10
 9  CN2                     16.00
 10 HOODG                   14.82
 11 FSS Naive Bayes         14.05
 12 IDTM (Decision table)   14.46
 13 Naive-Bayes             16.12
 14 Nearest-neighbor (1)    21.42
 15 Nearest-neighbor (3)    20.35
 16 OC1                     15.04
 17 Pebls                   Crashed.  Unknown why (bounds WERE increased)
```

Note: The performances reported are error rates, not BER. We tried to reproduce these performances and obtained 15.62% error with a linear ridge regression classifier. The performances slightly degraded when we tried to group features (15.67% when we replace the country code by a binary US/nonUS value and 16.40% with further reduction to 33 features.)

Used under the name ADA for the WCCI 2006 Performance Prediction Challenge, the NIPS 2006 Model Selection game and the IJCNN 2007 Agnostic Learning *vs.* Prior Knowledge challenge. See http://clopinet.com/challenges.

Used for the two first challenges organized by the Causality Workbench Team: (1) the Causation and Prediction Challenge (WCCI 2008), (2) the NIPS 2008 Pot-Luck challenge, as part of the LOCANET task (see http://clopinet.com/causality).

4) Experimental design

The following steps were performed to create the original ADA dataset:

- Convert the features to 14 numeric values $a \in 1 \ldots n$.

- Convert the numeric values to binary codes (a vector of n zeros with value one at the a^{th} position. This results in 88 features. The missing values get an all zero vector.
- Downsize the number of features to 48 by replacing the country code by a binary US/nonUS feature.
- Randomize the feature and pattern order.
- Remove the entries with missing values for workclass.

Table C.1: Features of the ADA datasets.

Feature name	min	max	numval	comments
Age	0.19	1	continuous	No missing value.
workclass_Private	0	1	2	2799 missing values (corresponding entries removed.)
workclass_Self_emp_not_inc	0	1	2	
workclass_Self_emp_inc	0	1	2	
workclass_Federal_gov	0	1	2	
workclass_Local_gov	0	1	2	
workclass_State_gov	0	1	2	
workclass_Without_pay	0	1	2	
workclass_Never_worked	0	1	2	
Fnlwgt	0.008	1	continuous	No missing value.
EducationNum	0.06	1	16	Number corresponding to 16 discrete levels of education
maritalStatus_Married_civ_spouse	0	1	2	No missing value.
maritalStatus_Divorced	0	1	2	
maritalStatus_Never_married	0	1	2	
maritalStatus_Separated	0	1	2	
maritalStatus_Widowed	0	1	2	
maritalStatus_Married_spouse_absent	0	1	2	
maritalStatus_Married_AF_spouse	0	1	2	
occupation_Tech_support	0	1	2	2809 missing values (corresponding entries removed.)
occupation_Craft_repair	0	1	2	
occupation_Other_service	0	1	2	
occupation_Sales	0	1	2	
occupation_Exec_managerial	0	1	2	
occupation_Prof_specialty	0	1	2	
occupation_Handlers_cleaners	0	1	2	
occupation_Machine_op_inspct	0	1	2	
occupation_Adm_clerical	0	1	2	
occupation_Farming_fishing	0	1	2	
occupation_Transport_moving	0	1	2	
occupation_Priv_house_serv	0	1	2	
occupation_Protective_serv	0	1	2	
occupation_Armed_Forces	0	1	2	

Table C.1: Features of the ADA datasets (cont).

Feature name	min	max	numval	comments
relationship_Wife	0	1	2	No missing value.
relationship_Own_child	0	1	2	
relationship_Husband	0	1	2	
relationship_Not_in_family	0	1	2	
relationship_Other_relative	0	1	2	
relationship_Unmarried	0	1	2	
race_White	0	1	2	No missing value.
race_Asian_Pac_Islander	0	1	2	
race_Amer_Indian_Eskimo	0	1	2	
race_Other	0	1	2	
race_Black	0	1	2	
Sex	0	1	2	0=female, 1=male. No missing value.
CapitalGain	0	1	continuous	No missing value.
CapitalLoss	0	1	continuous	No missing value.
HoursPerWeek	0.01	1	continuous	No missing value.
NativeCountry	0	1	2	0=US, 1=non-US. 857 missing values replaced by 1.

Four features were removed because they were found constant in training data. We generated the probes according to the method described in Appendix A. Specifically, we used for CINA the following parameters:

```
    n=size(X,2); % Number of true variables
    nc=round(n/2); % Number of confounder probes
    ne=round(n); % Number of effect probes
    np=nc; % Number of truly random probes
    tpnc = 3; % Number of parent true variables for confounders
    ppnc = 2; % Number of parent, which are noise, for confounders
    tpne = 2; % Number of parent confounder variables for effects
    ppne = 2; % Number of parent, which are noise, for effects
    nlval=2; % non-linearity level 2
    noise=0.05; % random noise level (fraction of output range)
    top_num=round(n/2); % number of top ranking causes kept
    noise_Y=0.1;
    X=full(X);
    num_manip=0;
    [X, parents]=add_probes(X, Y, np, nc, ne, tpnc, ppnc, tpne, ppne,
nlval, noise, num_manip, top_num, noise_Y);
```

Note: the probes are first created unmanipulated for CINA0 and then manipulated according to the methods described in Appendix A.

The statistics on the generated probes are found below:

```
== Total number of variables: 132 ==
## Real variables (44): 1 ... 44
## Probes:
== Random (22):
  5 spouses of true var:
  5 spouses of target:
  12 independent of target: 121 ... 132
== Confounders (22): 50 ... 71
  3.14+- 0.83 true variable parents,  2.32+- 0.48 parents unrelated to
target
== Effects (44): 77 ... 120
  2.48+- 0.51 confounder parents,  2.55+ 0.50 parents unrelated to
target
```

5) Number of examples and class distribution

CINA0	Positive ex.	Negative ex.	Total	Check sum
Training set	3939	12094	16033	142172387.00
Test set	2425	7575	10000	88827113.00
All	6364	19669	26033	230999500.00

CINA1	Positive ex.	Negative ex.	Total	Check sum
Training set	3939	12094	16033	142172387.00
Test set	2540	7460	10000	88535493.00
All	6479	19554	26033	230707880.00

CINA2	Positive ex.	Negative ex.	Total	Check sum
Training set	3939	12094	16033	142172387.00
Test set	2518	7482	10000	88617057.00
All	6457	19576	26033	230789444.00

Note: the training set is the same for all three datasets.

6) Type of input variables and variable statistics

Real variables	Random probes	Total
44	88	132

The variables are **mixed** (continuous and binary). There are **no missing values**.

7) Results of baseline methods on CINA

1. Cheating ranking of features

First, we created a ranking of features, using the knowledge of the causal relationships. All real variables are tentatively assumed to be parents of the target. The features belonging belonging to the MB of the post-manipulation distribution are ranked first (then sorted by Pearson correlation coefficient), all other features are ranked last (also sorted by Pearson correlation coefficient). We trained classifiers on increasing numbers of features using this ranking. The classifier used is a **linear ridge regression classifier**. DAUC and DBER are AUC and BER (balanced error rate) on training date. TAUC and TBER are performance on test data.

CINA0

Num. Var.	DAUC	DERR	TAUC	TERR
1	0.8891	0.0023	0.8889	0.0029
2	0.9045	0.0028	0.9054	0.0035
4	0.9260	0.0027	0.9260	0.0034
8	0.9300	0.0027	0.9314	0.0034
16	0.9478	0.0027	0.9497	0.0034
32	0.9623	0.0028	0.9632	0.0036
64	0.9661	0.0027	0.9663	0.0034
128	0.9674	0.0027	0.9670	0.0035
132	0.9674	0.0027	0.9670	0.0035

CINA1

Num. Var.	DAUC	DERR	TAUC	TERR
1	0.7556	0.0030	0.7600	0.0037
2	0.7574	0.0030	0.7603	0.0037
4	0.7730	0.0033	0.7726	0.0041
8	0.8691	0.0035	0.8684	0.0044
16	0.8845	0.0034	0.8838	0.0043
32	0.8907	0.0034	0.8900	0.004
64	0.9675	0.0027	0.7937	0.0052
128	0.9674	0.0027	0.7883	0.005
132	0.9674	0.0027	0.7873	0.0049

CINA2

Num. Var.	DAUC	DERR	TAUC	TERR
1	0.7556	0.0030	0.7605	0.0038
2	0.7574	0.0030	0.7605	0.0038
4	0.8558	0.0039	0.8573	0.0049
8	0.8691	0.0035	0.8723	0.0045
16	0.8834	0.0033	0.8848	0.0043
32	0.8909	0.0033	0.8910	0.0042
64	0.9675	0.0027	0.5492	0.0041
128	0.9674	0.0027	0.5483	0.0044
132	0.9674	0.0027	0.5481	0.0043

We then performed experiments with various causal discovery algorithms to select features. A linear SVM with C = 1 is used in these experiments.

Select features from **natural distribution w/o probes**

Estimate classification performance in **natural distribution w/o probes** (CINA0)

Experiment id	Variable subset	Classification AUC	# of selected variables	# of selected real features	# of selected probes	# of selected probes that are children of the target
1	**PC** (HITON-PC)	0.8982	23	23	0	0
2	**PC** (HITON-PC-FDR)	0.8982	23	23	0	0
3	**Parents** (MMHC)	0.8502	6	6	0	0
4	**Children** (MMHC)	0.7514	8	8	0	0
5	**All**	0.8999	44	44	0	0

Select features from **natural distribution with probes**

Estimate classification performance in **natural distribution with probes** (CINA0)

Experiment id	Variable subset	Classification AUC	# of selected variables	# of selected real features	# of selected probes	# of selected probes that are children of the target
6	**PC** (HITON-PC)	0.9721	37	20	17	16
7	**PC** (HITON-PC), real features only	0.8967	20	20	0	0
8	**PC** (HITON-PC), probes only	0.9201	17	0	17	16
9	**PC** (HITON-PC-FDR)	0.9721	37	20	17	16
10	**PC** (HITON-PC-FDR), real features only	0.8967	20	20	0	0
11	**PC** (HITON-PC-FDR), probes only	0.9201	17	0	17	16
12	**Parents** (MMHC)	0.8479	4	4	0	0
13	**Parents** (MMHC), real features only	0.8479	4	4	0	0
14	**Parents** (MMHC), probes only	0.5000	0	0	0	0
15	**Children** (MMHC)	0.9480	15	9	6	5
16	**Children** (MMHC), real features only	0.7743	9	9	0	0
17	**Children** (MMHC), probes only	0.8967	6	0	6	5
18	**All**	0.9728	132	44	88	44
19	**All**, real features only	0.8988	44	44	0	0
20	**All**, probes only	0.9447	88	0	88	44

Select features from **natural distribution with probes**

Estimate classification performance in **manipulated distribution with probes** (CINA1)

Experiment id	Variable subset	Classification AUC	# of selected variables	# of selected real features	# of selected probes	# of selected probes that are children of the target
21	**PC** (HITON-PC)	0.8496	37	20	17	16
22	**PC** (HITON-PC), real features only	0.8982	20	20	0	0
23	**PC** (HITON-PC), probes only	0.5114	17	0	17	16
24	**PC** (HITON-PC-FDR)	0.8496	37	20	17	16
25	**PC** (HITON-PC-FDR), real features only	0.8982	20	20	0	0
26	**PC** (HITON-PC-FDR), probes only	0.5114	17	0	17	16
27	**Parents** (MMHC)	0.8508	4	4	0	0
28	**Parents** (MMHC), real features only	0.8508	4	4	0	0
29	**Parents** (MMHC), probes only	0.5000	0	0	0	0
30	**Children** (MMHC)	0.6558	15	9	6	5
31	**Children** (MMHC), real features only	0.7693	9	9	0	0
32	**Children** (MMHC), probes only	0.5114	6	0	6	5
33	**All**	0.8482	132	44	88	44
34	**All**, real features only	0.8999	44	44	0	0
35	**All**, probes only	0.4987	88	0	88	44

Select features from **natural distribution with probes**

Estimate classification performance in **manipulated distribution with probes** (CINA2)

Experiment id	Variable subset	Classification AUC	# of selected variables	# of selected real features	# of selected probes	# of selected probes that are children of the target
21	**PC** (HITON-PC)	0.6643	37	20	17	16
22	**PC** (HITON-PC), real features only	0.8985	20	20	0	0
23	**PC** (HITON-PC), probes only	0.3445	17	0	17	16
24	**PC** (HITON-PC-FDR)	0.6643	37	20	17	16
25	**PC** (HITON-PC-FDR), real features only	0.8985	20	20	0	0
26	**PC** (HITON-PC-FDR), probes only	0.3445	17	0	17	16
27	**Parents** (MMHC)	0.8559	4	4	0	0
28	**Parents** (MMHC), real features only	0.8559	4	4	0	0
29	**Parents** (MMHC), probes only	0.5000	0	0	0	0
30	**Children** (MMHC)	0.4603	15	9	6	5
31	**Children** (MMHC), real features only	0.7631	9	9	0	0
32	**Children** (MMHC), probes only	0.3146	6	0	6	5
33	**All**	0.6584	132	44	88	44
34	**All**, real features only	0.9005	44	44	0	0
35	**All**, probes only	0.3257	88	0	88	44

The reference submission uploaded to the website of the challenge. (reference_hpc) is for HITON-PC (max−k=2, alpha=0.01) & Linear SVM (C=1) using original data for SVM and discrete data for HITON-PC. The test AUC results are:

CINA0	0.9721
CINA1	0.8496
CINA2	0.6643

The AUC best results of the challenge participants on CINA are:

CINA0	0.976
CINA1	0.869
CINA2	0.816

The best score of the LOCANET task is 1.70.

Dataset D: MARTI

1) Topic

MARTI stands for **M**easurement **ARTI**fact. MARTI was obtained from the same data generative process as REGED, a source of simulated genomic data, but a noise model was added to simulate the imperfections of the measurement device.

2) Sources

a. Original owners

This is a modified version of REGED (Alexander Statnikov and Constantin F. Aliferis, 2007) created by Isabelle Guyon.

b. Donor of database

This version of the database was prepared for the WCCI2008 by the Causality Workbench team.

c. Date prepared: Fall 2007.

d. Date released for the challenge: January 2008.

3) Past usage

Used for the two first challenges organized by the Causality Workbench Team: (1) the Causation and Prediction Challenge (WCCI 2008), (2) the NIPS 2008 Pot-Luck challenge, as part of the LOCANET task (see `http://clopinet.com/causality`).

4) Experimental design

The goal of MARTI, like REGED, is to find genes, which could be responsible of lung cancer. The target variable is binary; it separates malignant samples (adenocarcinoma) from control samples (squamous). The feature values representing measurements of gene expression levels are assumed to have been recorded from a two-dimensional microarray 32×32. The training set was perturbed by a zero-mean correlated noise model (neighboring values in one array are generally similarly affected, but the noise pattern is different in every training example).

The test sets have no added noise. This situation simulates a case where we would be using different instruments at "training time" and "test time", *e.g.,* we would use DNA microarrays to collect training data and PCR for testing. We avoided adding noise to the test set because it would be too difficult to filter it without visualizing the test data or computing statistics on the test data, which we forbid. So the scenario is that the second instrument (used at test time) is more accurate. In practice, the measurements would also probably be more expensive, so part of the goals of training would be to reduce the size of the feature set (we are not making this a requirement in this first challenge).

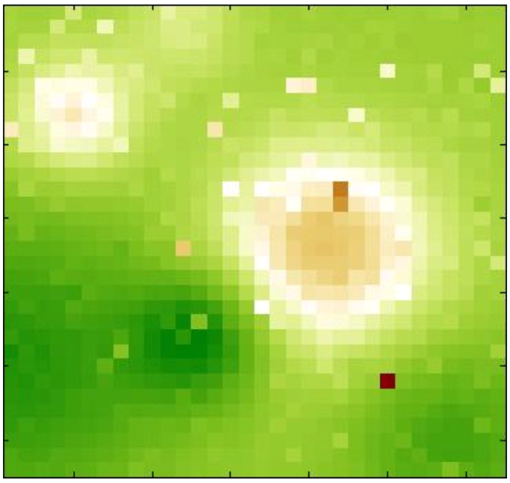


Figure 3: Example of MARTI simulated micro-array

Technical details:

- The features/variables are randomly arranged in a 2d array 32×32. Variables 1:32 form the first column, 33:64 the second, etc.
- To obtain 1024 features, the 999 features of REGED are complemented by 25 "calibrant features", which have value zero plus a small amount of Gaussian noise. The calibrants are spread regularly across the array and have variable indices 34 44 54 64 199 209 219 354 364 374 384 519 529 539 674 684 694 704 839 849 859 994 1004 1014 1024.
- Like for REGED, we proposed 3 tasks MARTI0, MARTI1, and MARTI2, all having the same training set of 500 examples (from the "unmanipulated distribution"), and different test sets of 20000 examples.
- Like for REGED, the three tasks differ in the test data distribution, which results from various types of manipulations:

MARTI0:	No manipulation (distribution identical to the training data).
MARTI1:	The following variables are manipulated: 5, 19, 27, 35, 37, 42, 49, 67, 70, 71, 102, 137, 144, 145, 153, 158, 185, 188, 194, 221, 225, 229, 232, 235, 244, 268, 273, 284, 294, 295, 305, 310, 331, 356, 368, 379, 385, 396, 398, 404, 411, 412, 413, 417, 425, 430, 455, 479, 481, 482, 491, 492, 509, 510, 550, 553, 555, 603, 609, 627, 642, 646, 654, 679, 682, 706, 736, 744, 755, 761, 763, 771, 807, 809, 812, 821, 853, 869, 870, 872, 888,

894, 895, 906, 914, 918, 926, 931, 932, 941, 963, 973, 978, 979, 986, 988, 990, 1001, 1010, 1017.

MARTI2: Many variables are manipulated, including all the consequences of the target.

Filtering the noise and/or taking into account the geometry of the array should be necessary to obtain good results.

In MARTI, what does it mean that "the noise pattern is different in every training example"?

Using our noise model, we drew a noise pattern for every example and added it to that example. When the features are arranged in a 2d 32×32 array (as explained in the documentation), the noise pattern has a smooth structure (neighboring coefficients have similar values). This is kind of background with low frequency. A different noise template is added to each example, but all noise templates are all drawn from the same noise model. If you visualize the training examples after rearranging them as a 32×32 array, you will see this right away. For each feature, the expected value of the noise is zero. But the noise of two neighboring features is correlated. We show in Figure 4 examples of noise patterns (positive values in red and negative values in green).

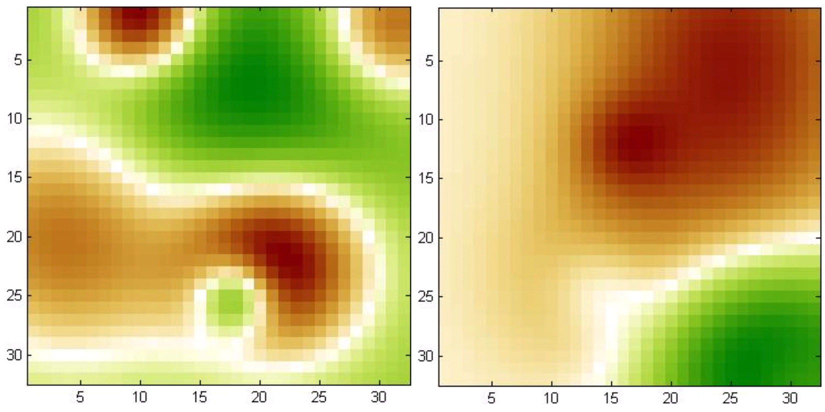


Figure 4: Examples of noise patterns

In MARTI, what does it mean that 25 "calibrant features" have value zero plus a small amount of Gaussian noise? The averages for every calibrant feature is far from zero.

We have 2 kinds of noise. The calibrants are 0±[small Gaussian noise]. Then, on top of that, in training data only, we add the correlated noise model. After we add the correlated noise, because of the small sample size and the large variance, the calibrant values are no longer close to zero (even on average) in training data. However, the median is close zero on average for almost all calibrants, relatively to the signal amplitude:

$$\text{abs}(\text{mean}(\text{median}(X(:,\text{calib})) / \text{std}(\text{abs}(X(:))))) \sim \text{e}-005.$$

In training data, we get: $\text{mean}(\text{abs}(\text{mean}(X))) \sim \text{e}+004$ but and $\text{mean}((\text{mean}(X(:,\text{calib})))) \sim 5\text{e}+003$. In test data, because we did not add noise, the calibrant values are close to zero, relatively speaking: mean(abs(mean(X))) ~ 5e+003 but mean(abs(mean(X(:, calib)))) ~ 1. The calibrants can be used to preprocess the training data by subtracting a bias value after the low frequency noise is removed, so that the calibrant values are zero after preprocessing the training data.

REGED and MARTI do not look like regular microarray data. What kind of normalization did you do?

REGED was obtained by fitting a model to real microrray data. REGED features were shifted and rescaled individually then rounded to integer spanning the range 0:999. MARTI was obtained from data generated by the same model as REGED without rescaling features individually. For MARTI, a particular type of correlated noise was added. The data were then scaled and quantized globally so the features span $-999999 : 999999$.

We chose to make the noise model simple but of high amplitude to make it easy to filter out the noise but hard to ignore it. If you think of the spots on a microarray as an image (MARTI patterns are 32×32 "images"), the noise in MARTI corresponds to patches of more or less intense values, added on top of the original image, representing some kind of slowly varying background. Nowadays, microarray technology has progressed to a point that such heavy backgrounds are not common and occasional contaminated arrays would not pass quality control; furthermore microarray reading software calibrate and normalize data so you would not see data that "bad". But for new instruments under development, such levels of noise are not uncommon.

MARTI illustrates the fact that if you do not take out correlated noise, the result of causal discovery may be severely impaired. Even though the amplitude of the noise is large, the noise is easy to filter out, using the fact that neighboring spots are affected similarly, and using the spots having constant values before noise is added (calibrants). After noise filtering, the residual noise may still impair causal discovery, so it its your challenge to see what can be done to avoid drawing wrong conclusions in the presence of correlated noise.

5) Number of examples and class distribution

MARTI0	Positive ex.	Negative ex.	Total	Check sum
Training set	59	441	500	4824538021.00
Test set	1852	18148	20000	72836533619.00
All	1911	18589	20500	77661071640.00

MARTI1	Positive ex.	Negative ex.	Total	Check sum
Training set	59	441	500	4824538021.00
Test set	1765	18235	20000	72845954638.00
All	1824	18676	20500	77670492659.00

MARTI2	Positive ex.	Negative ex.	Total	Check sum
Training set	59	441	500	4824538021.00
Test set	1662	18338	20000	72914605414.00
All	1721	18779	20500	77739143435.00

Note: the training set is the same for all three datasets.

6) Type of input variables and variable statistics

Artificial variables	Random probes	Total
1024	0	1024

All variables are **integer** quantized on 1000 levels. There are **no missing values**.

7) Results of baseline methods

Below are results for MARTI datasets. All feature sets mentioned in the table below are the true ones (*i.e.,* obtained from the data generating network). The first and three last columns are obtained by training on the raw training set as provided. We also include for comparison the results on REGED and tests of the unmanipulated data (version 0) when training with other versions of the training data:

- MARTI00: training data without noise added (should give results similar to REGED).
- MARTI01: like the MARTI0 training set, but after a crude filtering was performed.

Linear SVM with C = 0.001

	REGED0	MARTI00	MARTI01	MARTI0	MARTI1	MARTI2
Parents	0.941	0.936	0.842	0.883	0.854	0.853
Children	0.999	0.999	0.990	0.481	0.436	0.500
PC	1.000	1.000	0.994	0.749	0.872	0.853
MB	1.000	1.000	0.994	0.894	0.460	0.853
All	0.995	0.995	0.982	0.886	0.779	0.731
All \ MB	0.882	0.875	0.799	0.766	0.727	0.707
Calibrators	N/A	0.496	0.499	0.512	0.505	0.512
All \ Calibrators	N/A	0.99	5 0.982	0.887	0.773	0.724
MB in natural distr.	See results for MB above				0.793	0.746

Linear SVM with C = 1

	REGED0	MARTI00	MARTI01	MARTI0	MARTI1	MARTI2
Parents	0.952	0.947	0.861	0.883	0.855	0.853
Children	0.999	0.999	0.994	0.943	0.702	0.500
PC	1.000	1.000	0.995	0.941	0.866	0.853
MB	1.000	1.000	0.996	0.951	0.756	0.853
All	0.996	0.997	0.985	0.948	0.808	0.730
All \ MB	0.864	0.846	0.782	0.767	0.727	0.692
Calibrators	N/A	0.489	0.499	0.505	0.485	0.497
All \ Calibrators	N/A	0.997	0.985	0.948	0.808	0.730
MB in natural distr.	See results for MB above				0.865	0.614

Polynomial SVM optimized by cross-validation

	REGED0	**MARTI00**	**MARTI01**	**MARTI0**	**MARTI1**	**MARTI2**
Parents	0.948	0.943	0.867	0.883	0.617	0.853
Children	0.999	0.999	0.990	0.842	0.712	0.500
PC	1.000	1.000	0.997	0.981	0.845	0.853
MB	1.000	1.000	0.995	0.975	0.744	0.853
All	0.995	0.997	0.983	0.974	0.843	0.727
All \ MB	0.882	0.875	0.782	0.762	0.728	0.677
Calibrators	N/A	0.493	0.492	0.517	0.505	0.496
All \ Calibrators	N/A	0.997	0.985	0.974	0.843	0.728
MB in natural distr.	See results for MB above				0.865	0.614

We give in Appendix E the Matlab code to obtain MARTI01 from MARTI0 training data. Other preprocessing have been provided by the participants: `http://clopinet.com/isabelle/Projects/WCCI2008/Analysis.html#MARTIprepro`

The reference submission uploaded to the website of the challenge. (reference_hpc) is for HITON-PC (max−k=1, alpha=0.01) & Linear SVM (C = 0.001), using original for both algorithms. The test AUC results are:

MARTI0	0.9830
MARTI1	0.8595
MARTI2	0.7652

The best AUC performance of challenge participants are:

MARTI0	1.000
MARTI1	0.947
MARTI2	0.798

The best score on the LOCANET task obtained by participants is: 0.21

Appendix A: Generation of random probes

We describe a method aimed as assessing the fraction of non-causes in a subset of causes of a target variable selected by a causal discovery algorithm. The method consists in generating variables whose distribution resembles the real variables, but are either unrelated to the target, or related to it in a non causal way (consequences or confounders). Those variables, called "probes" by analogy to the probe method in feature selection, are intermixed with the real variables and a causal discovery algorithm is run. The fraction of probes in the variables selected as causes of the target may be used to determine the false positive rate and false discovery rate.

Notations

We call X the data matrix with p lines (patterns) and n columns (features/variables). We call R the matrix of random probes of dimension (p, r), which are unrelated to the target. We call C the matrix of confounders and consequences of dimension (p, c).

Generating variables unrelated to the target (R matrix)

Variables unrelated to the target are generated by taking blocks of variables in the original data matrix and permuting the order of the rows randomly. The resulting variables should be uncorrelated to the target, except for coincidental correlations due to the small number of samples. If the blocks are of size one, the variables generated are uncorrelated with one another. Otherwise, there are block correlations between them. We show the Matlab code in Appendix A1.

Generating consequences and confounders (C and E matrices)

To generate confounders, subsets of real variables (X matrix) and of the probes unrelated to the target (R matrix) are used as an input to a non-linear function. The same method is applied to generate consequences of the target by adding the target in the set of inputs. The code generating such probes in shown in Appendix A3.

As a non-linear function, we use a 2 layer neural network. The inputs are expected to lie approximately between -1 and 1 (or between 0 and 1). For all neurons, we use weights drawn randomly from $N(0,1)$/fanin. The hidden units use the tanh(ax) as squashing function. The slope a determines the amount of non-linearity added by the hidden layer because the second layer is connected both to the hidden units and directly to the inputs. Random noise drawn from $N(0,\varepsilon)$ is added to the output (ε is proportional to the output range). Finally, the distribution of the output values is mapped to the distribution of one of the real variables (this adds additional non-linearity). We show the Matlab code of the non-linear function in Appendix A2. In Figure A1, we show an example of function obtained for a univariate input between 0 and 1.

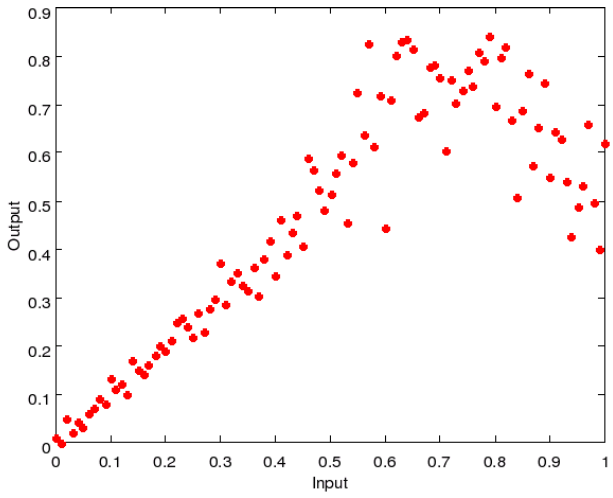


Figure A1: Example of artificial non-linear function. The example was obtained using the code of Appendix A2, with `x=[0:.01:1]; y=rand_func(x', sin(x'), 0.05, 1, 2, 1);`

Manipulations

Only probes are manipulated in test data in such a way that they become all independent of the target. This is achieved by permuting the values of each probe in test data.

Example architecture

In Figure A2, we show the architecture of the fake variable (probe) network. In Appendix A4, we reproduce the code of this probe network architecture:

- We do not know the causal relationships of the true variables to the target.
- We first draw `probe_num` random probes independent from the target (R).
- We reserve ½ of R, which we do not use as input to the probe network and thus remain fully independent of the target.
- We use ¼ of R as spouses of causes of the target to create `confounder_num` confounders (C).
- We use ¼ of R as spouses of the target to create `consequence_num` consequences (E).
- Random subsets of the confounders are also used to influence the consequences.
- The average fanin (number of variables influencing a probe) is monitored by the parameter "sparsity", which we chose to be 0.01 time the number of real variables. We use a parameter `slope_squash=2` to monitor the amount of non-linearity.

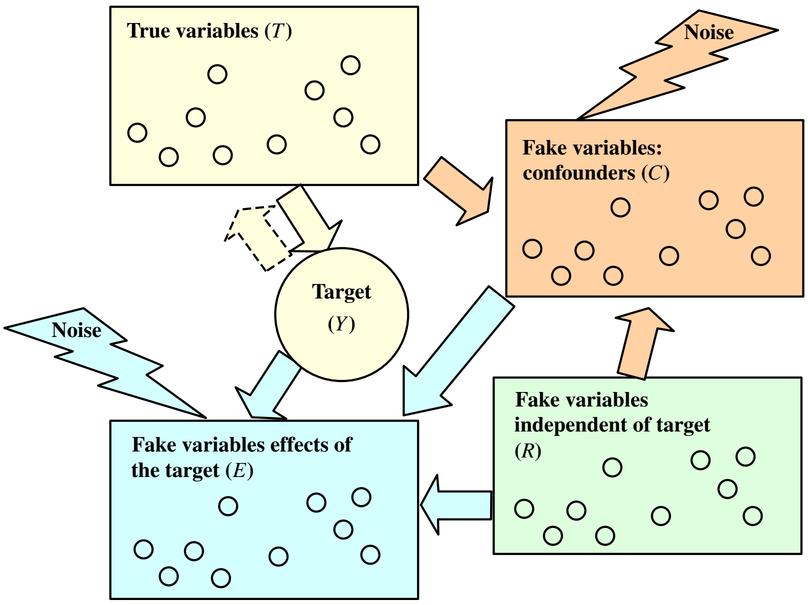

True variables (T)
Noise
Fake variables: confounders (C)
Target (Y)
Noise
Fake variables independent of target (R)
Fake variables effects of the target (E)

Figure A2: **Architecture of the probe network.** In yellow we indicate the dependencies of the real variables (some of which may be causes or consequences of the target). In green, we indicate the random probes drawn first, which are independent of the target. In orange, we indicate how the confounders are generated having as patents subsets of the green and yellow variables. In cyan, we indicate consequences are created using as parent the target and subsets of green and orange variables. This architecture exhibits the following conditional independencies: $T \perp R$, $Y \perp R$, $Y \perp C|T$, $E \perp T|(Y,C)$.

Testing

To test our simulator, we compute several correlation coefficients R for subsets of variables in unmanipulated and manipulated data. We call T_i, R_i, C_i, and E_i a column of the T, R, C, and

E matrices, respectively, and Y the target vector. We compute statistics for the absolute value, including:

- $R(T_i, R_j)$ – expected to be close to 0
- $R(T_i, R_j|C(T_i, R_j))$, $C(T_i, R_j)$ effect of T_i and R_j – expected to be **non-zero**
- $R(Y, R_j)$ – expected to be close to 0
- $R(Y, R_j|E(Y, R_j))$, $E(Y, R_j)$ effect of Y and R_j – expected to be **non-zero**
- $R(Y, C_j)$ – expected to be **non-zero**
- $R(Y, C_j|T)$ – expected to be close to 0, except in manipulated data
- $R(T_i, C_j)$ – expected to be **non-zero** for some pairs, except in manipulated data
- $R(Y, E_j)$ – expected to be **non-zero**, except in manipulated data
- $R(T_i, E_j|Y, C)$ – expected to be close to 0

To compute the conditional correlation coefficient of A_i and B_j given C, where C is a matrix of column vectors, we proceed as follows:

- standardize A_i, B_j and the columns of C
- project A_i, and B_j on the null space of C
- compute the correlation of the projections

It can be shown that if C is a single column then

$$R(A_i, B_j|C) = [R(A_i, B_j) - R(A_i, C)R(B_j, C)]/\sqrt{[(1 - R(A_i, C))^2(1 - R(B_j, C))^2]}$$

The verification code is reported in Appendix A5.

Here is a simple Matlab example that runs the code:

```
p=1000; % Number of samples
x=randn(1000,1);
y=sign(x);
x=x(:,ones(10,1))+randn(1000,10)/5; % replicate the same variable
and add noise
fnum=size(x, 2);
fprintf('Created dataset with %d features, with average correl to
target=%5.4f+-%5.4f\n', fnum, mean(condcor(x,y)), std(condcor(x,y)));
rp=20; % Completely random, not related to target
ca=10; % Confounders
ef=10; % Effects of target
nl=2;  % Level of non-linearity
mn=p/2; % Number of manipulated variables
% Add probes to the data matrix
[xx, parents]=add_probes(x, y, rp,ca,ef,[],[],[],[],nl,[],mn);

fprintf('\nAdded %d random probes:\n', (rp+ca+ef));
fprintf('%d not related to the target,\n', rp);
fprintf('%d confounders\n', ca);
fprintf('%d effects\n', ef);
fprintf('\n+++++++++++++++++++++++++++++++++\n');
fprintf('+++ Testing unmanipulated data +++\n');
fprintf('++++++++++++++++++++++++++++++++++\n\n');
test_net(xx(1:500,:), y(1:500), parents, fnum);
fprintf('\n++++++++++++++++++++++++++++++++\n');
fprintf('+++ Testing manipulated data +++\n');
fprintf('++++++++++++++++++++++++++++++++\n');
show_net_again=0;
test_net(xx(501:1000,:), y(501:1000), parents, fnum, show_net_again);
```

```
Created dataset with 10 features, with average correl to
target=0.7829+-0.0037
*** Creating 20 random probes by blocks of 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
*** Creating 10 confounders,
    with on average 1 parents drawn randomly from the 10 true variables
    and 1 random probe parents from a pool of 5 probes
Distillating
Keeping only 10/10 true variables most correlated to Y
Normalizing
Adding variables,
        average number of true parents=1
        average number of probe parents=1
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
*** Creating 10 effects of the target,
    with on average 1 parents drawn randomly from the 10 confounders
    and 1 random probe parents from a pool of 5 probes
Normalizing
Adding variables,
        average number of true parents=1
        average number of probe parents=1
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Added 40 random probes:
20 not related to the target,
10 confounders
10 effects
++++++++++++++++++++++++++++++++++
+++ Testing unmanipulated data +++
++++++++++++++++++++++++++++++++++

== Total number of variables: 50 ==
== Real variables (10): 1 ... 10
== Probes (20):
  4 spouses of true var:
(11 -> 16  18  20 )
(13 -> 17  19  22  23  24 )
(14 -> 16  21  24  25 )
(15 -> 17  19  22 )
  5 spouses of target:
(26 -> 35  38  40 )
(27 -> 34  37 )
(28 -> 31  33  39 )
(29 -> 36 )
```

```
(30 -> 31  32  36  37  38  39 )
  11 independent of target: Warning, some probes assigned to be
spouses are unused
== Confounders (10): 16 ... 25
(16 <- 10  6  14  11 )
(17 <- 4  13  15 )
(18 <- 10  4  11 )
(19 <- 10  13  15 )
(20 <- 10  11 )
(21 <- 5  4  14 )
(22 <- 1  15  13 )
(23 <- 2  13 )
(24 <- 8  13  14 )
(25 <- 7  2  14 )
== Effects (10): 31 ... 40
(31 <- 0  18  21  28  30 )
(32 <- 0  21  30 )
(33 <- 0  20  19  28 )
(34 <- 0  25  27 )
(35 <- 0  18  26 )
(36 <- 0  22  21  29  30 )
(37 <- 0  20  27  30 )
(38 <- 0  25  30  26 )
(39 <- 0  21  24  30  28 )
(40 <- 0  20  22  26 )

** Those should NOT be close to zero (ever) **
**> R (Y, Tj) -- Target dependent on true variables
    -- Top 1%:       <abs(R)>=0.7966+-0.0000
    -- Top 10%:      <abs(R)>=0.7966+-0.0000
    -- All:          <abs(R)>=0.7857+-0.0069

** Those should be close to zero **
==> R (Ti, Rj) -- Probes independent of the true variables
    -- Top 1%:       <abs(R)>=0.1129+-0.0003
    -- Top 10%:      <abs(R)>=0.0851+-0.0190
    -- All:          <abs(R)>=0.0310+-0.0242
==> R (Y, Rj) -- Probes independent of the target
    -- Top 1%:       <abs(R)>=0.1028+-0.0000
    -- Top 10%:      <abs(R)>=0.1006+-0.0030
    -- All:          <abs(R)>=0.0311+-0.0267
==> R (Y, Cj | T) -- Confounders independent of target given the true
variables (max of 50 confounders sampled)
    -- Top 1%:       <abs(R)>=0.0932+-0.0000
    -- Top 10%:      <abs(R)>=0.0932+-0.0000
    -- All:          <abs(R)>=0.0370+-0.0301
==> R (Ti, Ej | Y, C) -- True variables (top most corr w. Y)
independent of effects given the parents of the effects (max of 50
effects sampled)
    -- Top 1%:       <abs(R)>=0.1185+-0.0000
    -- Top 10%:      <abs(R)>=0.0862+-0.0143
    -- All:          <abs(R)>=0.0325+-0.0269
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and effects (same effects sampled)
    -- Top 1%:       <abs(R)>=0.7752+-0.0000
    -- Top 10%:      <abs(R)>=0.7461+-0.0118
    -- All:          <abs(R)>=0.5212+-0.2056
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and effects (all effects)
    -- Top 1%:       <abs(R)>=0.7752+-0.0000
    -- Top 10%:      <abs(R)>=0.7461+-0.0118
    -- All:          <abs(R)>=0.5212+-0.2056
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and the target (all effects)
    -- Top 1%:       <abs(R)>=0.7966+-0.0000
    -- Top 10%:      <abs(R)>=0.7966+-0.0000
    -- All:          <abs(R)>=0.7857+-0.0069

** Those should be close to zero ONLY in manipulated test data **
==> R (Ti, Cj) -- Confounders dependent on their parents (true
variables)
    -- Parents, which are true variables
    -- Top 1%:       <abs(R)>=0.8926+-0.0000
    -- Top 10%:      <abs(R)>=0.8926+-0.0000
    -- All:          <abs(R)>=0.4737+-0.3109
    -- Parents, which are probes
    -- Top 1%:       <abs(R)>=0.9526+-0.0000
    -- Top 10%:      <abs(R)>=0.9436+-0.0128
    -- All:          <abs(R)>=0.5310+-0.3201
==> R (Ti, Rj | C(Ti, Rj)), C(Ti, Rj) effect of Ti and Rj
    -- Dependency of true var and probes induced by confounders (max
of 50 confounders sampled)
    -- Top 1%:       <abs(R)>=0.7284+-0.0000
    -- Top 10%:      <abs(R)>=0.7275+-0.0013
    -- All:          <abs(R)>=0.3552+-0.2284
    -- For comparison, unconditioned dependency of same true var and
probes (same confounders sampled)
    -- Top 1%:       <abs(R)>=0.0518+-0.0000
    -- Top 10%:      <abs(R)>=0.0515+-0.0004
    -- All:          <abs(R)>=0.0339+-0.0102
    -- For comparison, unconditioned dependency of same true var and
probes (all samples)
    -- Top 1%:       <abs(R)>=0.0518+-0.0000
    -- Top 10%:      <abs(R)>=0.0515+-0.0004
    -- All:          <abs(R)>=0.0339+-0.0102
**> R (Y, Cj) -- Target dependent on counfounders
    -- Top 1%:       <abs(R)>=0.7135+-0.0000
    -- Top 10%:      <abs(R)>=0.7135+-0.0000
    -- All:          <abs(R)>=0.3563+-0.2551
==> R (Ci, Rj | E), E are effects of the target, Ci, and Rj are
parents of these effects
    -- Target spouses become dependent given their children (max of 50
effects sampled)
    -- Top 1%:       <abs(R)>=0.7728+-0.0000
    -- Top 10%:      <abs(R)>=0.6309+-0.2006
    -- All:          <abs(R)>=0.1670+-0.2078
    -- For comparison, the same without conditioning on the
effects(same effects)
    -- Top 1%:        <abs(R)>=0.0878+-0.0000
    -- Top 10%:       <abs(R)>=0.0819+-0.0083
    -- All:           <abs(R)>=0.0313+-0.0284
    -- For comparison, the same without conditioning on the effects
(all effects)
    -- Top 1%:        <abs(R)>=0.0878+-0.0000
    -- Top 10%:       <abs(R)>=0.0819+-0.0083
    -- All:           <abs(R)>=0.0313+-0.0284
    -- For comparison, effects and their probe parents (same effects)
    -- Top 1%:        <abs(R)>=0.8433+-0.0000
    -- Top 10%:       <abs(R)>=0.7681+-0.1063
    -- All:           <abs(R)>=0.3204+-0.2218
    -- For comparison, effects and their confounder parents (same
effects)
    -- Top 1%:        <abs(R)>=0.8768+-0.0000
    -- Top 10%:       <abs(R)>=0.8255+-0.0725
    -- All:           <abs(R)>=0.3394+-0.2670
==> R (Y, Rj | E(Y, Rj)), E(Y, Rj) effect of Y and Rj
    -- Target spouses and target become dependent given their children
(max of 50 souses sampled)
    -- Top 1%:        <abs(R)>=0.4694+-0.0000
    -- Top 10%:       <abs(R)>=0.4694+-0.0000
    -- All:           <abs(R)>=0.2432+-0.1422
    -- For comparison, correlation target spouses and target, without
conditioning (same spouses)
    -- Top 1%:        <abs(R)>=0.0985+-0.0000
    -- Top 10%:       <abs(R)>=0.0985+-0.0000
    -- All:           <abs(R)>=0.0420+-0.0342
    -- For comparison, correlation target spouses and target, without
conditioning (all spouses)
    -- Top 1%:        <abs(R)>=0.0985+-0.0000
    -- Top 10%:       <abs(R)>=0.0985+-0.0000
    -- All:           <abs(R)>=0.0420+-0.0342
**> R (Y, Ej) -- Effects of the target correlated to the target
    -- Top 1%:        <abs(R)>=0.8092+-0.0000
    -- Top 10%:       <abs(R)>=0.8092+-0.0000
    -- All:           <abs(R)>=0.6203+-0.2725
==> R (Ei, Cj) and R (Ei, Rj) -- Effects of the target correlated to
their other parents
    -- Parents, which are confounders
    -- Top 1%:        <abs(R)>=0.8768+-0.0000
    -- Top 10%:       <abs(R)>=0.8255+-0.0725
    -- All:           <abs(R)>=0.3394+-0.2670
    -- Parents, which are probes
    -- Top 1%:        <abs(R)>=0.8433+-0.0000
    -- Top 10%:       <abs(R)>=0.7681+-0.1063
    -- All:           <abs(R)>=0.3204+-0.2218

++++++++++++++++++++++++++++++++
+++ Testing manipulated data +++
++++++++++++++++++++++++++++++++
== Total number of variables: 50 ==
== Real variables (10): 1 ... 10
== Probes (20):
  4 spouses of true var:
  5 spouses of target:
  11 independent of target: Warning, some probes assigned to be spouses
are unused
== Confounders (10): 16 ... 25
  1.40+- 0.52 true variable parents,  1.50+- 0.53 parents unrelated to
target
== Effects (10): 31 ... 40
  1.50+- 0.53 confounder parents,  1.50+ 0.53 parents unrelated to
target

** Those should NOT be close to zero (ever) **
**> R (Y, Tj) -- Target dependent on true variables
    -- Top 1%:      <abs(R)>=0.7924+-0.0000
    -- Top 10%:     <abs(R)>=0.7924+-0.0000
    -- All:         <abs(R)>=0.7865+-0.0050

** Those should be close to zero **
==> R (Ti, Rj) -- Probes independent of the true variables
    -- Top 1%:      <abs(R)>=0.1304+-0.0015
    -- Top 10%:     <abs(R)>=0.0895+-0.0323
    -- All:         <abs(R)>=0.0274+-0.0265
==> R (Y, Rj) -- Probes independent of the target
    -- Top 1%:      <abs(R)>=0.0727+-0.0000
    -- Top 10%:     <abs(R)>=0.0715+-0.0018
    -- All:         <abs(R)>=0.0353+-0.0230
==> R (Y, Cj | T) -- Confounders independent of target given the true
variables (max of 50 confounders sampled)
    -- Top 1%:      <abs(R)>=0.0899+-0.0000
    -- Top 10%:     <abs(R)>=0.0899+-0.0000
    -- All:         <abs(R)>=0.0330+-0.0306
==> R (Ti, Ej | Y, C) -- True variables (top most corr w. Y)
independent of effects given the parents of the effects (max of 50
effects sampled)
    -- Top 1%:      <abs(R)>=0.1231+-0.0000
    -- Top 10%:     <abs(R)>=0.0990+-0.0103
    -- All:         <abs(R)>=0.0520+-0.0269
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and effects (same effects sampled)
    -- Top 1%:      <abs(R)>=0.1098+-0.0000
    -- Top 10%:     <abs(R)>=0.0971+-0.0053
    -- All:         <abs(R)>=0.0405+-0.0306
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and effects (all effects)
    -- Top 1%:      <abs(R)>=0.1098+-0.0000
    -- Top 10%:     <abs(R)>=0.0971+-0.0053
    -- All:         <abs(R)>=0.0405+-0.0306
    -- For comparison, unconditioned dependency of same true var (top
most corr w. Y) and the target (all effects)
    -- Top 1%:      <abs(R)>=0.7924+-0.0000
    -- Top 10%:      <abs(R)>=0.7924+-0.0000
    -- All:          <abs(R)>=0.7865+-0.0050

** Those should be close to zero ONLY in manipulated test data **
==> R (Ti, Cj) -- Confounders dependent on their parents (true
variables)
    -- Parents, which are true variables
    -- Top 1%:       <abs(R)>=0.0864+-0.0000
    -- Top 10%:      <abs(R)>=0.0864+-0.0000
    -- All:          <abs(R)>=0.0311+-0.0202
    -- Parents, which are probes
    -- Top 1%:       <abs(R)>=0.1102+-0.0000
    -- Top 10%:      <abs(R)>=0.0796+-0.0433
    -- All:          <abs(R)>=0.0347+-0.0252
==> R (Ti, Rj | C(Ti, Rj)), C(Ti, Rj) effect of Ti and Rj
    -- Dependency of true var and probes induced by confounders (max
of 50 confounders sampled)
    -- Top 1%:       <abs(R)>=0.0425+-0.0000
    -- Top 10%:      <abs(R)>=0.0425+-0.0001
    -- All:          <abs(R)>=0.0264+-0.0128
    -- For comparison, unconditioned dependency of same true var and
probes (same confounders sampled)
    -- Top 1%:       <abs(R)>=0.0437+-0.0000
    -- Top 10%:      <abs(R)>=0.0424+-0.0019
    -- All:          <abs(R)>=0.0270+-0.0125
    -- For comparison, unconditioned dependency of same true var and
probes (all samples)
    -- Top 1%:       <abs(R)>=0.0437+-0.0000
    -- Top 10%:      <abs(R)>=0.0424+-0.0019
    -- All:          <abs(R)>=0.0270+-0.0125
**> R (Y, Cj) -- Target dependent on counfounders
    -- Top 1%:       <abs(R)>=0.0803+-0.0000
    -- Top 10%:      <abs(R)>=0.0803+-0.0000
    -- All:          <abs(R)>=0.0368+-0.0188
==> R (Ci, Rj | E), E are effects of the target, Ci, and Rj are
parents of these effects
    -- Target spouses become dependent given their children (max of 50
effects sampled)
    -- Top 1%:       <abs(R)>=0.0934+-0.0000
    -- Top 10%:      <abs(R)>=0.0871+-0.0090
    -- All:          <abs(R)>=0.0328+-0.0279
    -- For comparison, the same without conditioning on the
effects(same effects)
    -- Top 1%:       <abs(R)>=0.0929+-0.0000
    -- Top 10%:      <abs(R)>=0.0870+-0.0083
    -- All:          <abs(R)>=0.0327+-0.0282
    -- For comparison, the same without conditioning on the effects
 (all effects)
    -- Top 1%:       <abs(R)>=0.0929+-0.0000
    -- Top 10%:      <abs(R)>=0.0870+-0.0083
    -- All:          <abs(R)>=0.0327+-0.0282
    -- For comparison, effects and their probe parents (same effects)
    -- Top 1%:       <abs(R)>=0.0674+-0.0000
    -- Top 10%:      <abs(R)>=0.0647+-0.0038
    -- All:          <abs(R)>=0.0355+-0.0240
    -- For comparison, effects and their confounder parents (same
effects)
    -- Top 1%:       <abs(R)>=0.0720+-0.0000
    -- Top 10%:      <abs(R)>=0.0699+-0.0029
    -- All:          <abs(R)>=0.0331+-0.0248
==> R (Y, Rj | E(Y, Rj)), E(Y, Rj) effect of Y and Rj
    -- Target spouses and target become dependent given their children
(max of 50 souses sampled)
    -- Top 1%:       <abs(R)>=0.0727+-0.0000
    -- Top 10%:      <abs(R)>=0.0727+-0.0000
    -- All:          <abs(R)>=0.0332+-0.0279
    -- For comparison, correlation target spouses and target, without
conditioning (same spouses)
    -- Top 1%:       <abs(R)>=0.0727+-0.0000
    -- Top 10%:      <abs(R)>=0.0727+-0.0000
    -- All:          <abs(R)>=0.0348+-0.0291
    -- For comparison, correlation target spouses and target, without
conditioning (all spouses)
    -- Top 1%:       <abs(R)>=0.0727+-0.0000
    -- Top 10%:      <abs(R)>=0.0727+-0.0000
    -- All:          <abs(R)>=0.0348+-0.0291
**> R (Y, Ej) -- Effects of the target correlated to the target
    -- Top 1%:       <abs(R)>=0.0886+-0.0000
    -- Top 10%:      <abs(R)>=0.0886+-0.0000
    -- All:          <abs(R)>=0.0364+-0.0272
==> R (Ei, Cj) and R (Ei, Rj) -- Effects of the target correlated to
their other parents
    -- Parents, which are confounders
    -- Top 1%:       <abs(R)>=0.0720+-0.0000
    -- Top 10%:      <abs(R)>=0.0699+-0.0029
    -- All:          <abs(R)>=0.0331+-0.0248
    -- Parents, which are probes
    -- Top 1%:       <abs(R)>=0.0674+-0.0000
    -- Top 10%:      <abs(R)>=0.0647+-0.0038
    -- All:          <abs(R)>=0.0355+-0.0240
```

Appendix A1: Generation of variables unrelated to the target

```
function Xp=create_random_probes(X, probe_num, block_size)
%Xp=create_random_probes(X, probe_num, block_size)
% Create a matrix Xp containing probes.
% This is done by permuting blocks of the original matrix.
% X              -- Data matrix p x n
% probe_num      -- dim(Xp, 2)=N
% block_size     -- number of features permuted in block. If
block_size=1,
%                   the all probes correspond to variables individually
permuted.
% Returns:
% Xp             -- matrix of probes of dim p x N
% Isabelle Guyon -- isabelle@clopinet.com -- October 2007

[p, n]=size(X);
N=probe_num;
if issparse(X)
    Xp=sparse(p, N);
else
    Xp=zeros(p, N);
end

beg0=1;
fin0=min(block_size, n);
beg1=beg0;
fin1=fin0;
while 1
    fprintf('%d ', fin1);
    % define a new block of data from X
    rng0=beg0:fin0;
    % define where to put it in Xp
    rng1=beg1:fin1;
    % Create a random permutation
    ip=randperm(p);
    % Assign values
    Xp(:,rng1)=X(ip,rng0);
    % next bounds in X
    beg0=fin0+1;
    if beg0>n, % restart at the beginning
        beg0=1;
        fin0=min(block_size, n);
    else
        fin0=min(fin0+block_size, n);
    end
    % next bounds in Xp
    beg1=fin1+1;
    if beg1>N, break; end
    fin1_new=fin1+length(beg0:fin0);
    if fin1_new>N
        fin0=fin0-(fin1_new-N);
        fin1=N;
    else
        fin1=fin1_new;
    end
end
fprintf('\n');
```

Appendix A2: Non-linear function used to generate confounders and consequences

```
function y=rand_func(x, r, noise, h, slope_squash, debug)
%y=rand_func(x, r, noise, h, slope_squash, debug)
% Take an x vector as an input and generates a random function from it
% We use a neural network of 2 layers with as many hidden units as
inputs
% and a direct connection from input to output.
% x: input vector (also works for data matrices, patterns in lines)
% Expects inputs roughly between 0 and 1 or between -1 and 1.
% r: variable of which we want to mimic the distribution
% noise: noise level of output_range*N(0, noise) added
% slope_squash: slope at origin of the squashing function (tanh)
% h: number of hidden units
% debug: debug flag

% Isabelle Guyon -- isabelle@clopinet.com -- October 2007

if nargin<2, r=[]; end
if nargin<3, noise=0.05; end
if nargin<4, h=[]; end
if nargin<5, slope_squash=1; end
if nargin<6, debug=0; end

[p, n]=size(x);
[pr, nr]=size(r);

% number of hidden units
if isempty(h), h=n; end
if debug, h=10; end

% First layer parameters
w1=randn(h,n)./n;
b1=randn(h,1)./n;
b1=b1(:,ones(1,p));
% Second layer parameters
w2=randn(1,n+h)./(n+h);
b2=randn./(n+h);

% Network computations
v=x*w1'+b1';
y1=tanh(slope_squash*v);
y1=[y1, x];
v=y1*w2'+b2';
if debug
    y=tanh(slope_squash*v); % Try also squashing
    [sv, vi]=sort(v);
    h1=figure; subplot(3,1,1); plot(sv, '.'); title('Blue: Raw v');
    [sy, si]=sort(y);
    h2=figure; subplot(3,1,1); plot(sy, '.'); title('Blue: Raw y');
    yorig=y;
    vorig=v;
end

% Add random noise
d=max(v)-min(v);
v=v+noise*d*randn(size(v));
if debug
    y=y+noise*randn(size(y));
    % Note: the effect of squashing and adding noise to y
    % at this level makes the noise not hoeoscedastic
    figure(h1); hold on; subplot(3,1,1); plot(v(si), 'r.');
```

```
title('Blue: Raw v distribution; Red: plus noise');
    figure(h2); hold on; subplot(3,1,1); plot(y(si), 'r.');
title('Blue: Raw y distribution; Red: plus noise');
    if ~isempty(r)
        % Mimic the distribution of r for y
        [ys, is]=sort(y);
        rs=sort(r);
        y(is)=rs;
    end
end

if ~isempty(r)
    % Mimic the distribution of r directly for v
    [vs, is]=sort(v);
    rs=sort(r);
    v(is)=rs;
end

if debug
    figure(h1); hold on; subplot(3,1,2); plot(v(vi), 'g.');
title('Green=NL mapping');
    figure(h2); hold on; subplot(3,1,2); plot(y(si), 'g.');
title('Green=NL mapping');
    figure(h1); hold on; subplot(3,1,3); plot(vorig, v, 'r.');
xlabel('v before'); ylabel('v after');
    figure(h2); hold on; subplot(3,1,3); plot(yorig, y, 'r.');
xlabel('y before'); ylabel('y after');
end

% In the end we find better to use the direct non-linear mapping of v
to r
y=v;

if debug
    figure; plot(vorig, y, 'r.'); xlabel('Net output'); ylabel('Output
mapped to matched desired distribution');
    if size(x, 2)==1
        figure; plot(x, y, 'r.'); xlabel('Input'); ylabel('Output');
    end
end
```

Appendix A3: Generation of variables related to others (may include the target)

```
function [Xc, parents]=create_confounders(X, Xp, Y, confounder_num,
true_parent_num, probe_parent_num, non_linearity_level, noise, top_num)
%[Xc, parents]=create_confounders(X, Xp, Y, confounder_num,
true_parent_num, probe_parent_num, non_linearity_level, noise, top_num)
% Create a matrix Xc containing confounders that are consequences of
real variables and probes.
% This is done by defining a sparse architecture and then applying
functions to the inputs to generate new inputs.
% The output is then made to resemble the distribution of a real
variable.
% X              -- Data matrix p x n containing real variables
% Xp              -- Data matrix p x m containing probes
% Y              -- Target values (p) (not provided if we want to
exclude
%                    the target)
% confounder_num -- dim(Xc, 2)= N
%                   the all probes correspond to variables individually
permuted.
% true_parent_num, probe_parent_num -- number of parents of the created
variables
% non_linearity_level -- Slope of the tanh in the hidden layer (the
larger
%                        the more non-linear. Use 1 for almost linear.
% noise -- Random noise level (as a fraction of the variable output
range).
% top_num      -- number of examples most correlated to Y used in X as
%                    input to confounders. Y is then not used as input.
% Returns:
% Xc             -- Matrix of probes of dim p x N
% parents        -- A cell array with lists of parents among input
variables
%                   The variables are numbers 1..n in X and n+1...n+m in
Xp
% If the target is given, we add it to all input variable sets (this
%                    creates consequences).

% Isabelle Guyon -- isabelle@clopinet.com -- October 2007

if nargin<8, noise=0.05; end
do_not_add_Y=0;
if nargin>=9,
    do_not_add_Y=1;
else
    top_num=size(X,2);
end

[p, n]=size(X);
[p, m]=size(Xp);
N=confounder_num;

if ~isempty(Y) & do_not_add_Y
    % Distillate the data
    fprintf('Distillating\n');
    idx_feat=balcor_select(X, Y);
    idx_good=idx_feat(1:top_num);
    Y=[];
    fprintf('Keeping only %d/%d true variables most correlated to Y\n',
length(idx_good), n);
else
    idx_good=[1:n]';
end
ng=length(idx_good);

if issparse(X)
```

```
    Xc=sparse(p, N);
else
    Xc=zeros(p, N);
end

% Divide variables by their maximum, to bring them between 0 and 1
Xs=[X Xp];
xmax=max(Xs);
fprintf('Normalizing\n');
for k=1:size(Xs,2)
    if xmax(k)~=0
        Xs(:,k)=Xs(:,k)./xmax(k);
    end
end

% Average number of variables to be selected
parents=\\;
fprintf('Adding variables,\n\taverage number of true parents=%d',
true_parent_num);
fprintf('\n\taverage number of probe parents=%d\n', probe_parent_num);
percent_done=0;
old_percent_done=0;
for k=1:N
    percent_done=floor(k/N*100);
    if ~mod(percent_done,10) & percent_done~=old_percent_done,
        fprintf('%d%% ', percent_done);
    end
    old_percent_done=percent_done;
    % Select a random subset of real variables and of probes
    fanin_real=max(1, min(ceil(true_parent_num*(1+(rand-0.5))), ng));
    rp_real=idx_good(randperm(ng)); % We prefer the variables
correlated to the target
    fanin_probe=max(1, min(ceil(probe_parent_num*(1+(rand-0.5))), m));
    rp_probe=randperm(m)';
    parents\k\=[rp_real(1:fanin_real); n+rp_probe(1:fanin_probe)];
    % Select a real variable at random
    rp=randperm(n);
    r=X(:,rp(1)); % This one should not be standardized
    % Input the variables to the non-linear function
    h=fanin_real+fanin_probe; % number of hidden units
    Xc(:,k)=rand_func([Xs(:,parents\k\), Y] , r, noise, h,
non_linearity_level);
    % Add Y as parent
    if ~isempty(Y)
        parents\k\=[0; parents\k\];
    end
end
function [idx_feat, cor_val]=balcor_select(X, Y, feat_num)
%[idx_feat, cor_val]=balcor_select(X, Y, feat_num)
% feature selection with correlation coefficient
% which balances the 2 classes by subsampling the second one.

% Isabelle Guyon -- isabelle@clopinet.com -- October 2007
```

```
pidx=find(Y==1);
nidx=find(Y==-1);
% take a random subset of negative class elements of the same size as
% the number of positive
rp=randperm(length(nidx));
nidx=nidx(rp(1:min(length(pidx), length(nidx))));
RR=condcor(X([pidx; nidx],:), Y([pidx; nidx]));
[cor_val, idx_feat]=sort(-abs(RR));

if nargin>2
    idx_feat=idx_feat(1:feat_num);
    cor_val=-cor_val(1:feat_num);
else
    cor_val=-cor_val;
end
```

Appendix A4: Generation of all probes

```
function [X, parents]=add_probes(X, Y, probe_num, confounder_num,
consequence_num, conf_true_parent_num, conf_probe_parent_num,
cons_true_parent_num, cons_probe_parent_num, non_linearity_level,
noise, num_manipulated, top_num, noise_Y)
%[X, parents]=add_probes(X, Y, probe_num, confounder_num,
consequence_num, conf_true_parent_num, conf_probe_parent_num,
cons_true_parent_num, cons_probe_parent_num, non_linearity_level,
noise, num_manipulated, top_num, noise_Y)
% Create a matrix X containing probes, urelated to the target or
% confounders and consequeces. The architecture is built in.
% X                   -- Data matrix p x n
% Y                   -- Target values (p) (not provided if we want to
exclude
%                     the target)
% probe_num           -- Number of probes not consequences of real
variables or
%                     the target
% confounder_num      -- Number of confounders
% consequence_num     -- Number of consequences
% conf_true_parent_num, conf_probe_parent_num -- number of parents of
% confounders
% cons_true_parent_num, cons_probe_parent_num -- number of parents of
% consequences
% non_linearity_level -- Slope of the tanh in the hidden layer (the
larger
%                         the more non-linear. Use 1 for almost linear.
% noise               -- Random noise level (as a fraction of the
variable output range).
% num_manipulated     -- for the num_manipulated last entries the probes
values
%                     are randomized, making all probes independent of
the
%                     target.
% top_num             -- number of examples most correlated to Y used in
X as
```

```
%                    input to confounders
% tone_Y_down        -- Multipicative factor to tone Y down as a cause
of its
%                    effects
%
% Returns:
% Xnew           -- Matrix of probes of dim p x N
% parents        -- A cell array with lists of parents of all the
variables
%                   Variables are numbered 1 to n. The target is 0.

% Isabelle Guyon -- isabelle@clopinet.com -- October 2007

[p, n]=size(X);

if nargin<6 | isempty(conf_true_parent_num), conf_true_parent_num=1;
end
if nargin<7 | isempty(conf_probe_parent_num), conf_probe_parent_num=1;
end
if nargin<8 | isempty(cons_true_parent_num), cons_true_parent_num=1;
end
if nargin<9 | isempty(cons_probe_parent_num), cons_probe_parent_num=1;
end
if nargin<10 | isempty(non_linearity_level), non_linearity_level=1; end
if nargin<11 | isempty(noise), noise=0.05; end
if nargin<12 | isempty(num_manipulated), num_manipulated=0; end
if nargin<13 | isempty(top_num), top_num=size(X,2); end
if nargin<14 | isempty(noise_Y), noise_Y=0; end

% Repartition of the probes marginally independent of the target
spouse_cause=floor(probe_num/4);
spouse_target=floor(probe_num/4);
true_random=probe_num-spouse_cause-spouse_target;

% Creation of probes marginally independent of the target
block_size=max(1,round(n/10));
fprintf('*** Creating %d random probes by blocks of %d\n', probe_num,
block_size);
Xp=create_random_probes(X, probe_num, block_size);

% Split probes into spouses and purely random
% First shuffle them
rp=randperm(size(Xp, 2));
Xp=Xp(:, rp);
% Then split into 3 parts
Xp1=Xp(:,1:spouse_cause);
Xp2=Xp(:,spouse_cause+1:spouse_cause+spouse_target);
Xp3=Xp(:,spouse_cause+spouse_target+1:probe_num);
clear Xp;

% Create confounders, not consequences of the target, using true
variables
% and a subset of the previously drawn probes
```

```
fprintf('*** Creating %d confounders, \n     with on average %d parents
drawn randomly from the %d true variables \n     and %d random probe
parents from a pool of %d probes\n', ...
    confounder_num, conf_true_parent_num, size(X,2),
conf_probe_parent_num, size(Xp1, 2) );
[Xc, p1_parents]=create_confounders(X, Xp1, Y, confounder_num,
conf_true_parent_num, conf_probe_parent_num, non_linearity_level,
noise, top_num);

% Create effects of the target and a subset of the previously drawn probes
fprintf('\n*** Creating %d effects of the target, \n     with on average
%d parents drawn randomly from the %d confounders \n     and %d random
probe parents from a pool of %d probes\n', ...
    consequence_num, cons_true_parent_num, size(Xc,2),
cons_probe_parent_num, size(Xp2, 2));
if noise_Y>0,
    % flip noise_Y*p examples
    rp=randperm(p);
    rp=rp(1:round(noise_Y*p));
    Y(rp)=-Y(rp);
end
[Xe, p2_parents]=create_confounders(Xc, Xp2, Y, consequence_num,
cons_true_parent_num, cons_probe_parent_num, non_linearity_level,
noise);

% Add everything together
X=[X, Xp1, Xc, Xp2, Xe, Xp3];
total_var=n+probe_num+confounder_num+consequence_num;
parents=cell(total_var,1);
% Confounder parents: X (n variables) and Xp1 (spouse_cause variables)
% have no parents: start at n+spouse_cause+1
parents(n+spouse_cause+1:n+spouse_cause+confounder_num)=p1_parents;
% Effect parents: Xp2 have no parents, start at
% n+spouse_cause+confounder_num+spouse_target.
% Offset the indices of p2_parents by n+spouse_cause
for k=1: length(p2_parents)
    p2_parents\k\=p2_parents\k\+n+spouse_cause;
    p2_parents\k\(1)=0; % The target value is not offset
end
parents(n+spouse_cause+confounder_num+spouse_target+1:total_var-
true_random)=p2_parents;

if num_manipulated>0
    probe_idx=(n+1):size(X,2);
    manip_idx=(p-num_manipulated+1):p;
    for k=1:length(probe_idx)
        rp=randperm(num_manipulated);
        X(manip_idx,probe_idx(k))=X(manip_idx(rp),probe_idx(k));
    end
end
```

Appendix A5: Verification code

```
function test_net(X, Y, parents, true_num, debug, lean)
```

```
%test_net(X, Y, parents, true_num, , lean)
% Test the independencies in the net
% X -- data matrix (p samples x n variables)
% Y -- target vector (dim p)
% parents -- cell aray of lists of parents of the variables
% true_num -- number of true variables (first in the X matrix)
% debug -- flag to show or not the network
% lean -- flag to remove the calculation of the pvalues

[p, n]=size(X);
if nargin<4, true_num=[]; end
if nargin<5, debug=0; end % display net
if nargin<6, lean=1; end

% Maximum number of conditional correlation coeff computed (for
% computational reasons)
maxval=50;

[parents, children, no_parent_idx, effect_idx, confounder_idx,
true_var_idx, rp_idx, spouse_target_idx, spouse_true_idx,
other_rp_idx]= ...
    draw_net(parents, true_num, debug);

% Reduce the test matrix by selecting a balanced number of examples
pidx=find(Y==1);
nidx=find(Y==-1);
rp=randperm(length(nidx));
nidx=nidx(rp(1:min(length(pidx), length(nidx)))); % The positive class
is usually more depleted
X=X([pidx;nidx],:);
Y=Y([pidx;nidx]);

% Find the true variables
T=X(:,true_var_idx);
% Find the random probes indep Y
R=X(:,rp_idx);
% Find the consequences of the target
E=X(:,effect_idx);
% The rest are the counfounders
C=X(:,confounder_idx);

% Find candidate causes (top maxval variables most correlated to the
target)
fprintf('\n** Those should NOT be close to zero (ever) **\n');
[RR, PP]=condcor(Y, T, [], lean);
fprintf('**> R (Y, Tj) -- Target dependent on true variables\n');
show_R(RR,PP);
[SR, IR]=sort(-abs(RR));
top_cause_idx=IR(1:min(maxval, length(IR)));

% Compute the other dependencies
warning off
% Independence of random probes with other variable
```

```
fprintf('\n** Those should be close to zero **\n');
[RR, PP]=condcor(T, R, [], lean);
fprintf('==> R (Ti, Rj) -- Probes independent of the true
variables\n');
show_R(RR,PP);

[RR, PP]=condcor(Y, R, [], lean);
fprintf('==> R (Y, Rj) -- Probes independent of the target\n');
show_R(RR,PP);

if ~isempty(C)
    % Confounders and target shielded by true variables
    fprintf('==> R (Y, Cj | T) -- Confounders independent of target
given the true variables (max of %d confounders sampled)\n', maxval);
    RR=[]; PP=[];
    for k=1:min(maxval, length(confounder_idx)) % Too long if we
calculate for all
        p_idx=parents\confounder_idx(k)\;
        p_idx=intersect(p_idx, true_var_idx);
        [rr,pp]=condcor(Y, C(:,k), X(:,p_idx), lean);
        RR=[RR rr];
        PP=[PP pp];
    end
    show_R(RR,PP);

    % Effects and true variables shielded by parents
    fprintf('==> R (Ti, Ej | Y, C) -- True variables (top most corr w.
Y) independent of effects given the parents of the effects (max of %d
effects sampled)\n', maxval);
    RR=[]; PP=[];
    ms=min(length(effect_idx), maxval);
    for k=1:ms
        p_idx=parents\effect_idx(k)\;
        p_idx=intersect(p_idx, confounder_idx);
        [rr,pp]=condcor(T(:, top_cause_idx), E(:,k), [Y X(:,p_idx)],
lean);
        RR=[RR rr];
        PP=[PP pp];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, unconditioned dependency of same
true var (top most corr w. Y) and effects (same effects sampled)\n');
    [RR,PP]=condcor(T(:, top_cause_idx), E(:, 1:ms), [], lean);
    show_R(RR,PP);
    fprintf('    -- For comparison, unconditioned dependency of same
true var (top most corr w. Y) and effects (all effects)\n');
    [RR, PP]=condcor(T(:, top_cause_idx), E, [], lean);
    show_R(RR,PP);
    fprintf('    -- For comparison, unconditioned dependency of same
true var (top most corr w. Y) and the target (all effects)\n');
    [RR, PP]=condcor(T(:, top_cause_idx), Y, [], lean);
    show_R(RR,PP);
end
```

```
if ~isempty(C)
    fprintf('\n** Those should be close to zero ONLY in manipulated
test data **\n');
    % Confounders
    % C and parents
    fprintf('==> R (Ti, Cj) -- Confounders dependent on their parents
(true variables)\n');
    fprintf('    -- Parents, which are true variables\n');
    RR=[]; PP=[];
    for k=1:length(confounder_idx)
        p_idx=parents\confounder_idx(k)\;
        p_idx=intersect(p_idx, true_var_idx);
        [rr,pp]=condcor(X(:,p_idx), C(:,k), [], lean);
        RR=[RR; rr];
        PP=[PP; pp];
    end
    show_R(RR,PP);
    fprintf('    -- Parents, which are probes\n');
    RR=[]; PP=[];
    for k=1:length(confounder_idx)
        p_idx=parents\confounder_idx(k)\;
        p_idx=intersect(p_idx, rp_idx);
        [rr,pp]=condcor(X(:,p_idx), C(:,k), [], lean);
        RR=[RR; rr];
        PP=[PP; pp];
    end
    show_R(RR,PP);

    % Induced by C
    fprintf('==> R (Ti, Rj | C(Ti, Rj)), C(Ti, Rj) effect of Ti and
Rj\n    -- Dependency of true var and probes induced by confounders
(max of %d confounders sampled)\n', maxval);
    RR=[]; PP=[];
    for k=1:min(length(confounder_idx), maxval)
        p_idx=parents\confounder_idx(k)\;
        tpar_idx=intersect(p_idx, true_var_idx);
        rpar_idx=intersect(p_idx, rp_idx);
        [rr, pp]=condcor(X(:,tpar_idx), X(:,rpar_idx), C(:,k), lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);

    fprintf('    -- For comparison, unconditioned dependency of same
true var and probes (same confounders sampled)\n');
    RR=[]; PP=[];
    for k=1:min(length(confounder_idx), maxval)
        p_idx=parents\confounder_idx(k)\;
        tpar_idx=intersect(p_idx, true_var_idx);
        rpar_idx=intersect(p_idx, rp_idx);
        [rr, pp]=condcor(X(:,tpar_idx), X(:,rpar_idx), [], lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);

    fprintf('    -- For comparison, unconditioned dependency of same
true var and probes (all samples)\n');
    RR=[]; PP=[];
    for k=1:length(confounder_idx)
        p_idx=parents\confounder_idx(k)\;
        tpar_idx=intersect(p_idx, true_var_idx);
        rpar_idx=intersect(p_idx, rp_idx);
        [rr, pp]=condcor(X(:,tpar_idx), X(:,rpar_idx), [], lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);

    % Y and C
    fprintf('**> R (Y, Cj) -- Target dependent on counfounders\n');
    [RR, PP]=condcor(Y, C, [], lean);
    show_R(RR,PP);
end

if ~isempty(E)
    % Parents of the effects
    fprintf('==> R (Ci, Rj | E), E are effects of the target, Ci, and
Rj are parents of these effects\n    -- Target spouses become
dependent given their children (max of %d effects sampled)\n', maxval);
    RR=[]; PP=[];
    for k=1:min(length(effect_idx), maxval)
        p_idx=parents\effect_idx(k)\;
        rpar_idx=intersect(p_idx, spouse_target_idx);
        tpar_idx=intersect(p_idx, confounder_idx);
        [rr, pp]=condcor(X(:,tpar_idx) , X(:,rpar_idx), E(:,k), lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, the same without conditioning on
the effects(same effects)\n');
    RR=[]; PP=[];
    for k=1:min(length(effect_idx), maxval)
        p_idx=parents\effect_idx(k)\;
        rpar_idx=intersect(p_idx, spouse_target_idx);
        tpar_idx=intersect(p_idx, confounder_idx);
        [rr, pp]=condcor(X(:,tpar_idx), X(:,rpar_idx), [], lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, the same without conditioning on
the effects(all effects)\n');
    RR=[]; PP=[];
    for k=1:length(effect_idx)
        p_idx=parents\effect_idx(k)\;
        rpar_idx=intersect(p_idx, spouse_target_idx);
        tpar_idx=setdiff(setdiff(p_idx, rpar_idx), [0]);
        [rr, pp]=condcor(X(:,tpar_idx), X(:,rpar_idx), [], lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, effects and their probe parents
(same effects)\n');
    RR=[]; PP=[];
    for k=1:min(length(effect_idx), maxval)
        p_idx=parents\effect_idx(k)\;
        rpar_idx=intersect(p_idx, spouse_target_idx);
        tpar_idx=intersect(p_idx, confounder_idx);
        [rr, pp]=condcor(E(:,k), X(:,rpar_idx), [], lean);
        RR=[RR; rr(:)];
       PP=[PP; pp(:)];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, effects and their confounder
parents (same effects)\n');
    RR=[]; PP=[];
    for k=1:min(length(effect_idx), maxval)
        p_idx=parents\effect_idx(k)\;
        rpar_idx=intersect(p_idx, spouse_target_idx);
        tpar_idx=intersect(p_idx, confounder_idx);
        [rr, pp]=condcor(E(:,k), X(:,tpar_idx), [], lean);
        RR=[RR; rr(:)];
        PP=[PP; pp(:)];
    end
    show_R(RR,PP);


    % Target spouses
    fprintf('==> R (Y, Rj | E(Y, Rj)), E(Y, Rj) effect of Y and Rj\n
-- Target spouses and target become dependent given their children
(max of %d souses sampled)\n', maxval);
    RR=[]; PP=[];
    ms=min(length(spouse_target_idx), maxval);
    for k=1:ms
        c_idx=childrenspouse_target_idx(k);
        [rr, pp]=condcor(Y, X(:,spouse_target_idx(k)), X(:,c_idx),
lean);
        RR=[RR rr];
        PP=[PP pp];
    end
    show_R(RR,PP);
    fprintf('    -- For comparison, correlation target spouses and
target, without conditioning (same spouses)\n');
     [RR, PP]=condcor(Y, X(:,spouse_target_idx(1:ms)), [], lean);
    show_R(RR,PP);
    fprintf('    -- For comparison, correlation target spouses and
target, without conditioning (all spouses)\n');
    [RR, PP]=condcor(Y, X(:,spouse_target_idx), [], lean);
    show_R(RR,PP);

    % Effects and Y
    fprintf('**> R (Y, Ej) -- Effects of the target correlated to the
target\n');
    [RR, PP]=condcor(Y, E, [], lean);
    show_R(RR,PP);
    % Effects and the other parents
    fprintf('==> R (Ei, Cj) and R (Ei, Rj) -- Effects of the target
correlated to their other parents\n');
    fprintf('    -- Parents, which are confounders\n');
    RR=[]; PP=[];
    for k=1:length(effect_idx)
        p_idx=parents\effect_idx(k)\;
        p_idx=intersect(p_idx, confounder_idx);
        [rr,pp]=condcor(E(:,k), X(:,p_idx), [], lean);
        RR=[RR rr];
        PP=[PP pp];
    end
    show_R(RR,PP);
    fprintf('    -- Parents, which are probes\n');
    RR=[]; PP=[];
    for k=1:length(effect_idx)
        p_idx=parents\effect_idx(k)\;
        p_idx=intersect(p_idx, spouse_target_idx);
        [rr,pp]=condcor(E(:,k), X(:,p_idx), [], lean);
        RR=[RR rr];
        PP=[PP pp];
    end
    show_R(RR,PP);

end



function [parents, children, no_parent_idx, effect_idx, confounder_idx,
true_var_idx, rp_idx, spouse_target_idx, spouse_true_idx,
other_rp_idx]=draw_net(parents, true_num, debug)
%[parents, children, no_parent_idx, effect_idx, confounder_idx,
%true_var_idx, rp_idx, spouse_target_idx, spouse_true_idx,
other_rp_idx]=draw_net(parents, true_num, debug)
% Show the network.
% Inputs:
% parents    -- A cell array containing lists of variable parents
%                if true_num is given, it is assumed that the first few
variables are true variables
%                the variables are numbered 1, ..., i, ...n and
parents\i\ is the list of parents
%                of variable i.
%                if true_num=[], parents\1\ is the list of true
```

```
variables
%                and parent\i+1\ are the parents of i.
% true_num  -- Number of true variables.
% debug     -- debug blag: if 1, show the whole structure.
% Returns:
% parents   -- parents of the variables numbered 1, ..., i, ...n:
parents\i\
%                is the list of parents of variable i.
% children  -- children\i\ is the list of children of variable i.
% no_parent_idx -- indices of variables having no parents (includes
true
%                 and random probes independent of the target.
% effect_idx -- indices of probes which are effects of the target.
% confounder_idx -- indices of probes which are consequences of true
variables
% true_var_idx -- indices of true variables
% rp_idx      -- indices of random probes independent of the target
% spouse_target_idx -- indices of spouses of the target (probes)
% spouse_true_idx -- indices of spouses of true variables (probes)
% other_rp_idx -- indices of other random probes, indept of target
% length(other_rp_idx)+length(spouse_target_idx)+length(spouse_true_idx)
% ==length(rp_idx)

% Isabelle Guyon -- isabelle@clopine.,com -- October 2007

if nargin<2 | isempty(true_num),
    true_var_idx=parents\1\;
    parents=parents(2:length(parents));
else
    % Find the true variables
    true_var_idx=1:true_num;
end
if nargin<3, debug=1; end

% Invert the index
children=cell(size(parents));
no_parent_idx=[];
effect_idx=[];
confounder_idx=[];
for k=1:length(parents)
    par=parents\k\;
    if isempty(par)
        no_parent_idx=[no_parent_idx k];
    else
        if par(1)==0
            effect_idx=[effect_idx k];
            par=par(2:length(par));
        else
            confounder_idx=[confounder_idx k];
        end
        for j=1:length(par)
            children\par(j)\=[children\par(j)\ k];
        end
    end
end

% Find the random probes indep Y
rp_idx=setdiff(no_parent_idx, true_var_idx);

% Spouses:
spouse_true_idx=[];
spouse_target_idx=[];
for k=1:length(rp_idx)
    c_idx=childrenrp_idx(k);
    if ~isempty(c_idx)
        if ~isempty(intersect(c_idx, effect_idx))
            spouse_target_idx=[spouse_target_idx, rp_idx(k)];
        else
            spouse_true_idx=[spouse_true_idx, rp_idx(k)];
        end
    end
end
other_rp_idx=setdiff(setdiff(rp_idx, spouse_target_idx),
spouse_true_idx);

if true_num+length(rp_idx)+length(confounder_idx)+length(effect_idx)
~=length(parents)
    error('Number of variables do no add up');
end
if length(other_rp_idx)+length(spouse_target_idx)+length(spouse_true_idx)
~=length(rp_idx)
    error('Number of probes do no add up');
end

fprintf('== Total number of variables: %d ==\n', length(parents));
if ~isempty(true_num)
    fprintf('== Real variables (%d): 1 ... %d\n', true_num, true_num);
else
    fprintf('== Real variables (%d):\n', length(true_var_idx));
end
fprintf('== Probes (%d): \n', length(rp_idx));
fprintf('  %d spouses of true var: \n', length(spouse_true_idx));
if debug
    for k=1:length(spouse_true_idx)
        c_idx=childrenspouse_true_idx(k);
        fprintf('(%d ->',  spouse_true_idx(k));
        for j=1:length(c_idx)
            fprintf(' %d ', c_idx(j));
        end
        fprintf(')\n');
    end
end
fprintf('  %d spouses of target: \n', length(spouse_target_idx));
if debug
    for k=1:length(spouse_target_idx)
        c_idx=childrenspouse_target_idx(k);
        fprintf('(%d ->',  spouse_target_idx(k));
        for j=1:length(c_idx)
            fprintf(' %d ', c_idx(j));
        end
        fprintf(')\n');
    end
end
fprintf('  %d independent of target: ', length(other_rp_idx));
MM=max(other_rp_idx);
mm=min(other_rp_idx);
if(MM-mm+1== length(other_rp_idx))
    fprintf('%d ... %d\n', mm, MM);
else
    fprintf('Warning, some probes assigned to be spouses are
unused\n');
end
fprintf('== Confounders (%d): ', length(confounder_idx));
MM=max(confounder_idx);
mm=min(confounder_idx);
if(MM-mm+1== length(confounder_idx))
    fprintf('%d ... %d\n', mm, MM);
else
    if ~isempty(confounder_idx),
        fprintf('Warning, wierd set\n');
    end
end
if debug
    for k=1:length(confounder_idx)
        p_idx=parentsconfounder_idx(k);
        fprintf('(%d <-'',  confounder_idx(k));
        for j=1:length(p_idx)
            fprintf(' %d ', p_idx(j));
        end
        fprintf()\n);
    end
else
    nt=[];
    np=[];
    for k=1:length(confounder_idx)
        p_idx=parentsconfounder_idx(k);
        pt_idx=intersect(p_idx, true_var_idx);
        pr_idx=intersect(p_idx, rp_idx);
        nt=[nt length(pt_idx)];
        np=[np length(pr_idx)];
    end
    fprintf(' %5.2f+-%5.2f true variable parents, %5.2f+-%5.2f parents
unrelated to target\n', mean(nt), std(nt), mean(np), std(np));
end

fprintf('== Effects (%d): ', length(effect_idx));
MM=max(effect_idx);
mm=min(effect_idx);
if(MM-mm+1== length(effect_idx))
```

```
    fprintf('%d ... %d\n', mm, MM);
else
    if ~isempty(effect_idx), fprintf('Warning, wierd set\n'); end
end
if debug
    for k=1:length(effect_idx)
        p_idx=parentseffect_idx(k);
        fprintf('(%d <-',  effect_idx(k));
        for j=1:length(p_idx)
            fprintf(' %d ', p_idx(j));
        end
        fprintf(')\n');
    end
else
    nc=[];
    np=[];
    for k=1:length(effect_idx)
        p_idx=parentseffect_idx(k);
        pc_idx=intersect(p_idx, confounder_idx);
        pr_idx=intersect(p_idx, rp_idx);
        nc=[nc length(pc_idx)];
        np=[np length(pr_idx)];
    end
    fprintf(' %5.2f+-%5.2f confounder parents, %5.2f+%5.2f parents
unrelated to target\n', mean(nc), std(nc), mean(np), std(np));

end




function [r, pval]=condcor(x, y, C, lean)
%[r, pval]=condcor(x, y, C, lean)
% Computes the correlation between the column vectors x and y
% given the column vectors of matrix C.
% lean -- flag, if 1, do not compute pvalue

if nargin<3, C=[]; end
if nargout>1
    if lean
        pval_compute=0;
    else
        pval_compute=1;
    end
else
    pval_compute=0;
end
pval=[];

debug=0;

[p, n]=size(x);
[pp, m]=size(y);
```

```
if p~=pp, error('wrong dimensions'); end

v=1/sqrt(p);

% Center and normalize
x=v*standard(x);
y=v*standard(y);
if ~isempty(C)
    C=v*standard(C);
end

if debug & length(C)==length(x)
    r_verif= (x'*y - (x'*C) * (y'*C))/sqrt((1-(x'*C)^2) * (1-(y'*C)^2))
end

% Project on null space
if ~isempty(C)
    proj=C*pinv(C);
    x=x-proj*x;
    y=y-proj*y;
    % Center and normalize again
    x=v*standard(x);
    y=v*standard(y);
end

% Compute dot product
if pval_compute
    [R, P]=corrcoef([x, y]);
    r=R(1:n,n+1:n+m);
    pval=P(1:n,n+1:n+m);
else
    r=x'*y;
end

return

% verification:
% condcor(a, b, c) = (condcor(a, b)-condcor(a, c)*condcor(b,
c))/sqrt((1-condcor(a, c)^2)*(1-condcor(b, c)^2) )



function X=standard(X)
%X=standard(X)
% Standardize matrix of column vectors

[p, n]=size(X);
M=mean(X);
S=std(X,1);
X=(X-M(ones(p,1),:));
S(find(S==0))=1;
X=X./S(ones(p,1),:);
X=X./S(ones(p,1),:);
```

```
function show_R(RR, PP)
%show_R(RR, PP)
% Show statistics about vectors of correlation coefficients RR and their
% pvalues PP.

% Isabelle Guyon -- isabelle@clopinet.com -- October 2007

RR=full(abs(RR(:)));
PP=full(PP(:));
[SR0, IR]=sort(-RR);

fprintf('    -- Top 1%%:\t');
mval=max(1, round(length(SR0)/100));
SR=-SR0(1:mval);
fprintf('    <abs(R)>=%5.4f+-%5.4f', mean(SR), std(SR));
if ~isempty(PP),
    SP=PP(IR(1:mval));
    fprintf(', <pval>=%5.4f+-%5.4f\n', mean(SP), std(SP));
else fprintf('\n');
end

fprintf('    -- Top 10%%:\t');
mval=max(1, round(length(SR0)/10));
SR=-SR0(1:mval);
fprintf('    <abs(R)>=%5.4f+-%5.4f', mean(SR), std(SR));
if ~isempty(PP),
    SP=PP(IR(1:mval));
    fprintf(', <pval>=%5.4f+-%5.4f\n', mean(SP), std(SP));
else fprintf('\n');
end

fprintf('    -- All:\t');
fprintf('\t    <abs(R)>=%5.4f+-%5.4f', mean(RR), std(RR));
if ~isempty(PP), fprintf(', <pval>=%5.4f+-%5.4f\n', mean(PP), std(PP));
else fprintf('\n'); end
```

Appendix B: Probe method for scoring causes & consequences

This appendix provides an algorithm to **compute the AUC for ROC curves** plotting hit rate *vs.* false alarm rate in the classification of **"relevant" *vs.* "irrelevant" variables.** "Relevancy" can take one of several meanings, including dependency to the target, causal relationships to the target, etc. The method is therefore applicable to **variable selection,** where relevant variables are those, which are predictive of a given outcome (e.g. a target variable), and irrelevant variables are not. It is also applicable to **causal discovery**, where a score can indicate causal proximity to the target, with the goal of separating *e.g.,* causes from non-causes or direct causes from other variables.

The assumption we make is that **we do not know the truth values** of the variable classification (relevant *vs.* irrelevant) but **we know the "null distribution"** of irrelevant variables and we can draw as many artificial examples of such irrelevant variables as we want (we call them "probes"). It is assumed that an empirical variable ranking (from most relevant to least relevant) can be established using training data (samples of variable values) and an algorithm of our choice. For instance, such ranking may be established using a variable score, where a low score indicates that the variable is more likely to belong to one of the classes (*e.g.,* the "relevant variables") and a high score that it belongs to the other (*e.g.,* the "irrelevant ones"). Using the ranking method, we compute the AUC for sets of variables intermixed with "random probes", as an estimate of the AUC for the classification "relevant" *vs.* "irrelevant" variables.

The algorithm (Matlab implementation in Appendix B5)

The original data consists of a matrix of m lines (samples) and n_r columns (real variables). The n_r real variables include a n_+ positive examples ("relevant" variables) and a n_- negative ($n_r = n_+ + n_-$). It is not known which variables are relevant (truth values) nor how many of them are relevant, thus n_+ and n_- are not known.

1) A number n_p of artificial random variables called "probes" are drawn from an assumed "null distribution". In turn m samples are drawn from these probes and the resulting ($m \times n_p$) values are added to the original matrix to form an ($m \times (n_r + n_p)$) matrix.
2) All real variables and probes are ranked with a given algorithm, in decreasing order of relevance (most relevant variables come first).
3) The sum of the ranks of the probes SPR is formed.
4) The area under the ROC curve for the data including probes is estimated as

$$\mathrm{PAUC} = [\mathrm{SPR} - n_p \cdot (n_p + 1)/2]/(n_p \cdot n_r)$$

In the asymptotic case of infinite number of real variables and probes, PAUC is linearly related to the AUC for the classification "relevant" *vs.* "irrelevant" variables:

$$\mathrm{PAUC} = (n_+/n_r)\mathrm{AUC} + 0.5(n - /n_r).$$

This monotonic dependency allows us to use PAUC as a surrogate for the real AUC for algorithm comparison and model selection. In the finite sample case, we will use the following estimator of the PAUC standard deviation:

$$\sigma = 0.5\mathrm{sqrt}[\mathrm{sen}(1 - \mathrm{sen})/n_r + \mathrm{spe}(1 - \mathrm{spe})/n_p]$$

where spe $= 1 - k/\mathrm{neg}$, sen $= (r_k - k)/\mathrm{pos}$, r_k is the rank of the k^{th} probe and k maximizes the average of sen and spe. The algorithm is justified in what follows.

Calculation of the AUC and the Gini index

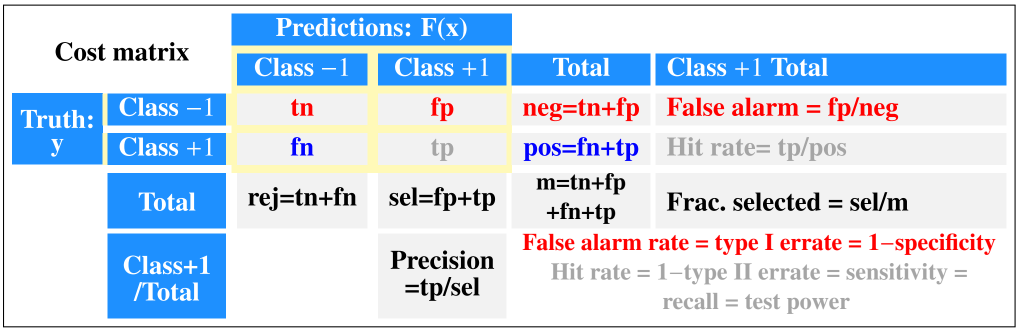
Assume we are given a ranked list of objects belonging to one of 2 classes, a positive and a negative class (for instance, causes and non-causes). We have "pos" examples of the positive class and "neg" examples of the negative class, and neg + pos = m = tot (the total number of examples).

We can compute, for each value of the rank:
fp: the number of false positive
tp: the number of true positive
fn: the number of false negative

tn: the number of true negative.
We have tp+fn=pos and tn+fp=neg

We define:
fpr (false positive rate or false alarm rate)=fp/neg
fnr (false negative rate)=fn/pos
Hit rate=sensitivity=tp/pos=1−fnr
Specificity=tn/neg=1−fpr
sel (the fraction of selected up to the rank)=fp+tp

Figure B1 shows how these statistics relate to one another.



Cost matrix		Predictions: F(x) Class −1	Class +1	Total	Class +1 Total
Truth: y	Class −1	tn	fp	neg=tn+fp	False alarm = fp/neg
	Class +1	fn	tp	pos=fn+tp	Hit rate= tp/pos
	Total	rej=tn+fn	sel=fp+tp	m=tn+fp +fn+tp	Frac. selected = sel/m
	Class+1 /Total		Precision =tp/sel		

False alarm rate = type I errate = 1−specificity
Hit rate = 1−type II errate = sensitivity = recall = test power

Compare $F(x) = \text{sign}(f(x))$ to the target y, and report:

- **Error rate = (fn + fp)/m**
- **{Hit rate, False alarm rate} or {Hit rate, Precision} or {Hit rate, Frac.selected}**
- **Balanced error rate (BER) = (fn/pos + fp/neg)/2 = 1 − (sensitivity + specificity)/2**
- **F measure = 2 precision.recall/(precision+recall)**

Vary the decision threshold $F(x) = \text{sign}(f(x) + \theta)$ and plot:

- **ROC curve: Hit rate *vs.* False alarm rate**
- **Lift curve: Hit rate *vs.* Fraction selected**
- **Precision/recall curve: Hit rate *vs.* Precision**

Figure B1: Performance Assessment

To avoid notation confusions, in what follows, if we are considering the real variables only, we use:

tot = n_r = number of real variables
pos = n_+ = number of examples of the positive class
neg = n_- = number of the negative class

If we are adding probe variables, we use:

tot = $n_r + n_p$ = number of variables including real and probes
pos = n_r = number of real variables
neg = n_p = number of probes

The ROC curve (Figure B2) plots the “hit rate” *vs.* the “false alarm rate” i.e. (1 − fnr) *vs.* fpr. The AUC is the area under the ROC curve. Note that it is identical to the area under the curve plotting sensitivity (aka “hit rate”) *vs.* specificity (1 − fpr).

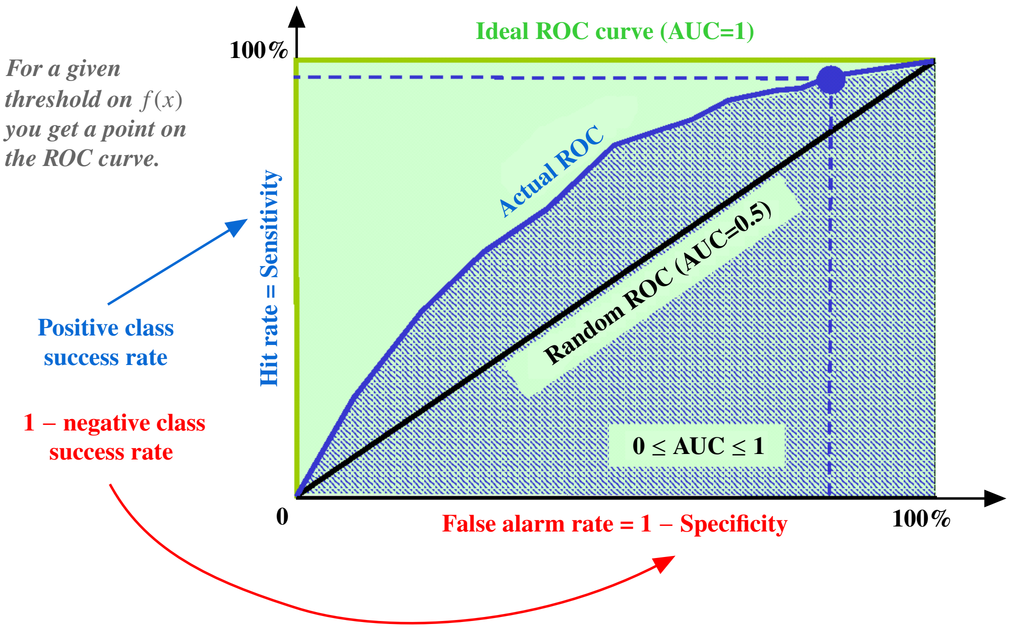

For a given threshold on $f(x)$ you get a point on the ROC curve.
Positive class success rate
1 – negative class success rate
100%
Hit rate = Sensitivity
Ideal ROC curve (AUC=1)
Actual ROC
Random ROC (AUC=0.5)
$0 \leq$ AUC ≤ 1
0
False alarm rate = 1 – Specificity
100%

Figure B2: ROC Curve

The lift curve (often used in marketing) plots “hit rate” *vs.* the fraction of selected “sel” (Figure B3). The Gini index is defined as the ratio M/O and it can be shown (Appendix B1) that **Gini = 2 AUC − 1**.

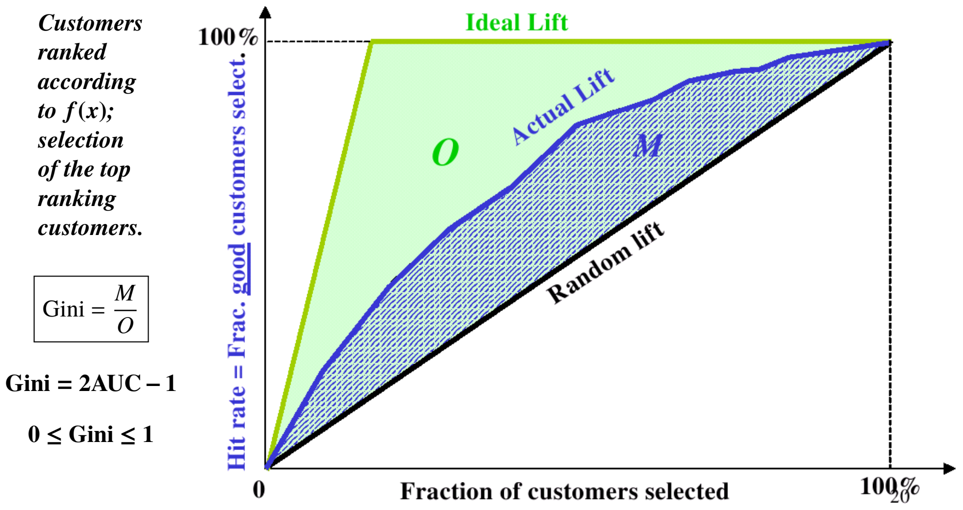

Customers ranked according to $f(x)$; selection of the top ranking customers.
$\boxed{\text{Gini} = \frac{M}{O}}$
Gini = 2AUC – 1
$0 \leq$ Gini ≤ 1
100%
Hit rate = Frac. good customers select.
Ideal Lift
O
Actual Lift
M
Random lift
0
Fraction of customers selected
100%
20

Figure B3: Lift Curve

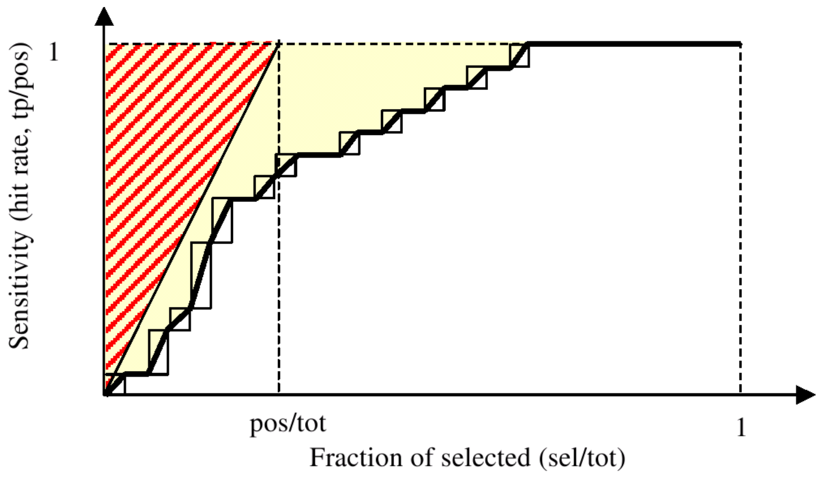


Figure B4: The yellow area represents the area above the lift curve. It is estimated by the trapeze method as [sum(rank_of_pos)−pos/2] /(pos*tot). The red shaded area is the area above the ideal lift, which is equal to 0.5 pos/tot.

This provides a means of computing efficiently the AUC using the area under the lift curve, because it is easy to compute the area under the lift curve. The area above the lift curve (AALC) can be upper and lower bounded by Lebesgue integrals using the sum of the ranks of the objects of the positive class (when those are sorted with the most relevant coming first) normalized by pos*tot (Figure B4):

$$[\text{sum(ranks\_of\_pos)} - \text{pos}]/(\text{pos} * \text{tot}) < \text{AALC} < \text{sum(ranks\_of\_pos)}/(\text{pos} * \text{tot})$$

Hence the estimation of the AALC by the trapeze method:

$$\text{AALC} \sim [\text{sum(ranks\_of\_pos)} - \text{pos}/2]/(\text{pos} * \text{tot})$$

Thus the area M is:

$$M = 0.5 - [\text{sum(ranks\_of\_pos)} - \text{pos}/2]/(\text{pos} * \text{tot})$$

The area O is given by:

$$O = 0.5 - 0.5 * \text{pos/tot}$$

Thus

$$\text{Gini} = M/O = \{0.5 - [\text{sum(ranks\_of\_pos)} - \text{pos}/2]/(\text{pos} * \text{tot})\}/(0.5 - 0.5 * \text{pos/tot})$$

$$\mathbf{Gini = [pos * (tot + 1) - 2\,sum(ranks\_of\_pos)]/[pos * (tot - pos)]}$$

Note that by symmetry with the negative class, we also have:

$$\mathbf{Gini = [neg * (tot + 1) - 2\,sum(ranks\_of\_neg)]/[neg * (tot - neg)]}$$

where sum(ranks_of_neg) is the sum of the ranks of the negative class when the ranking is such that the most likely to be negative come first, i.e. the ranking is done in order of increasing probability of being « relevant ».

Note that we can sort one way or the other. For instance, if we sort in order of increasing probability of being « relevant »(object believe to be from the negative class come first) and compute the sum of the ranks of the positive class and call it Sp we can relate it to sum(ranks_of_pos), the quantity defined above when sorting in the other direction:

$$\text{sum(ranks\_of\_pos)} = \text{sum}_{\text{pos}}(\text{tot} - j + 1) = \text{pos} * \text{tot} - \text{Sp} + \text{pos}$$

thus

$$\begin{aligned}\text{Gini} &= [\text{pos} * \text{tot} - 2(\text{pos} * \text{tot} - \text{Sp} + \text{pos}) + \text{pos}]/[\text{pos} * (\text{tot} - \text{pos})] \\ &= [2\text{Sp} - \text{pos} * (\text{tot} + 1)]/(\text{pos} * \text{neg})\end{aligned}$$

and

$$\text{AUC} = (\text{Gini} + 1)/2 = 0.5[2\text{Sp} - \text{pos} * \text{tot} - \text{pos} + \text{pos} * \text{tot} - \text{pos}^2]/(\text{pos} * \text{neg})$$

$$\mathbf{AUC = [Sp - pos * (pos + 1)/2]/(pos * neg)}$$

This last formula is the basis for the algorithm shown in Appendix B2.

We can also sort in decreasing order of relevance (most relevant objects believed to be from the positive class come first) and compute the sum of the ranks of the positive class and call it Sn. Similarly as before, we have:

$$\mathbf{Gini = [2Sn - neg * (tot + 1)]/[neg * (tot - neg)]} \qquad (1)$$

Thus the alternative formula for the AUC:

$$\mathbf{AUC = [Sn - neg * (neg + 1)/2]/(pos * neg)} \qquad (2)$$

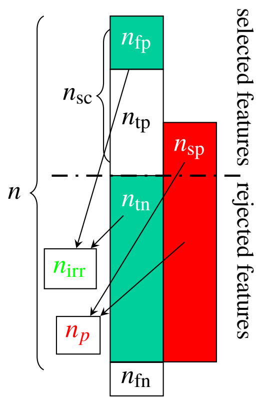
$$\text{FDR} = n_{\text{fp}}/n_{\text{sc}}$$

fp = false positive=features falsely found relevant
sc = selected candidate features
n_{fp} is unknown, but FPR can be calculated from pval or using the probe method.
Bound the FDR:

$$\begin{aligned}\text{FPR} &= n_{\text{fp}}/n_{\text{irr}} \geq n_{\text{fp}}/n \quad \text{(irr=irrelevant feat.)} \\ \text{FDR} &= (n_{\text{fp}}/n)(n/n_{\text{sc}}) \leq \text{FPR}\, n/n_{\text{sc}}\end{aligned}$$

$$\text{FDR} \leq \text{FPR}\, n/n_{\text{sc}} \leq \alpha$$

We obtain FPR $\leq \alpha n_{\text{sc}}/n$, intermediate between FPR $\leq \alpha$ and FPR $\leq \alpha/n$.



$$\text{FPR} = n_{\text{fp}}/n_{\text{irr}} \cong n_{\text{sp}}/n_p$$

Figure B5: False Discovery Rate

Relationship between ROC curve and negative lift curve

While in marketing the lift curve is the most useful way of visualizing the data, for our purpose, we rather focus on the negative class for the purpose of using the "probe" method. In Figure B6, we represent the non-normalized "negative lift curve", that is the number of false positive as a function of the total number of examples selected. The negative lift curve is obtained by normalizing the x axis by "tot" and the y axis by "neg".

If we change coordinates to the green axes, we obtain the non-normalized ROC curve. The ROC curve is obtained by normalizing the number of true positive by "pos" to get the sensitivity and the number of true negative by "neg" to get the specificity (Here we adopt as the definition of the ROC curve the plot sensitivity *vs.* specificity, which has same AUC as hit rate *vs.* false alarm rate and is obtained by reversing the x axis of the ROC curve).

From this diagram, we easily see how the AUC relates to the area under the false positive rate A (negative lift) and the ideal negative lift A^*. The AUC is the green shaded area, after normalizing by pos neg. hence:

$$\text{AUC} = (1 - A - A^*)(\text{tot/pos}).$$

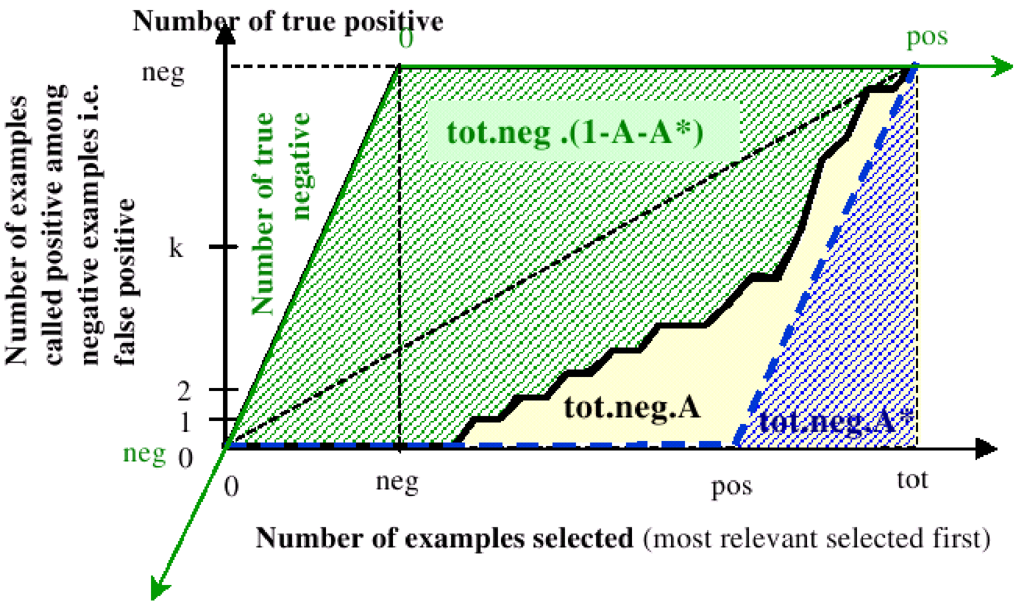

Number of true positive
0
pos
neg
k
2
1
neg
0
Number of examples called positive among negative examples i.e. false positive
Number of true negative
tot.neg .(1-A-A*)
tot.neg.A
tot.neg.A*
0
neg
pos
tot
Number of examples selected (most relevant selected first)

Figure B6: Relationship between ROC curve and lift curve

From Figure B6, we also see that for a given point on the ROC curve, the sensitivity is given by:

$$\textbf{specificity} = \mathbf{1} - \boldsymbol{k}/\textbf{neg} \tag{3}$$
$$\textbf{sensitivity} = (\boldsymbol{r_k} - \boldsymbol{k})/\textbf{pos} \tag{4}$$

where r_k is the rank of the k^{th} negative example in the example ordering, where most relevant come first.

We easily confirm Equation (2) with

$$\text{AUC} = (1/\text{neg}) \sum_{k=1:\text{neg}} \text{sensitivity} \tag{5}$$

AUC error bar

Many estimators of the AUC error bars have been proposed. Some are easy to compute but provide loose bounds, others are more accurate but very computationally expensive. For the purpose of the challenge, we propose to compromise and use an empirical formula easy to justify and which gives satisfactory results in numerical experiments.

We define the balanced accuracy (BAC) as:

$$\text{BAC} = 0.5(\text{sensitivity} + \text{specificity})$$

where, if we call tp the number of true positive and tn the number of true negative, we define sensitivity=tp/pos (accuracy of classification for positive examples) and specificity=tn/neg (accuracy of classification for negative examples). In the case where the score upon which the ranking is based is binary (e.g. hard classification decisions are used rather than a discriminant value), we have exactly AUC = BAC = 1 − BER, where BER is the balanced error rate defined as 1 − BAC (see Appendix B3 for a proof).

The idea is to **approximate the AUC with the maximum BAC on the ROC curve** (Figure B7). Subsequently, we will use the BAC error bar to estimate the AUC error bar. From Equations (3) and (4) giving the sensitivity and specificity, we see that our approximation amounts to computing:

$$\text{AUC} \cong \max \text{BAC} = \max_k 0.5[(r_k - k)/\text{pos} + 1 - k/\text{neg}]$$

In what follows, we call $k*$ the value of k maximizing BAC and sen and spe the corresponding values of the sensitivity and specificity.

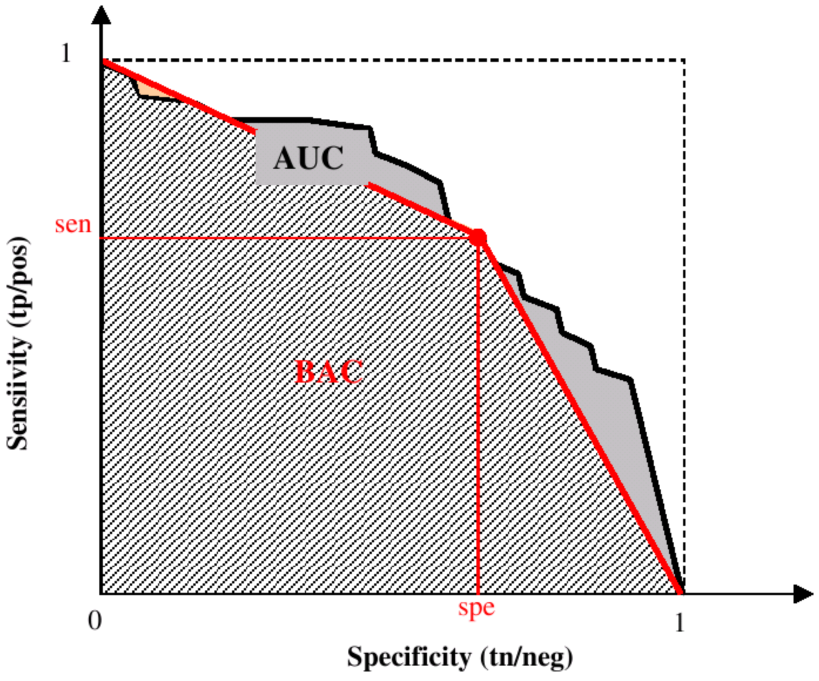


Figure B7: Approximation of the ROC curve. We replace the AUC by the largest BAC.

A BAC or a BER error bar estimator is obtained as follows. As is known, for i.i.d. errors corresponding to Bernoulli trials with a probability of error p, the standard deviation of the error rate E computed on a data set of size n is sqrt($p(1-p)/n$). This result can be adapted to the balanced error rate. Let us call pos the number of examples of the positive class, neg the number of examples of the negative class, p_+ the probability of error on examples of the positive class (one minus the expected value of the sensitivity), p_- the probability of error on examples of the negative class (one minus the expected value of the specificity), and E_+ and E_- their corresponding empirical estimates. Both processes generating errors on the positive or negative class are Bernoulli processes. By definition, the balanced error rate is BER = $(1/2)(E_+ + E_-)$, and its variance is var(BER) = $(1/4)(\mathrm{var}(E_+) + \mathrm{var}(E_-))$. Therefore, the standard deviation of the BER (and that of the BAC) using n_+ and n_- examples is:

$$\sigma = 0.5\sqrt{(p_+(1-p_+)/\mathrm{pos} + p_-(1-p_-)/\mathrm{neg})}$$

For sufficiently large data sets, we may substitute p_+ by E_+ and p_- by E_- to compute σ. Equivalently, since sensitivity = $1-E_+$ and specificity = $1-E_+$ we obtain the following estimator of the BAC standard deviation:

$$\boldsymbol{\sigma = 0.5\sqrt{(\mathbf{sen}(1-\mathbf{sen})/\mathbf{pos} + \mathbf{spe}(1-\mathbf{spe})/\mathbf{neg})}} \quad (6)$$

where we abbreviate sensitivity by sen and specificity by spe.

Application to the probe method

Assume that we are using the "probe" method and inject artificial probes, which are examples of the negative class for which we know the truth value (negative). The "real variables" may be either from the positive class or the negative class. Let us call:
$\boldsymbol{n_r}$: the total number of real variables
$\boldsymbol{n_p}$: the total number of probes
$\boldsymbol{n_{sp}}$: the number of selected probes
It is common to plot the fraction of probes selected n_{sp}/n_p as a function of the number of variables selected (Figure B8).

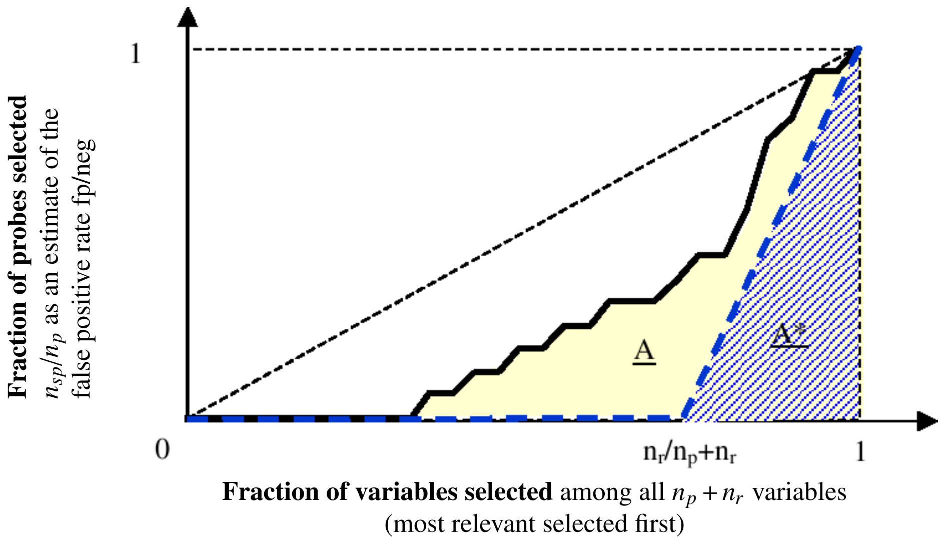


Figure B8: Area under the fraction of probe selected.

Doing that, we assume that we rank the variables in decreasing order of relevance (best first). We can define the sum of the rank of the probes SPR for that order. The area A is given by

$$A = 1 - (\mathrm{SPR} - n_p/2)/(n_p(n_p + n_r)).$$

The area A^* corresponding to the best achievable A (where all the probes show up last in the ranking) is given by

$$A^* = 0.5n_p/(n_p + n_r) = 0.5(1 - n_r/(n_p + n_r)).$$

Using Equation (1), we get for the probes

$$\mathbf{PGini = [2SPR - \boldsymbol{n_p} \cdot (\boldsymbol{n_p} + \boldsymbol{n_r} + 1)]/(\boldsymbol{n_p} \cdot \boldsymbol{n_r}),}$$

therefore

$$\mathrm{PGini} = (1 - 2A)(n_p + n_r)/n_r$$
$$\mathrm{PGini} = (1 - 2A)/(1 - 2A^*)$$
$$\mathbf{PGini = (0.5 - \boldsymbol{A})/(0.5 - \boldsymbol{A}^*)}$$

This last formula is equivalent to that of Figure B4.

Simply, the AUC for the probe method, which we call PAUC is obtained by computing the regular AUC for truth values +1 for all real variables and −1 for all probes (instead of +1 for the positive class variables and −1 for the negative class variables, in the absence of probes). Thus, the real variables become the positive class and the probes the negative class. For Equation (2), we get:

$$\mathbf{PAUC = [SPR - \boldsymbol{n_p} \cdot (\boldsymbol{n_p} + 1)/2]/(\boldsymbol{n_p} \cdot \boldsymbol{n_r})}$$

We show in Appendix B4 that asymptotically (for an infinite number of examples and probes):

$$\mathbf{PAUC = (\boldsymbol{n_+}/\boldsymbol{n_r})AUC + 0.5\boldsymbol{n_-}/\boldsymbol{n_r}}$$

where AUC is the true AUC, which cannot be computed and n_+ and n_- are the unknown number for positive and negative examples for the real variables.

An error bar on PAUC is obtained in the finite sample case from Equation (6):

$$\boldsymbol{\sigma} \mathbf{= 0.5\sqrt{(sen(1 - sen)/pos + spe(1 - spe)/neg)}}$$

with (from Equations (3) and (4))

$$\mathbf{spe = 1 - \boldsymbol{k}/neg}$$
$$\mathbf{sen = (\boldsymbol{r_k} - \boldsymbol{k})/pos}$$

for the value of k, which maximizes: BAC = 0.5(sen + spe).

The error BAR on PAUC may be use to determine the significance of the difference between two ranking methods yielding values of PAUC P_1 and P_2 and corresponding standard deviations σ_1 and σ_2. The difference will be called significant e.g. if $\mathrm{abs}(P_1 - P_2) > 2\sqrt{(\sigma_1^2 + \sigma_2^2)}$.

Numerical simulations:

We illustrate the result PAUC = (n_+/n_r)AUC + 0.5(n_-/n_r) with some simple numerical simulation. The code is reported in Appendix B6. In this example, we have 2000 "real variables" and 2000 "probes". We vary the fraction of positive examples (n_+/n_r) and compute a noisy score for variables as

```
D=Y+0.5*randn(size(Y))+k*noise*randn(size(Y));
```

From D we compute the "real" AUC and PAUC. We plot PAUC as a function of AUC (Figure B9).

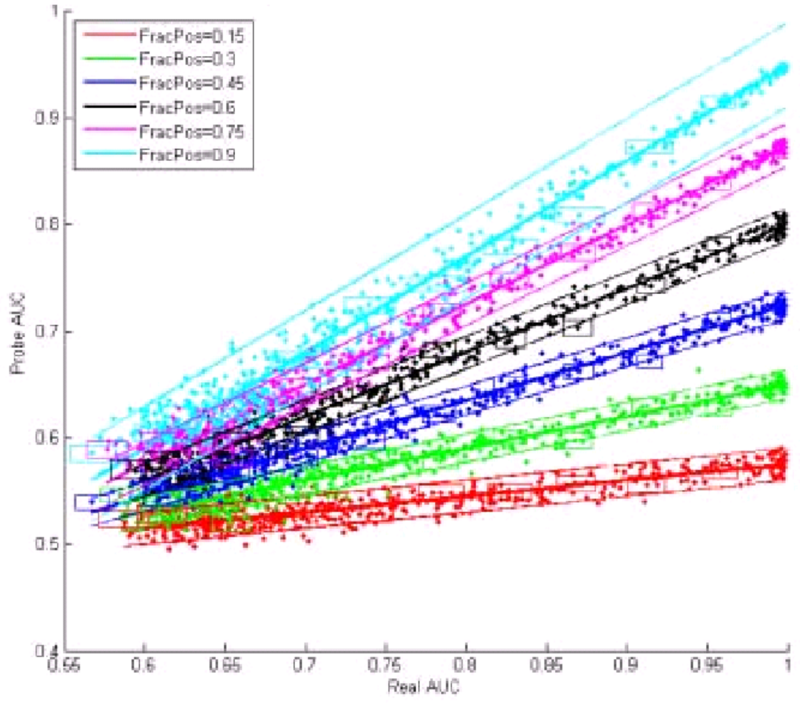


Figure B9: Relationship between the real AUC and that estimated by the probe method. The dots represent samples of the pairs {"real"AUC, PAUC} for various fractions of positive examples, when 2000 real variables and 2000 probes are used. The boxes show the 1 sigma error bars for AUC and PAUC. The thick lines are plots of $y = (n_+/n_r)\text{AUC} + 0.5(n_-/n_r)$. The thin lines indicate the estimated error tube.

We verified that for all fractions of positive examples considered the regression coefficients match closely the theoretical values obtained in the final sample size limit:

1) frac_pos=0.15, frac_neg=0.85, w=0.148399, 2*b=0.851758
2) frac_pos=0.3, frac_neg=0.7, w=0.297821, 2*b=0.702991
3) frac_pos=0.45, frac_neg=0.55, w=0.445261, 2*b=0.556114
4) frac_pos=0.6, frac_neg=0.4, w=0.601203, 2*b=0.398861
5) frac_pos=0.75, frac_neg=0.25, w=0.737146, 2*b=0.271517
6) frac_pos=0.9, frac_neg=0.1, w=0.8821, 2*b=0.124806

In the finite sample case, we have errors both on the estimates of both AUC and PAUC. Note that in practice we would not be able to compute the "real" AUC. Still, to verify the validity of our error bar estimates, we can use it here. We compute:

- The average empirical sigma as the average distance of the points to the line $y = (n_+/n_r)\text{AUC} + 0.5(n_-/n_r)$.
- The average theoretical sigma as the $\sigma_{\text{th}} = \text{mean}\left(\sqrt{\sigma^2_{\text{AUC}} + \sigma^2_{\text{PAUC}}}\right)$, computing σ_{AUC} and σ_{PAUC} with formula (6).

1) Average empirical sigma=0.0089, Average theoretical sigma=0.0149
2) Average empirical sigma=0.0091, Average theoretical sigma=0.0127
3) Average empirical sigma=0.0085, Average theoretical sigma=0.0121
4) Average empirical sigma=0.0092, Average theoretical sigma=0.0122
5) Average empirical sigma=0.0095, Average theoretical sigma=0.0132
6) Average empirical sigma=0.0126, Average theoretical sigma=0.0171

We see that our estimate is slightly pessimistic, but gives the right order of magnitude. To visualize our error bar estimates, we drew boxes of sides $2\sigma_{\text{AUC}} \times 2\sigma_{\text{PAUC}}$ around a few points. The box usually overlaps with the thick line. We also drew thin lines at $\sigma_{\text{th}}/\cos(\alpha)$, where α is the slope of the line. This allows us to draw an error bar taking into account both the error for estimating AUC and that for estimating PAUC.

Appendix B1: Proof of Gini = 2AUC − 1

$$\begin{aligned}
L &= \text{lift}\\
\text{Hitrate} &= \text{tp/pos}\\
\text{Farate} &= \text{fp/neg}\\
\text{Selected} &= \text{sel/tot} = (\text{tp}+\text{fp})/\text{tot} = \text{pos/tot}\cdot\text{tp/pos} + \text{neg/tot}\cdot\text{fp/neg}\\
&= \frac{\text{pos}}{\text{tot}}\,\text{Hitrate} + \frac{\text{neg}}{\text{tot}}\,\text{Farate}\\
\text{AUC} &= \text{sum Hitrate } d(\text{Farate})\\
L &= \text{sum Hitrate } d(\text{Selected})\\
&= \text{sum Hitrate } d\left(\frac{\text{pos}}{\text{tot}}\,\text{Hitrate} + \frac{\text{neg}}{\text{tot}}\,\text{Farate}\right)\\
&= \frac{\text{pos}}{\text{tot}}\,\text{sum Hitrate } d\,\text{Hitrate} + \frac{\text{neg}}{\text{tot}}\,\text{sum Hitrate } d\,\text{Farate}\\
&= \frac{1}{2}\,\frac{\text{pos}}{\text{tot}} + \frac{\text{neg}}{\text{tot}}\,\text{AUC}\\
2L-1 &= -(1-\text{pos/tot}) + 2(1-\text{pos/tot})\text{AUC} = (1-\text{pos/tot})(2\text{AUC}-1)\\
\text{Gini} &= (L-1/2)/(1-\text{pos/tot})/2\\
&= (2L-1)/(1-\text{pos/tot}) = 2\text{AUC}-1
\end{aligned}$$

Appendix B2:

```
function area = auc(Output, Target)
```

```
%area = auc(Output, Target)
% This computation gives the same results as the AUC when there are no
% ties.
% Inputs:
%  Output -- Matrix of classifier discriminant values of dim (num
pattern, num tries)
%  Target -- Vector of corresponding +-1 target values.
%  Returns:
%  area -- Area under the ROC curve.
% We still need to work out the case of ties.
% From Hollander and Wolfe pp 107 & 117.

% Isabelle Guyon -- isabelle@clopinet.com -- June 2005

% Compute the Wilcoxon statistic
midx=find(Target<0);
nidx=find(Target>0);
m=length(midx);
n=length(nidx);
[u,i]=sort(Output);
S(i)=1:n+m;
W=sum(S(nidx));

% Compute the Mann-Withney statistic
U=W-n*(n+1)/2;

% Compute the AUC
area=U/(m*n);
```

Appendix B3: Demonstration that AUC = 1 − BER in the case of binary outputs.

Assume that the outputs are binary ±1 instead of being discriminant values. Then we have the following situation for the histogram of output values:

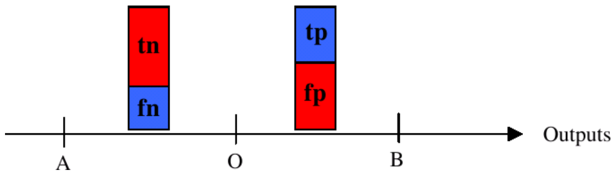


Figure B10: Histogram of output values

With at mid point the sensitivity and specificity given by $\text{Sen}_0 = \text{tp}/(\text{tp}+\text{fn})$ and the specificity given by $\text{Spe}_0 = \text{tn}/(\text{tn}+\text{fp})$. We have the ROC curve shown in Figure B11.

Therefore,

$$\begin{aligned}1-\text{AUC} &= \text{Spe}_0(1-\text{Sen}_0)/2+\text{Sen}_0(1-\text{Spe}_0)/2+(1-\text{Sen}_0)(1-\text{Spe}_0)\\ &= (1/2)(\text{Spe}_0-\text{Spe}_0\text{Sen}_0+\text{Sen}_0-\text{Spe}_0\text{Sen}_0+2-2\text{Spe}_0-2\text{Sen}_0+2\text{Spe}_0\text{Sen}_0)\\ &= 1-(\text{Spe}_0+\text{Sen}_0)/2\\ &= \text{BER}\end{aligned}$$

Since we have BER = 1 − BAC, we deduce that BAC = AUC.

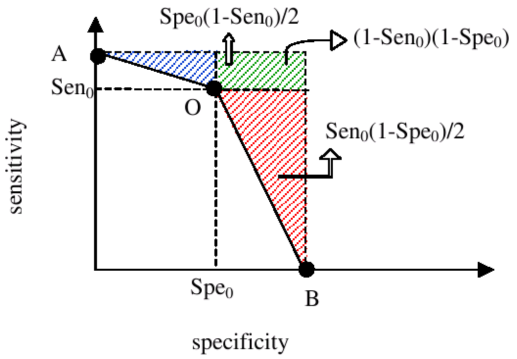


Figure B11: ROC curve

Appendix B4: Proof of PAUC = (n_+/n_r)AUC + 0.5n_-/n_r

From Equations (4) and (5), we have:

$$\text{AUC} = (1/\text{neg}) \sum_{k=1:\text{neg}} (r_k - k)/\text{pos}$$

which for the real variables gives:

$$n_+\text{AUC} = (1/n_-) \sum_{k=1:n_-} (r_k - k)$$

and for the combination of real variables and probes gives:

$$n_r\text{PAUC} = (1/n_p) \sum_{k=1:n_p} (r_k^{(\text{probe})} - k)$$

We assume that the negative examples and the probes are drawn from the same distribution. Consider the case where real variables and probes are intermixed. On average, half of the negative example fall before the mean value of the rank of the probe and half after. Hence, if we call r_k the rank of a probe if there were no negative examples (hence only positive examples and probes), we have on average over all possible drawings of negative examples and probes:

$$(1/n_p) \sum_{k=1:n_p} (r_k^{(\text{probe})}) = (1/n_p) \sum_{k=1:n_p} (r_k) = (1/n_p) + \text{neg}/2.$$

In the limit of infinite number of probes and negative examples we define the 2 following quantities:

$$A_r = \lim_{n_- \to \infty} (1/n_-) \sum_{k=1:n_-} (r_k - k)$$

$$A_p = \lim_{n_p \to \infty} (1/n_p) \sum_{k=1:n_p} (r_k^{(\text{probe})} - k) = \lim_{n_p \to \infty} (1/n_p) \sum_{k=1:n_p} (r_k - k) + \text{neg}/2$$

Thus $A_p = A_r + \text{neg}/2$

In the limit of infinite number of probes and negative examples

$$n_+\text{AUC} = A_r \text{ and } n_r\text{PAUC} = A_p$$

therefore

$$n_r\text{PAUC} = A_p = A_r + \text{neg}/2 = n_+\text{AUC} + \text{neg}/2$$

and

$$\text{PAUC} = (n_+/n_r)\text{AUC} + 0.5n_-/n_r \qquad €$$

We get a similar result with the BAC:

$$\text{BAC} = 0.5(\text{tp}/n_+ + \text{tn}/n_-) = 0.5(\text{sen} + \text{spe})$$
$$\text{PBAC} = 0.5((\text{tp} + \text{fp})/n_r + (n_p - n_{sp})/n_p)$$

Let us call SEN and SPE the expected value of the sensitivity and specificity. We have:

$$\text{SPE} = \lim_{n_-\to\infty} \text{tn}/n_- = \lim_{n_p\to\infty} (n_p - n_{sp})/n_p$$
$$\text{SEN} = \lim_{n_+\to\infty} \text{tp}/n_+$$
$$\lim_{n_p\to\infty} \text{PBAC} = 0.5((\text{tp} + \text{fp})/n_r + \text{SPE}$$
$$\lim_{n_p\to\infty} \lim_{n_-\to\infty} \text{PBAC} = 0.5((\text{SEN}n_+ + (n_- - \text{tn}))/n_r + \text{SPE}$$
$$\lim_{n_p} \lim_{n_-\to\infty} \lim_{n_+\to\infty} \text{PBAC} = 0.5((\text{SEN}n_+ + (1 - \text{SPE})n_-)/n_r + \text{SPE}$$
$$= 0.5(n_+/n_r)(\text{SEN} + \text{SPE}) + 0.5n_-/n_r$$
$$= (n_+/n_r) \lim_{n_-\to\infty} \lim_{n_+\to\infty} \text{BAC} + 0.5n_-/n_r$$

Appendix B5: The final algorithm

```
function [area, sigma] = auc2(Output, Target, pos_small)
%[area, sigma] = auc2(Output, Target, pos_small)
% This is the algorithm proposed for
% computing the AUC and the error bar.
% It is assumed that the outputs provide a score
% with the negative examples having the lowest score
% unless the flag pos_small = 1.

% Isabelle Guyon -- isabelle@clopinet.com -- November 2007

if nargin<3, pos_small=0; end
if ~pos_small, Output=-Output; end

negidx=find(Target<0);
posidx=find(Target>0);
neg=length(negidx);
pos=length(posidx);
[u,i]=sort(Output); % best come first
S(i)=1:(neg+pos);
SEN=(sort(S(negidx))-[1:neg])/pos;
SPE=1-(1:neg)/neg;
area=sum(SEN)/neg;

two_BAC=SEN+SPE;
[u,k]=max(two_BAC);
sen=SEN(k);
```

```
spe=SPE(k);
sigma= 0.5 * sqrt(sen*(1-sen)/ pos + spe*(1-spe)/ neg);
```

Appendix B6: Numerical simulations

```
% We verify the formula AUC = frac_pos PAUC + 0.5 frac_neg
% in the large sample size limit.

col='rgbkmc';
noise=0.01;
probe_num=2000;
real_num=2000;
N=probe_num+probe_num;
fp=\\;
repeat_num=500;
AReal=zeros(repeat_num, length(col));
AProbe=zeros(repeat_num, length(col));
EReal=zeros(repeat_num, length(col));
EProbe=zeros(repeat_num, length(col));
frac_pos=zeros(length(col),1);
frac_neg=zeros(length(col),1);
for j=1:length(col)
    frac_pos(j)=0.15*j;
    fp\j\=['FracPos=' num2str(frac_pos(j))];
    frac_neg(j)=1-frac_pos(j);
    pos_num=real_num*frac_pos(j);
    neg_num=real_num*frac_neg(j);

    Y=ones(N,1);
    probe_idx=1:probe_num;
    neg_idx=probe_num+1:probe_num+neg_num;
    pos_idx=probe_num+neg_num+1:N;
    real_idx=[neg_idx, pos_idx];
    Y(probe_idx)=-1;
    Y(neg_idx)=-1;

    Yprobe=ones(N,1);
    Yprobe(probe_idx)=-1;
    for k=1:repeat_num
        % Compute a fake discriminant value correlated with Y
        D=Y+0.5*randn(size(Y))+k*noise*randn(size(Y));
        Dreal=D(real_idx);
        Yreal=Y(real_idx);

        [Aprobe, Eprobe]=auc2(D, Yprobe);
        [Areal, Ereal]=auc2(Dreal, Yreal);
        AReal(k,j)=Areal;
        AProbe(k,j)=Aprobe;
        EReal(k,j)=Ereal;
        EProbe(k,j)=Eprobe;
    end
    % Linear fit
    w=[AReal(:,j), ones(size(AReal(:,j)))]\AProbe(:,j);
    Probe_hat=[AReal(:,j), ones(size(AReal(:,j)))]*w;
    fprintf('frac_pos=%g, frac_neg=%g, w=%g, 2*b=%g\n', frac_pos(j),
frac_neg(j), w(1), 2*w(2));
end

figure; hold on
for j=1:length(col)
    plot(AReal(:,j), frac_pos(j) * AReal(:,j) + 0.5 *
frac_neg(j),[col(j) '-'] );
end
legend(fp, 'Location', 'NorthWest');
for j=1:length(col)
    plot(AReal(:,j), AProbe(:,j), [col(j) '.']); xlabel('Real AUC');
ylabel('Probe AUC');
end
% Take at random a few points and draw the error box
Mini=min(min(AReal));
Maxi=1;
div=10;
vals=[Mini:(Maxi-Mini)/div:Maxi];
for j=1:length(col)
    for k=1:div+1
        [m,i]=min(abs(AReal(:,j)-vals(k)));
        xm=AReal(i,j)-EReal(i,j);
        xM=AReal(i,j)+EReal(i,j);
        ym=AProbe(i,j)-EProbe(i,j);
        yM=AProbe(i,j)+EProbe(i,j);
        plot([xm, xM, xM, xm, xm], [yM, yM, ym, ym, yM], [col(j) '-']);
    end
end

% Note: it is normal that empirical is larger then theoretical because
of
% the uncertainty on AReal
% Other method
Ediag=zeros(size(EProbe));
for j=1:length(col)
    % Compute the average distance to the line
    W1=frac_pos(j);
    W2=-1;
    N=sqrt(W1.^2+W2.^2);
    W1=W1./N;
    W2=W2./N;
    W0=0.5 * frac_neg(j)./N;
    sigma_emp=sqrt(mean((W1 * AReal(:,j) + W2 * AProbe(:,j) + W0).^2));
    sigma_th=mean(sqrt(EReal(:,j).^2+EProbe(:,j).^2));
    fprintf('Average empirical sigma=%5.4f, Average theoretical
sigma=%5.4f\n', sigma_emp, sigma_th);
    alpha=asin(frac_pos(j));
    Eboth(:,j)= sigma_th/cos(alpha); % Correct for the uncertainty of
real AUC
end
```

Appendix C: ChemTK QSAR descriptors used for SIDO

ChemTK generates a variety of descriptors or keys, which are used as features in the SIDO dataset. These include properties such as molecular weight, hydrogen-bond donor and acceptor counts, and rotatable bond counts, for a particular data set. Other keys include 2- and 3-point pharmacophore descriptors. We provide below *excerpts of the ChemTK manual to help decoding the key symbols.*

Smarts keys

ChemTK supports the Smarts query language developed by Daylight CIS, Inc. That company's website (`www.daylight.com`) provides an excellent tutorial for the Smarts language, so details of the syntax will not be provided here. Note that Boolean queries (e.g., "[c,n;!D3]") and recursive queries (e.g., "[$(C=O)]") are both supported.

In addition, users may specify named-property queries (see below) within a Smarts pattern using the angled bracket syntax (<>). For instance, the query "<HAcc>~C~<HAcc>" can be used to search for two hydrogen-bond acceptors connected via a single Carbon atom. The query names can correspond either to ChemTK defaults, or to user-defined queries as described below. Note that any named-property queries are assumed to describe single atoms; if multi-atom queries are used, only the information pertaining to the first atom will be retained.

Pharmacophore keys

ChemTK uses a type of pharmacophore that measures distance via bond connectivity rather than a typical three-dimensional distance. For instance, to describe a hydrogen-bond acceptor and hydrogen-bond donor separated by five connecting bonds, the corresponding key string would be "HAcc.HDon.5". More generally, a ChemTK pharmacophore contains two components: a list of features (hydrogen-bond acceptor, donor, etc.) and a list of pairwise distances measured in bond counts. Thus a 3-point pharmacophore has three features and three distances, while a 4-point pharmacophore has four features and six distances. In a key string used to represent a pharmacophore, all elements of the pharmacophore are separated by ".". Thus the following are examples of valid pharmacophore strings:
HAcc.HAcc.1
HDon.HDon.ExtArom.2.2.2
HAcc.HDon.Pos.ExtRing.2.5.2.3.1.2
The order in which the bond-based distances are listed in the above examples must correspond exactly to the order of the listed features, and should reflect the order of pairwise iteration. Hence in the third example, the six consecutive distances correspond to the following feature pairs: HAcc-HDon, HAcc-Pos, HAcc-ExtRing, HDon-Pos, HDon-ExtRing, Pos-ExtRing.

The following are included in the default feature list:

HAcc.	Hydrogen-bond acceptor.
HDon.	Hydrogen-bond donor.
Neg.	Explicit negative charge.
Pos.	Explicit positive charge.
ExtRing.	Ring atom having a neighbor atom external to the ring.
ExtArom.	Aromatic ring atom having a neighbor atom external to the ring.
ExtAliph.	Aliphatic ring atom having a neighbor atom external to the ring.

Ring System keys

This option yields a key set comprised of all unique ring systems contained in the set of input molecules. A ring system is defined as any number of single or fused rings connected by an unbroken chain of atoms. The simplest example would be either a single ring (e.g., benzene) or a single fused system (e.g., naphthalene). A more complex case would be these same two example systems (benzene and naphthalene) connected together by a chain of carbon atoms. Any connecting chains must be devoid of any terminating branches (such as a carbonyl group), such that all atoms in the final ring-system structure will always be connected to at least two other atoms. The user can place lower and upper bounds on the number of individual rings allowed in a ring system.

Keys are distinguished using atom type and aromaticity only, and are represented using a notation similar to Daylight Smarts. Thus a benzene ring would be represented using the notation "c1ccccc1". Counts of rings in a system are based on the typical SSSR definition, whereby, for example, benzene has one ring, naphthalene two, and a basic steroid scaffold four.

Unbranched Fragment keys

This option yields a key set comprised of all unique non-branching fragments contained in the set of input molecules. The user must specify a maximum and minimum size (in atoms) for all fragments. Keys are distinguished using atom type and aromaticity only, and are represented using a notation similar to Daylight Smarts. Thus, using the aniline molecule as an example, the two-atom keys are "cc" and "cN"; the three-atom keys are "ccc" and "ccN"; and so forth.

ChemTK may generate a special type of unbranched fragments called **isotopic fragments**. Keys of this type are annotated with special symbols that describe the precise ring topology of the fragment. For example a simple non-annotated key such as "cc" describes two aromatic carbon atoms connected by a single or aromatic bond. In contrast, the annotated key "[c;i1][c;i2]" describes a similar fragment, but also specifies a requirement that the first atom belong to a single ring (arbitrarily labeled 1) and that the second atom belong to a single ring **different** from the first. While the first key could equally match a single benzene ring, the juncture atoms of a naphthalene system, or the bridge atoms in a bi-phenyl structure, the latter key can match only the third example, since only in that case do the two atoms belong to single and distinct rings. Note that given a particular set of bounds on fragment size, the number of isotopic fragments is ordinarily far greater than the number of standard unbranched fragments, and the time required for key generation is correspondingly greater as well.

Branched Fragment keys

This option yields a key set comprised of all unique branched fragments contained in the set of input molecules. This method is intended to provide keys having a richer, more complex description than those available through the Ring System and Unbranched Fragment approaches. A detailed definition of branched fragment will not be provided in this Reference Guide. Briefly, each fragment is constructed through an "assembly" of shortest-path unbranched fragments, where each of the latter is required to be bounded by two atoms belonging to one or more pre-defined "terminal-atom" types selected by the user. The following options are available as terminal-atom types:

C.	Non-aromatic Carbon atom.
c.	Aromatic Carbon atom.
N.	Non-aromatic Nitrogen atom.

n.	Aromatic Nitrogen atom.
O.	Non-aromatic Oxygen atom.
o.	Aromatic Oxygen atom.
S.	Non-aromatic Sulfur atom.
s.	Aromatic Sulfur atom.
P.	Non-aromatic Phosphorus atom.
HAcc.	Hydrogen-bond acceptor.
HDon.	Hydrogen-bond donor.
ExtRing.	Ring atom having a neighbor atom external to the ring.
ExtArom.	Aromatic ring atom having a neighbor atom external to the ring.
ExtAliph.	Aliphatic ring atom having a neighbor atom external to the ring.

Some of these feature types are discussed in the section on **Pharmacophore** keys, as well as in the **Query Formats** section of this Reference Guide. In particular, see the discussion in the former section regarding **group features**, of which ExtRing, ExtArom, and ExtAliph are examples. The treatment of group features in the branched-fragment method is somewhat different from their treatment in the pharmacophore approach. Here, the requirement is that any branched fragment containing a group feature must contain no other feature that intersects the atoms described by the group feature. Thus, a branched-fragment key containing the ExtArom feature might validly describe a pyridine-containing molecule, but it is invalid for the features ExtArom and n (aromatic Nitrogen) to simultaneously describe the same pyridine ring: the ExtArom group feature excludes all other features from hitting the ring.

Branched-fragment keys are distinguished using atom type and aromaticity only, and are represented using a notation similar to Daylight Smarts. All feature atoms are designated using the angled brackets ("<>") notation. Thus a key using the features O, N and ExtArom might have a representation similar to the following: "[<O>]CC[<N>]CC[<ExtArom>]1ccccc1".

Named-property keys

Act.	Molecule activity. Note that if a user does not specify an activity field when an SD file is first opened, a value of zero is stored.
ArRing.	Aromatic ring atom. [**Group feature**].
ExtRing.	Ring atom having a neighbor atom external to the ring. [**Group feature**].
ExtArom.	Aromatic ring atom having a neighbor atom external to the ring. [**Group feature**].
ExtAliph.	Aliphatic ring atom having a neighbor atom external to the ring. [**Group feature**].
HAcc.	Hydrogen-bond acceptor atom.
HDon.	Hydrogen-bond donor atom.
MolWeight.	Molecular weight.
Neg.	Explicit negative charge.
Pos.	Explicit positive charge.
Ring.	Ring atom. [**Group feature**].

RotBond. Rotatable bond.

Generally, these names are used in conjunction with the "NPQ" tag to form an **encoded query** (discussed in a following section). For instance, the query "NPQ MolWeight < 500" can be used to search a document for all molecules having a molecular weight less than 500. Named-property queries can also be used within pharmacophore, branched-fragment, and Smarts queries, as described in those sections.

The named-property queries labeled as **group features** in the above list all describe more than a single atom. Such queries have special significance when used in the definitions of pharmacophore and branched-fragment keys. See those topics in the **Generating Keys** section for a description of how group features are handled in each case. Note that by using recursive Smarts syntax, a user can create a query that includes more than one atom in the definition but is not a group feature. For instance, the query "[$(c1ccccc1)]" cannot define a group feature since it is actually a single-atom query. In contrast, the similar query "c1ccccc1" is a six-atom query and therefore can define a group feature.

Users also have the option of defining new named-property queries, using the **Define Named Queries** option on the **Search** menu. When a new query is defined, the query name is stored internally along with the default names, and can be used in an identical fashion. ChemTK remembers the name until the application is closed, after which time the query must be re-loaded by the user. To define a new query, it is first necessary to create a special type of query file containing the query name and its definition. The **Create Query File** option on the **Search** menu contains an option to save a query file having the appropriate syntax. The following is an example of such a query file (containing two queries):

```
NAMEQUERY ZincBinder
SLQ ONC=O
$$$$
NAMEQUERY KeyScaffold
SLQ n1ccccc1
$$$$
```

In the first line of each query entry, the NAMEQUERY tag indicates that a new named-property query is being defined. The subsequent name (e.g., "ZincBinder") is the name by which the new query will be referenced, just as "MolWeight" is used to reference the molecular-weight query. The second line defines the query itself, and like all query-file records is in the form of an encoded query. Thus "SLQ" indicates a Smarts pattern, and (for example) "ONC=O" is the pattern for a hydroxamic acid group. See the subsequent discussion of encoded queries, and the section on **File Formats**, for additional information on these topics.

Once an appropriate query file has been generated, it is loaded using the **Load** button on the **Define named queries** dialog. At this point the new queries should appear in the window along with the defaults. Note that a user may elect to override these default named-property queries. For example, the name "HAcc" could be used in the above NAMEQUERY entry, in which case the new definition would take precedence over the ChemTK default.

Encoded queries

One of the principal query formats supported by ChemTK is the **encoded query**. This internal format provides users considerable flexibility in creating queries based on substructure, pharmacophore signature, and molecular property, with the additional option for Boolean logic and range specification (e.g., a molecular weight range). ChemTK supports the use of encoded queries in document searches and in key generation, and these queries serve as the format for individual records within query files. Each of these topics is covered in a relevant section of this Reference Guide.

A single encoded query is constructed by writing one or more encoded-query **primitives**, separated by **logical operators**. The format for an encoded-query primitive is:

QUERY_CODE QUERY_STRING SEARCH_CODE RANGE

The individual elements, which may be separated by any type of whitespace (excluding newline), are discussed below.

QUERY_CODE. This is a three-letter code that specifies the particular type of query that is being requested (e.g., a Smarts query). The following list provides the most important of the supported codes. A few additional codes, including **RSQ**, **SGQ** and **NULL**, are used internally and are not described.

NPQ. Specifies a named-property query. This code indicates that the subsequent query string will be a special name recognized by ChemTK as a synonym for a molecular property. Recognized names include MolWeight (molecular weight), Act (activity), HAcc (hydrogen-bond acceptor), as well as any names defined by the user. See the earlier discussion of named-property queries in this section of the Reference Guide for the complete list of supported query names and for instructions on defining customized named-property queries.

SLQ. Specifies a query based on the Smarts syntax defined by Daylight CIS, Inc. See `www.daylight.com` for a detailed tutorial on the Smarts language. Note that the SLQ code also permits specification of named-property queries, using the angled bracket syntax (<>). For instance, the string "<HAcc>~C~<HDon>" specifies a hydrogen-bond donor and acceptor group, separated by a single Carbon atom. See the discussion of Smarts queries in the **Query Formats** section for more detail.

PHQ. Specifies a pharmacophore query. An example query is "HAcc.HDon.2", which specifies a hydrogen-bond donor and acceptor, separated by two bonds. See the earlier sections (e.g., **Generating Keys**) for a discussion of the required syntax.

NAQ. Specifies a numeric-attribute query. This code indicates that the subsequent query string will be the name of a numeric molecule attribute previously loaded by the user. For instance, if a user has loaded a numeric attribute named "cLogP," he can search against this property using a syntax such as "NAQ cLogP > 0.5 < 3.5". Note that molecules not having the requested attribute (e.g., missing values) are treated as having a value of 0.0 when query results are derived. For more information regarding molecule attributes, see the section of this Reference Guide entitled **Loading Molecule Activities/Attributes**.

QUERY_STRING. This is the actual query, written in a format appropriate for the preceding query code. Hence "MolWeight" would be an appropriate query string for the "NPQ" code, while "c1ccccc1" (benzene) would be an appropriate query string for the "SLQ" code.

SEARCH_CODE. (Optional). One of three single-letter codes to indicate whether to perform a full match ("F"), single match ("S"), or unique match ("U"). For instance, if a benzene query is used to search a naphthalene molecule, a full match will return 24 hits, a unique match two hits, and a single match one hit. Note that if no code is provided, the default is to perform a full match.

RANGE. (Optional). A range specification applied to the match result. This range must take the form of either one or two symbol/value pairs, where the symbol is either ">", "<", or "=". For example, "= 3" indicates a result equal to 3, and "> 1 < 4" indicates a result that is greater than 1 and less than 4. The match-result value depends on the type of match requested. For example, "NPQ MolWeight" returns an actual weight value, while "SLQ c1ccccc1" returns the number of benzene matches. This convention is identical to that of key-value calculations; see the relevant discussion in the **Generating Keys** section for more detail.

The following are examples of valid encoded-query primitives:

```
NPQ MolWeight < 500
NPQ HAcc > 2 < 11
SLQ c1ccccc1 U = 2
SLQ <HDon>~<HAcc> S
PHQ HAcc.HAcc.HAcc.2.2.2
50
```

The first query specifies molecules having a molecular weight less than 500. The second query specifies molecules having between 3 and 10 hydrogen-bond acceptor groups. The third query specifies molecules having two distinct benzene rings. Note that here the "U" specifies a unique match, without which a large number of redundant matches would be returned. The fourth query specifies all molecules containing at least one example of a hydrogen-bond acceptor connected to a hydrogen-bond donor. The "S" ensures that at most a single match will be identified in each molecule. The fifth query specifies all molecules containing the particular pharmacophore.

Appendix D: Chemical Computing Group (CCG) QSAR descriptors

The CGC QSAR descriptors are partitioned into *classes*:

- **2D.** 2D descriptors only use the atoms and connection information of the molecule for the calculation. 3D coordinates and individual conformations are not considered.
- **i3D.** Internal 3D descriptors use 3D coordinate information about each molecule; however, they are invariant to rotations and translations of the conformation.
- **x3D.** External 3D descriptors also use 3D coordinate information but also require an absolute frame of reference (e.g., molecules docked into the same receptor).

Details on the CGC features are found at: http://www.chemcomp.com/.

Appendix E: Matlab code to filter MARTI data

```
function X=nreged_recover(Xold, data_dir, data_name)

cidx=load([data_dir '/' upper(data_name) '/' data_name '_feat.calib'
]);
cal0=load([data_dir '/' upper(data_name) '/' data_name '_feat.calval'
]);

view=0;
[p, n]=size(Xold);
X=zeros(p,n);
for k=1:p
    X(k,:)=nreged_filter(Xold(k,:), view, cidx, cal0);
end

function XF=nreged_filter(X, view, cidx, cal0)
% XF=nreged_filter(X)
% Filters a 2d pattern

if nargin<2, view=0; end
if nargin<3, cidx=[]; cal0=1; end

n=length(X);
t=sqrt(n);
val=sort(X);
% Compute background
background=median(val(1:50));
% Reshape as square
XP=reshape(X, t, t);
if view, cmat_display(XP); end

% add border
XB=zeros(size(XP)+6);
XB([1:t]+3, [1:t]+3)=XP;
XB([t-2:t]+6,[1:3])=(XP(t-1,1)+XP(t-2,2))/2 *ones(3);
XB([1:3],[1:3])=(XP(1,1)+XP(2,2))/2 *ones(3);
XB([1:3], [t-2:t]+6)=(XP(1, t-1)+XP(2, t-2))/2 *ones(3);
XB([t-2:t]+6,[t-2:t]+6)=(XP(t-1,t-1)+XP(t-2,t-2))/2 *ones(3);

XB([1:t]+3, 1:3)=(XP(:,[1 1 1])+XP(:,[2 2 2]))/2;
XB([1:t]+3, [t-2:t]+6)=(XP(:,[t-1, t-1, t-1])+XP(:,[t t t]))/2;
XB(1:3, [1:t]+3)=(XP([1 1 1],:)+XP([2 2 2],:))/2;
XB([t-2:t]+6, [1:t]+3)=(XP([t-1, t-1, t-1],:)+XP([t t t],:))/2;
if view, cmat_display(XB); end

% Remove the outliers
XBN=XB;
st=std(X);
tt=size(XB,1);
for i=1:tt
    for j=1:tt
        I=[i-1 i i+1];
        gidx=find(I>0 & I<tt);
        I=I(gidx);
        J=[j-1 j j+1];
        gidx=find(J>0 & J<tt);
        J=J(gidx);
        m=-XB(i,j);
        for ii=I
```

```
            for jj=J
               m=m+XB(ii,jj);
            end
        end
        m=m/9;
        if XB(i,j)>m+st | XB(i,j)<m-st
            XBN(i,j)=m;
        end
    end
end
if view, cmat_display(XBN); end

XB=XBN;

ker=[1 4 6 4 1]'*[1 4 6 4 1];
ker=ker./sum(sum(ker));
XBS=conv2(XB, ker, 'same');
XPS=XBS(4:t+3,4:t+3);
if view, cmat_display(XPS); end
if isempty(cidx)
    XC=XP-XPS+background;
else
    XC=XP-XPS;
end
if view, cmat_display(XC); end

XF=XC(:)';
if ~isempty(cidx)
    cal=mean(XF(cidx));
    XF=XF-cal+cal0;
end
```

Appendix III

Causal Explorer Software Library

Causal Explorer: A Matlab Library of Algorithms for Causal Discovery and Variable Selection for Classification

Alexander Statnikov[4], Ioannis Tsamardinos[1,2], Laura E. Brown[1], Constantin F. Aliferis[1,3,4]

[1] Department of Biomedical Informatics, Vanderbilt University, Nashville, TN, USA
[2] Department of Computer Science, University of Crete, Iraklio, Greece
[3] Department of Biostatistics, Vanderbilt University, Nashville, TN, USA
[4] Center of Health Informatics and Bioinformatics, New York University, New York, NY, USA

Abstract

Causal Explorer is a Matlab library of computational causal discovery and variable selection algorithms. *Causal Explorer* offers a wide variety of major prototypical and state-of-the-art algorithms in the field and a unified and easy-to-learn programming interface. *Causal Explorer* is designed for all researchers performing data analysis with the desire to gain an understanding in the underlying causal mechanisms that generated their data. In addition to the causal discovery methods, *Causal Explorer* contains related variable selection techniques. The variable selection algorithms in *Causal Explorer* are based on theories of causal discovery and the selected variables have specific causal interpretation. The *Causal Explorer* code emphasizes efficiency, scalability, and quality of discovery. The implementations of previously published algorithms included in *Causal Explorer* are more efficient than their original implementations. A unique advantage of *Causal Explorer* is the inclusion of very large scale and high quality algorithms developed by the authors of this chapter. The first version of *Causal Explorer* was introduced several years ago to the biomedical community. The purposes of this chapter are to re-introduce the library to a broader audience, to describe new functionality of the library, and to provide information on the use of *Causal Explorer* in community as a whole and in the Causation and Prediction Challenge.

1. Introduction

Discovery of causal knowledge is crucial for advancing research, developing new technology, and making sound policy, financial, and marketing decisions. Biologists need to know the factors that cause a disease to devise new therapeutic procedures. Public health policy makers need to know the factors that cause an increase in the number of medical errors in order to reduce them. Epidemiologists seek the factors causing disease in order to prevent it. Launching a new advertisement campaign requires knowing the factors that affect consumer behavior regarding the product. Increasing the number of visitors to a web site requires knowledge of what attracts them to the site.

Classically-trained statisticians often quote the maxim "association is not causation" to indicate that causal discovery is impossible without experiments. For example, simply observing a high occurrence of yellow stains on the fingers in patients with lung cancer relative to normal subjects does not imply a causal relation between cancer and staining (in reality heavy smoking is causing both to co-occur often). Similarly, observing that two items tend to be purchased together in high frequency does not necessarily imply that increasing the sales of the first item will be followed by an increase of the sales of the second item.

Unfortunately, discovering causal relations strictly by randomized experimentation is inefficient and often impractical, unethical, or simply impossible. Recent advances in computational causal discovery theory and algorithm research and development mathematically prove and experimentally show respectively the feasibility of causal discovery from observational data alone under broad conditions. The acceptance and application of causal discovery methods are steadily gaining ground. The following are just a few of important references in this emerging and exciting branch of science and technology: (Neapolitan, 2004; Spirtes et al., 2000; Pearl, 2000; Glymour and Cooper, 1999; Neapolitan, 1990; Pearl, 1988).

A large body of causal discovery algorithm research and development relies on the formalisms of graphical models such as Bayesian Networks (BNs) and Causal Probabilistic Networks (CPNs). BNs are computational and mathematical objects that represent compactly joint probability distributions by means of a directed acyclic graph denoting dependencies and independencies among variables and conditional probability distributions of each variable given its parents in the graph (Neapolitan, 1990). The fundamental axiom of BNs is the *Markov Condition* that allows for a concise factorization of the joint distribution and captures the main characteristic of causation in macroscopic systems, namely that causation is *local* (Glymour and Cooper, 1999). This leads naturally to Causal Probabilistic Networks (CPNs), i.e., a special class of Bayesian Networks (BNs) in which edges between any two variables in the graph denote direct causal relationships between the two variables (Spirtes et al., 2000). A review of applications of CPNs and BNs is outside the scope of this chapter; however we do note that CPNs and BNs although introduced a mere 20 years ago have already led to a long series of pioneering applications in various scientific disciplines (Neapolitan, 2009; Taroni, 2006; Popp and Yen, 2006; Gámez et al., 2004; Friedman et al., 2000; Heckerman et al., 1992; Heckerman and Nathwani, 1992).

Causal graphical models such as CPNs are also recognized in bioinformatics and computational biology, as important representations for modeling causal relationships at a finer granularity than standard clustering or regression methods, and as having sound statistical foundations for handling noise, missing data and doing inference (Neapolitan, 2009; Baldi and Hatfield, 2002). The appeal of CPNs is that, contrary to the heuristic approaches for generation of causal hypotheses in bioinformatics and medical research, (e.g., methods that were based on clustering, regression, and variable selection as in (Li et al., 2001; Eisen et al., 1998; Spellman et al., 1998)) the recently-developed theory of *causal induction* using graphical models and related distributions, provides guarantees for highly sensitive and specific discovery of causal relationships (Spirtes et al., 2000). For example, it has been theoretically proven that such methods can be used to reliably infer causal relationships among variables in: distributions captured by acyclic graphs (Spirtes et al., 2000); continuous linear Gaussian systems with feedback loops in equilibria (Spirtes et al., 2000); dynamic systems outside equilibrium sampled at discrete time points (Friedman et al., 1998); and linear or non-linear systems of discrete variables in equilibria (Pearl and Dechter, 1996).

It has also been shown that under certain broad conditions, a Markov blanket which is the minimal set of predictors needed for the classification of a response variable of interest is the set of direct causes, direct effects, and direct causes of the direct effects of the response variable in a CPN (Tsamardinos and Aliferis, 2003). Thus, causal discovery algorithms that find the Markov blanket by necessity solve the variable selection problem.

We have recently introduced to the biomedical audience the powerful technology of causal discovery and variable selection encapsulated in the *Causal Explorer* library (Aliferis et al., 2003b). Over the years, we have added more algorithms and new functionality to the library. The purposes of this chapter are to re-introduce *Causal Explorer* to a broader audience, to describe new functionality of the library, and to provide information on the use of *Causal Explorer* in community as a whole and in the Causation and Prediction Challenge. In addition, we wish to stimulate research with the set of causal discovery and variable selection algorithms that we have developed for datasets with very large numbers of variables (Aliferis et al., 2009a,;

Tsamardinos et al., 2006a; Brown et al., 2005,; Tsamardinos and Aliferis, 2003; Aliferis et al., 2003a; Tsamardinos et al., 2003,; Aliferis et al., 2002).

2. The *Causal Explorer* Library

Currently a rich variety of software is available for modeling and inference with BNs but only a limited amount of commercial and public domain software for learning causal graph models from data is available to researchers (for a comprehensive collection of software tools see: `http://www.ai.mit.edu/~murphyk/Software/bnsoft.html`).

Causal graph induction algorithms come in three flavors: Bayesian (or search-and-score) approaches, constraint-based conditional independence approaches, and hybrid approaches. When a researcher is interested in a specific region of the causal graph (e.g., to find causes and effects of the response variable or to find a pathway), there is no need to induce the entire causal graph (i.e., perform "*global causal discovery*"), instead one can induce that specific region of interest (i.e., perform "*local causal discovery*") which is typically much more computationally efficient (Aliferis et al., 2009a,; Chickering et al., 1994). In our experience, local causal discovery methods can be applied to datasets with hundreds of thousands variables where global causal discovery methods may not be practical. Also, the so-called Markov blanket induction methods (which is a sub-family of local causal discovery techniques) provably solve the variable selection problem under the assumptions about the learner and loss function (Tsamardinos and Aliferis, 2003).

We describe here a software library (which we call *Causal Explorer*) that provides researchers with code that can be used for causal discovery (global and local) and variable selection. *Causal Explorer* can be used primarily to:

1. Discover the direct causal or probabilistic relations around a response variable of interest (e.g., disease is directly caused by and directly causes a set of variables/observed quantities).

2. Discover the set of all direct causal or probabilistic relations among the variables.

3. Discover the Markov blanket of a response variable of interest, i.e., the minimal subset of variables that contains all necessary information to optimally predict the response variable.

The selection of algorithms in *Causal Explorer* (see next section) emphasizes highly-scalable causal discovery, reliable and fast implementations and convenient integration to custom code. Such algorithms have been frequently employed in analysis of data in psychology, medicine, biology, weather forecasting, animal breeding, agriculture, financial modeling, information retrieval, natural language processing, and other fields. They can be used to automatically construct decision support systems from data (e.g., for medical diagnosis), or to generate plausible causal hypotheses (e.g., which gene regulates which).

The *Causal Explorer* library is provided as a collection of Matlab functions. The reasons for this choice are fourfold: (a) Matlab is a versatile and wide-spread environment for experimentation with data mining and modeling tasks; (b) Matlab codes can be interfaced with practically any standard language such as C++, Java, etc. (c) As newer versions of the contained algorithms are being developed, transfer to the library can be made very quickly (e.g., compared to the much slower process of re-writing the new algorithms in C/C++); (d) Matlab code if written correctly (i.e., in "vectorized" form) is very efficient and in our experiments it often outperforms native implementations of the algorithms written in C/C++ and other languages.

The *Causal Explorer* library is provided free of charge for non-commercial research. Code, example data, and documentation are available online at: `http://www.dsl-lab.org/causal_explorer`.

3. Causal Discovery and Variable Selection Algorithms

In this section, we describe the algorithms implemented in *Causal Explorer*. Most constraint-based algorithms currently support three statistical tests of independence (or measures of association depending on context): G^2 and thresholded mutual information for multinomial distributions and Fisher's Z-test for multivariate Gaussian distributions (Anderson, 2003; Cover et al., 1991). In most cases this extends the functionality of the algorithms from their original published form. We also note that the algorithms HITON-PC, HITON-MB, MMHC, MMPC, and MMMB were not included in the first version of *Causal Explorer* (Aliferis et al., 2003b). The detailed information on running algorithms, their inputs, and outputs can be found in the user's manual that is included in the installation package of *Causal Explorer*.

3.1 PC

PC is a prototypical global causal discovery constraint-based algorithm with well-developed theory and many applications (Spirtes et al., 2000). The *Causal Explorer* implementation of PC does not impose limits on the number of variables or cases in the input, and is conveniently callable from other code via the provided API.

3.2 TPDA (Three Phase Dependency Analysis)

TPDA is also a global causal discovery algorithm that achieves polynomial-time execution if a constraint on the distribution of variables is enforced (Cheng et al., 2002). The *Causal Explorer* implementation of TPDA employs a very fast implementation of mutual information and does not restrict the number of input variables or cases unlike the version distributed by the TPDA inventors in BN PowerConstructor software (`http://www.cs.ualberta.ca/~jcheng/bnpc.htm`). It is also easily callable from other code.

3.3 SCA (Sparse Candidate Algorithm)

This is a fast search-and-score global causal discovery algorithm designed for sparsely connected domains, e.g., gene networks (Friedman et al., 1999).

3.4 MMHC (Max-Min Hill Climbing)

MMHC is a highly scalable hybrid global causal discovery algorithm that has been shown to outperform in speed and quality several state-of-the-art algorithms including techniques mentioned above (Tsamardinos et al., 2006a). MMHC first uses a local discovery algorithm MMPC to learn a skeleton of the network and then it uses search-and-score method for its orientation.

3.5 KS (Koller-Sahami)

The Koller-Sahami algorithm returns a heuristic approximation to the Markov blanket of the response variable (Koller and Sahami, 1996). A very fast implementation of expected cross entropy is used in the algorithm implementation.

3.6 LCD2

The LCD2 algorithm is a local causal discovery algorithm that requires knowledge of one or more instrumental variables (i.e., variables that have no parents within the studied set of variables) (Cooper, 1997).

3.7 GS (Grow-Shrink)

The Grow-Shrink algorithm returns the Markov blanket of a variable (Margaritis and Thrun, 1999). In multinomial distributions, this algorithm requires sample size exponential to the number of variables in the Markov blanket.

3.8–3.11 IAMB (Incremental Association Markov Blanket), IAMBnPC, InterIAMB, interIAMBnPC

These are algorithms that return the Markov blanket of a variable (Tsamardinos and Aliferis, 2003; Tsamardinos et al., 2003). *IAMBnPC*, *InterIAMB*, *interIAMBnPC* either use the PC algorithm (Spirtes et al., 2000) or interleaved pruning to reduce the number of returned false positives relative to IAMB (trading off sample for speed) (Tsamardinos et al., 2003). In multinomial distributions, all these algorithms require sample size exponential to the number of variables in the Markov blanket.

3.12–3.13 HITON-PC (HITON Parents and Children) and MMPC (Max-Min Parents and Children)

HITON-PC and MMPC are local causal discovery algorithms that return the set of direct causes and effects of the response variable (Aliferis et al., 2009a,; Tsamardinos et al., 2006a; Aliferis et al., 2003a; Tsamardinos et al., 2003b). HITON-PC uses univariate heuristic for prioritization of variables, while MMPC uses max-min association heuristic. These are highly sample efficient discovery techniques.

3.14–3.15 HITON-MB (HITON Markov Blanket) and MMPC (Max-Min Markov Blanket)

These are Markov blanket induction algorithms that require much less sample compared to GS and IAMB family of Markov blanket inducers (Aliferis et al., 2009a,; Tsamardinos et al., 2006a; Aliferis et al., 2003a; Tsamardinos et al., 2003b). HITON-MB uses univariate heuristic for prioritization of variables, while MMMB uses max-min association heuristic.

4. Other Tools

In addition to the causal discovery algorithms mentioned in section 3, *Causal Explorer* also includes several tools that facilitate causal discovery experiments and development of new algorithms. These tools are outlined below. None of these tools were provided in the first version of *Causal Explorer* (Aliferis et al., 2003b). The detailed information on running these tools can be found in the user's manual that is included in the installation package of *Causal Explorer*.

4.1 Bayesian network tiling tool

It is well recognized in the field that the major technique for evaluating and comparing causal discovery algorithms is by simulation of data from a network of known structure. Then, it is easy to compare the reconstructed network as learnt by an algorithm with the true data-generating network to assess the quality of learning. For the results of the evaluations to carry to real-world data distributions the networks used for data simulations have to be representative of the real-world examples. Typically, the networks employed for the data simulation are extracted from real-world BN-based decision support systems. Unfortunately, the size of the existing known BNs is relatively small in the order of at most a few hundred variables. Thus, typically causal discovery algorithms were so far validated on relatively small networks (e.g., with less than 100 variables), such as the classical ALARM network (Beinlich et al., 1989) or other "toy-networks". Algorithms have also been developed to generate large random BNs. The *BNGenerator* system is one example for generating large random BNs from a uniform distribution (Ide and Cozman, 2002). However, the *BNGenerator* system and other algorithms of this type do not provide any guarantees that these networks resemble the networks of the distributions likely to be encountered in practice (Aliferis and Cooper, 1994). The emergence of datasets of very high-dimensionality poses significant challenges to the development of new causal discovery algorithms.

To address this problem, *Causal Explorer* implements an algorithm for generating arbitrarily large discrete Bayesian networks by tiling smaller real-world known networks. The algorithm preserves the structural and probabilistic properties of the tiles so that the distribution of the resulting tiled network resembles the real-world distribution of the original tiles (Tsamardinos et al., 2006b).

4.2 Bayesian network data simulator

Causal Explorer implements a procedure to simulate data from Bayesian networks using the Gibbs sampling algorithm (Russell and Norvig, 2003). Such data can be used for evaluation of existing causal discovery algorithms and development of new methods.

4.3 Utility for supervised discretization of continuous data

In order to discretize continuous data, *Causal Explorer* implements a supervised discretization method that works as follows:

1. Data is normalized so that each variable has mean 0 and standard deviation 1.

2. After normalization, association of each variable with the response variable is computed using either Wilcoxon rank sum test (for binary response variable) or Kruskal-Wallis non-parametric ANOVA (for multicategory response variable) at 0.05 alpha level (Hollander and Wolfe, 1999).

3. If a variable is not significantly associated with the response variable, it is discretized a follows:

   - *0* for values less than -1 standard deviation
   - *1* for values between -1 and 1 standard deviation
   - *2* for values greater than 1 standard deviation

4. If a variable is significantly associated with the response variable, it is discretized using sliding threshold (into binary) or using sliding window (into ternary). The discretization threshold(s) is determined by the Chi-squared test to maximize association with the response variable (Agresti, 2002).

The discretization procedure can be instructed to compute necessary statistics only using training samples of the data to ensure unbiased estimation of error metrics on the testing data.

5. General Guidelines and Context of Use

The algorithms in *Causal Explorer* can be used in several different experimental tasks and contexts: (a) to gain insight in the causal structure of the studied domain; (b) to locate promising variables for subsequent experimentation or detailed modeling; and (c) to derive a provably optimal minimal set of predictors for classification purposes.

In general, global causal discovery algorithms PC, TPDA, and SCA can be practically run when the number of variables is up to a few hundred and the connectivity (i.e., number of direct causes/effects around variables) of the data-generating process is uniformly small. When the number of variables is of the order of thousands, MMHC algorithm will be most helpful as it is the most scalable method.

Local causal discovery algorithms will be most helpful when the number of variables is very large, or when the connectivity around the response variable is small (relative to available sample) while around other variables it may be large.

In particular, when the sample is large relative to the size of the Markov blanket of the response variable (as a rule of thumb when several hundred samples are available for a Markov blanket with ~ 5 variables), GS and the IAMB variants will return excellent results. When the sample is smaller, HITON-MB and MMMB should be applied instead. Also, in many cases algorithms HITON-PC and MMHC that return the set of direct causes and effects of the response variable can be used for approximation of the Markov blanket.

6. Statistics of Registered Users

At the time of writing this chapter, *Causal Explorer* has 739 registered users in more than 50 countries all over the world. Based on provided information in the user registration form, 402 (54%) users are affiliated with educational, governmental, and non-profit organizations and 337 (46%) users are either from private or commercial sectors. Major commercial organizations that have registered users of *Causal Explorer* include IBM, Intel, SAS Institute, Texas Instruments, Siemens, GlaxoSmithKline, Merck, and Microsoft. Table 1 provides a list of major U.S. institutions that have registered users of *Causal Explorer*.

7. Use of *Causal Explorer* in the Causation and Prediction Challenge

Causal Explorer library was used both by participants and organizers of the Causation and Prediction Challenge.

7.1 Use of *Causal Explorer* by the Challenge participants

Here are major achievements enabled by the *Causal Explorer* library in the Causation and Prediction Challenge:

1. Gavin Cawley used *Causal Explorer* to become one of the Challenge winners. The software library allowed him to achieve the best prediction accuracy on SIDO and MARTI datasets (p11).

2. Jianxin Yin et al. used *Causal Explorer* to become the Challenge winners in the "*best overall contribution*" category. Specifically, they obtained the best position of the Pareto front of the Fscore vs. Tscore graph over all datasets (pp11–12).

3. Laura Brown and Ioannis Tsamardinos used *Causal Explorer* to achieve the top overall ranking on REGED dataset (p31).

Table 1: A list of major U.S. institutions that have registered users of *Causal Explorer*.

1.	Boston University	30.	University of California Berkley
2.	Brandies University	31.	University of California Los Angeles
3.	Carnegie Mellon University	32.	University of California San Diego
4.	Case Western Reserve University	33.	University of California Santa Cruz
5.	Central Washington University	34.	University of Cincinnati
6.	College of William and Mary	35.	University of Colorado Denver
7.	Cornell University	36.	University of Delaware
8.	Duke University	37.	University of Houston-Clear Lake
9.	Harvard University	38.	University of Idaho
10.	Illinois Institute of Technology	39.	University of Illinois at Chicago
11.	Indiana University-Purdue University Indianapolis	40.	University of Illinois at Urbana-Champaign
12.	Johns Hopkins University	41.	University of Kansas
13.	Louisiana State University	42.	University of Maryland Baltimore County
14.	M. D. Anderson Cancer Center	43.	University of Massachusetts Amherst
15.	Massachusetts Institute of Technology	44.	University of Michigan
16.	Medical College of Wisconsin	45.	University of New Mexico
17.	Michigan State University	46.	University of Pennsylvania
18.	Naval Postgraduate School	47.	University of Pittsburgh
19.	New York University	48.	University of Rochester
20.	Northeastern University	49.	University of Tennessee Chattanooga
21.	Northwestern University	50.	University of Texas at Austin
22.	Oregon State University	51.	University of Utah
23.	Pennsylvania State University	52.	University of Virginia
24.	Princeton University	53.	University of Washington
25.	Rutgers University	54.	University of Wisconsin-Madison
26.	Stanford University	55.	University of Wisconsin-Milwaukee
27.	State University of New York	56.	Vanderbilt University
28.	Tufts University	57.	Virginia Tech
29.	University of Arkansas	58.	Yale University

Summary of the use of *Causal Explorer* by the Challenge participants is provided in Table 2.

Table 2: Summary of the use of *Causal Explorer* by the Challenge participants.

Participant team	**Algorithms used in *Causal Explorer***	**Challenge ranking on Tscore**				**Reference**
		REGED	**SIDO**	**CINA**	**MARTI**	
Gavin Cawley	• HITON-MB, • HITON-PC, • MMHC	2	1	3	1	p97
Jianxin Yin et al.	• MMPC, • Supervised discretization	3	5	4	2	p85, p172
Cristian Grozea	• Markov blanket algorithm (details are not provided)	7	12	7	6	p142
H. Jair Escalante and Luis Enrique	• HITON-PC	6	8	9	5	p144
Ernest Mwebaze and John Quinn	• HITON-PC	9	7	8	—	p153
Marius Popescu	• HITON-MB, • TPDA	5	—	—	—	p161
Wu Zhili	• HITON-MB, • HITON-PC	13	13	14	11	p175
Laura Brown and Ioannis Tsamardinos	• MMPC, • MMMB • MMHC, • HITON-MB	*Excluded from the Challenge ranking due to conflict of interest*				p31,p126

7.2 Use of *Causal Explorer* by the Challenge organizers

The *Causal Explorer* library was also used by the Challenge organizers. First, resimulation of REGED dataset (p179) employed HITON-PC algorithm (as a part of HITON-Bach method) that was implemented in *Causal Explorer*. In addition, HITON-PC and MMHC algorithm implementations from *Causal Explorer* were used as the baseline methods to gain insight into the problem difficulty (p179). At the time of Challenge the use baselines was not disclosed to the participants.

8. Discussion

CPNs and other causal graphical models are powerful mathematical formalisms that are useful for variable selection, dimensionality reduction, causal hypothesis generation, and automatic creation of predictive/classification tools and decision support systems. Unfortunately the complexity of most related algorithms prevents many researchers from employing them in experiments since proper implementation often requires extensive familiarity with the theory and a substantial investment of resources for proper coding and testing. In addition, the existing code in the public domain typically comes in stand-alone executable form and may contain hard-coded limitations on input data size.

The first contribution of the present chapter is that it re-introduces the *Causal Explorer* library to a broader audience and describes new functionality compared to the previous version of the library (Aliferis et al., 2003b). The second contribution is that the library makes available to the research community a suite of algorithms designed by authors of this chapter for coping efficiently and reliably with thousands of variables. These algorithms have been recently tested with a variety of datasets with excellent results (Aliferis et al., 2009a,), however at this stage the potential of these methods is practically untapped. Finally, we also describe the use of *Causal Explorer* in community at a whole and in the context of Causation and Prediction Challenge. It is our hope that the *Causal Explorer* library will stimulate interest in, and experimentation with this important class of mathematical and computational tools by the broader research community.

9. Acknowledgements

This work supported by NIH grants R56 LM007948-04A1, R01 LM007948-01, P20 LM007613-01, and Vanderbilt AVCF for the Discovery Systems Laboratory.

References

Agresti, A. (2002) *Categorical data analysis.* Wiley-Interscience, New York, NY, USA.

Aliferis, C.F. and Cooper, G.F. (1994) An evaluation of an algorithm for inductive learning of Bayesian belief networks using simulated data sets. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI).*

Aliferis, C.F. et al. (2009a) Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification. Part I: Algorithms and Empirical Evaluation. (In press) *Journal of Machine Learning Research.*

Aliferis,C.F. et al. (2009b) Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification. Part II: Analysis and Extensions. (In press) *Journal of Machine Learning Research.*

Aliferis, C.F., Tsamardinos, I. and Statnikov, A. (2002) Large-scale feature selection using Markov blanket induction for the prediction of protein-drug binding. *Technical Report DSL 02–06.*

Aliferis, C.F., Tsamardinos, I. and Statnikov, A. (2003a) HITON: a novel Markov blanket algorithm for optimal variable selection. *AMIA 2003 Annual Symposium Proceedings*, 21–25.

Aliferis, C.F. et al. (2003b) *Causal Explorer*: a causal probabilistic network learning toolkit for biomedical discovery. *Proceedings of the 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS).*

Anderson, T.W. (2003) *An introduction to multivariate statistical analysis.* Wiley-Interscience, Hoboken, N.J.

Baldi, P. and Hatfield, G.W. (2002) *DNA microarrays and gene expression.* Cambridge University Press, Cambridge, UK.

Beinlich, I. et al. (1989) The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence in Medicine.*

Brown, L.E., Tsamardinos, I. and Aliferis, C.F. (2004) A novel algorithm for scalable and accurate Bayesian network learning. *Medinfo 2004*, 11, 711–715.

Brown, L.E., Tsamardinos, I. and Aliferis, C.F. (2005) A comparison of novel and state-of-the-art polynomial Bayesian network learning algorithms. *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI).*

Cheng, J. et al. (2002) Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137, 43–90.

Chickering, D.M., Geiger, D. and Heckerman, D. (1994) Learning Bayesian networks is NP-hard. *Technical Report MSR-TR-94-17.*

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

Cover, T.M. et al. (1991) *Elements of information theory.* Wiley New York.

Eisen, M.B. et al. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. U. S. A*, 95, 14863–14868.

Friedman, N. et al. (2000) Using Bayesian networks to analyze expression data. *J Comput. Biol.*, 7, 601–620.

Friedman, N., Murphy, K. and Russell, S. (1998) Learning the structure of dynamic probabilistic networks. pp. 139–147.

Friedman, N., Nachman, I. and Pe'er, D. (1999) Learning Bayesian network structure from massive datasets: the "Sparse Candidate" algorithm. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI).*

Gámez, J., Moral, S. and Salmerón, A. (2004) *Advances in Bayesian networks.* Springer, Berlin.

Glymour, C.N. and Cooper, G.F. (1999) *Computation, causation, and discovery.* AAAI Press, Menlo Park, Calif.

Heckerman, D.E., Horvitz, E.J. and Nathwani, B.N. (1992) Toward normative expert systems: Part I. The Pathfinder project. *Methods Inf. Med*, 31, 90–105.

Heckerman, D.E. and Nathwani, B.N. (1992) Toward normative expert systems: Part II. Probability-based representations for efficient knowledge acquisition and inference. *Methods Inf. Med*, 31, 106–116.

Hollander, M. and Wolfe, D. (1999) *Nonparametric statistical methods.* Wiley, New York, NY, USA.

Ide, J.S. and Cozman, F.G. (2002) Random generation of Bayesian networks. *Lecture Notes in Computer Science*, 366–375.

Koller, D. and Sahami, M. (1996) Toward optimal feature selection. *Proceedings of the International Conference on Machine Learning*, 1996.

Li, L. et al. (2001) Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics*, 17, 1131–1142.

Margaritis, D. and Thrun, S. (1999) Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems*, 12, 505–511.

Neapolitan, R.E. (1990) *Probabilistic reasoning in expert systems: theory and algorithms.* Wiley, New York.

Neapolitan, R.E. (2004) *Learning Bayesian networks.* Pearson Prentice Hall, Upper Saddle River, NJ.

Neapolitan, R.E. (2009) *Probabilistic methods for bionformatics (with an introduction to Bayesian networks).* Morgan Kaufmann Publishers, Burlington, MA.

Pearl, J. (1988) *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers, San Mateo, California.

Pearl, J. (2000) *Causality: models, reasoning, and inference.* Cambridge University Press, Cambridge, U.K.

Pearl, J. and Dechter, R. (1996) Identifying independencies in causal graphs with feedback. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI)*, 420–426.

Popp, R.L. and Yen, J. (2006) *Emergent information technologies and enabling policies for counter-terrorism.* Wiley-Interscience, Hoboken, N.J.

Russell, S.J. and Norvig, P. (2003) *Artificial intelligence: a modern approach.* Prentice Hall/Pearson Education, Upper Saddle River, N.J.

Spellman, P.T. et al. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol Cell*, 9, 3273–3297.

Spirtes, P., Glymour, C.N. and Scheines, R. (2000) *Causation, prediction, and search.* MIT Press, Cambridge, Mass.

Taroni, F. (2006) *Bayesian networks and probabilistic inference in forensic science.* Wiley, Chichester, England.

Tsamardinos, I. and Aliferis, C.F. (2003) Towards principled feature selection: relevancy, filters and wrappers. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AI & Stats).*

Tsamardinos, I., Aliferis, C.F. and Statnikov, A. (2003a) Algorithms for large scale Markov blanket discovery. *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 376–381.

Tsamardinos, I., Aliferis, C.F. and Statnikov, A. (2003b) Time and sample efficient discovery of Markov blankets and direct causal relations. *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD)*, 673–678.

Tsamardinos, I., Brown, L.E. and Aliferis, C.F. (2006a) The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65, 31–78.

Tsamardinos, I. et al. (2006b) Generating Realistic Large Bayesian Networks by Tiling. *Proceedings of the 19th International Florida Artificial Intelligence Research Society (FLAIRS) Conference.*