Predictive Conditional Independence Testing

with applications in graphical model structure learning

Samuel Burkart

Supervisor : Dr. Franz Király

MSc Data Science
Department of Statistical Science
University College London

August 31, 2017

Abstract

Multivariate conditional independence testing and graphical model structure learning are complex problems, and current best practices for both field lack general, simple algorithms to address the tasks. These problems include the need for heuristic or subjective manual choices when applying the test, high computational complexity and a general lack of models that are feasible for the general case. This paper aims to address some of these problems, by deriving a link between classical statistical independence testing and predictive modelling, to overcome these issues in current state-of-the-art methodology. Additionally, it allows for any advances in the highly researched predictive modelling workflow to be automatically leveraged into more powerful conditional independence tests. By implementing the test as a wrapper for the popular scikit-learn package, the derived routines are simple and easy-to-use. It is shown, that the proposed algorithm for conditional independence testing fares well against current best practices, and the newly proposed exact graphical model structure learning algorithm, which runs in quadratic time in the size of the graph, asymptotically recovers the true graph, while keeping the variance of the outputs low. This paper, and the 'pcit' package accompanying it, thus provide powerful, scalable and easy-to-use methods for conditional independence testing and graphical model structure learning.

Acknowledgements

I would like to thank my supervisor, Dr. Franz Király, for his support in creating this paper. The proofs for the statements in Section 3 are largely based on personal communication and an unpublished draft manuscript of his.

Contents

List of Figures

List of Tables

1 Introduction

The study of dependence is at the heart of any type of statistical analysis, and independence testing is an important step in many scientific step, be it to determine if two things are related, or to assess if an intervention had the desired effect. There are many different approaches to independence testing, kernel-based methods, density estimation and copulas, to name a few. However most are constrained to specific cases and struggle with multivariate or conditional tests. One field where independence testing is especially important, is the area of graphical models, where one tries to encode a set of independence statements arising from a probability distribution into a graph. Like independence testing, structure learning algorithms are themselves subject to complex problems, with the most important one being combinatorial explosion in the size of the graph. State-of-the-art methods avoid exponential run time through strong assumptions on distributions, and as such impose a potentially strong bias on the learning procedure, just to make the methodology feasible.

The present work addresses some of these issues through the introduction of a novel multivariate conditional independence testing routine, which overcomes problems related to manual and subjective modelling choices and provides a highly efficient, scalable and powerful algorithm that works well for multivariate and conditional independence testing. Additionally, an undirected graph structure learning routine that runs in quadratic time and provides exact results (without making any strong assumptions on the distribution) based on the new conditional independence routine is proposed.

Section 2 shows a review of the current state-of-the-art methods for independence testing. The proposed predictive conditional independence test (PCIT), which is derived in Section 3, links the classical task of independence testing to the supervised learning workflow to overcome these current issues in independence testing methodology and to directly benefit from advances in that field. The sophisticated understanding of parameter tuning for predictive modelling algorithms directly benefits the proposed independence routine and overcomes the need for most manual tuning and heuristic choices prevalent in current methodology.

The PCIT is then applied in a graphical model structure learning context, where one tries to capture the independence structure of a distribution giving

rise to a data set. Section 4 introduces the concept of graphical models and surveys the different approaches to graphical model structure learning. The drawbacks of current methodology, such as the need for strong assumptions on distributions that gave rise to data sets, or, alternatively, the taxing computational complexity of exact approaches, are outlined.
Some of these issues are tackled by a newly proposed algorithm to estimate the undirected skeleton of a graph, outlined in Section 5. The algorithm is exact and has a computational complexity of $\mathrm{O}(p^2)$, where p is the number of variables in the model. Additionally, it is shown how the PCIT can be used to trade off between computational complexity and power of the algorithm, allowing to adapt the structure learning routine easily to the task at hand.
An implementation of the conditional independence routine and the structure learning algorithm can be found on `https://github.com/SamBurkart/PCIT` and is distributed under the name 'pcit' in the Python Package Index (PyPI). Section 6 gives on overview of the package API, important subroutines and Section 7 describes how the package draws it's power from the Python package scikit-learn.
Section 8 then compares the performance of the PCIT against current best practices and shows that, in some scenarios, a considerable improvement can be achieved by switching to the PCIT. Then, performance statistics for different settings of the structure learning algorithm are produced, indicating that asymptotically the true graph is recovered, while the number of false discoveries can be controlled at a predefined rate, and can be traded off against power of the algorithm. Lastly, consistency under resamples is assessed and some example outputs on real world data sets are shown.
Section 9 concludes the paper and proposes directions for future research.

2 Statistical independence testing

Statistical independence is a property attained by random variables, which declares that the state of one variable X is independent of the state of the other variable Y, usually denoted as $X \perp\!\!\!\perp Y$.

Definition 1. *Random variables X and Y taking values in $\mathcal{X}$ and $\mathcal{Y}$ are marginally independent if $\forall X^* \subseteq \mathcal{X}$ and $\forall Y^* \subseteq \mathcal{Y}$*

$$P(X^* \cap Y^*) = P(X^*)P(Y^*),$$

assuming that the probabilities P are well defined, where $P(X^*)$ is the probability of X taking a value in X^*

This formulation allows for X and Y to be defined over sets of random variables that are a mixture of continuous and discrete, as well as being univariate or multivariate.

Definition 2. *Random variables X and Y taking values in $\mathcal{X}$, $\mathcal{Y}$ are conditionally independent given Z taking values in $\mathcal{Z}$ if $\forall X^*, Y^*, z^*$ such that $X^* \subseteq \mathcal{X}$, $Y^* \subseteq \mathcal{Y}$ and $z^* \in \mathcal{Z}$*

$$P(X^* \cap Y^* | z^*) = P(X^* | z^*)P(Y^* | z^*),$$

assuming that the probabilities P are well defined, where $P(X^* | z^*)$ is the probability of X taking a value in X^* when z^* is the observed (known) state of Z

For continuous X and Y, independence implies that the joint distribution equals the product of the marginal distributions. Thus, if the goal is to reason about X, Y does not bear any additional information about X if we already know Z. Independence is symmetric, if the equations in Definition 1 or 2 are true, they are also true for X and Y interchanged.

This notion of statistical independence has a variety of applications, such as:

- When conducting market research, one might be interested in questions such as “Are our advertisement expenditures independent of our profits?” (hopefully not), or the more sophisticated version “conditional on the state of the market, are advertisement expenditures independent of our profits?”, which, if found to be true, would mean we are unlikely to increase profits through an increase in our advertising budget (subject to the usual issues with inferring causality from data).
- When collecting data for a medical study on the occurrence of an outcome Y, one might ask “In the presence of data about *attributes A* for the subjects, should we still collect data for *attributes B*”. If Y is independent of *attributes B* given *attributes A*, additionally collecting information about *attributes B* will not improve the knowledge of the state of Y.

The task of independence testing (in various settings) has been tackled from many different angles. A few of the most prominent methodologies, which will be discussed in detail in sections 2.1 to 2.3, are:

- **Density estimation**: The classical approach. The multivariate probability density function contains all the necessary information about independence structures in a set of random variables.
- **Copulas**: An approach that is widely used in finance and risk management. Copulas are multivariate probability distributions with uniform marginals. To use copulas for independence tests, one transforms the marginals of a multivariate distribution into uniform distributions, and the resulting copula contains all information about the independence structure. An example using the empirical cdf can be found in Genest and Rémillard [2004].
- **Kernels**: A relatively recent approach using optimization for manually chosen kernels in reproducing kernel Hilbert spaces to answer independence queries based on various test-statistics.

All these approaches have been applied to various independence testing tasks, and have their own strengths and weaknesses. The following sections will give an overview of the current state of the art methods for marginal and conditional independence testing, as well as of two-sample tests which evaluate the probability that two samples stem from the same underlying probability distribution. Subsequently, an overview of the drawbacks/constraints of the methods is given.

2.1 Marginal independence testing

This section will provide a brief overview on the methods that explicitly test for independence between random variables X and Y, based on limited samples from the underlying distributions, however independence testing and two-sample testing are strongly related, as will be shown in Section 2.3. This paper will mainly focus on multivariate testing, since there already exist powerful tests for the univariate case. Classical measures for univariate statistical independence between random variables are given by the Pearson correlation coefficient, Kendall's τ and Spearman's ρ. For discrete variables, Pearson's χ^2 provides an additional statistic. These test statistics, paired with their respective null-hypothesis and null-distributions, provide simple and powerful measures for univariate independence test. A more sophisticated test based on the squared difference in the joint distribution and the product of the two marginal distributions can be found in Hoeffding [1948].
Multivariate independence tests have been tackled through the use of copulas [Schweizer and Wolff, 1981] and density estimation-based information-theoretical measures [Dionisio et al., 2006]. Recent work focused on more powerful kernel-based methods that tend to perform better in higher-dimensional settings by avoiding this need for density estimation. In 2005, Arthur Gretton introduced a new measure for dependence, the Hilbert-Schmidt Independence Criterion (HSIC), which is defined as the squared Hilbert Schmidt norm of the cross-covariance operator (the covariance between two difference spaces), details can be found in Section 2 of Gretton et al. [2005]. Gretton et al. [2008] further expends on the theoretical foundations of the HSIC's distribution under the null, resulting in a hypothesis test for the independence of two variables X and Y using the HSIC criterion. In practice, the method can suffer from computational complexity (squared in the sample size) and the manual choice of

hyperparameters, as well as it's prohibitive mathematical complexity, making it hard for average users to perform tasks that require deviating from default values.

2.2 Conditional independence testing

Testing for conditional independence is an inherently more difficult problem than marginal independence [Bergsma, 2004]. Hence, few viable options exist to date to test if two sets of variables are conditionally independent given a conditioning set. While tests that are based on binning the (continuous) data and comparing $P(X|Z)$ to $P(X|Y, Z)$, for example, are easily extendable to the conditional setting, it is obvious that the estimation of conditional probability densities demand for much larger sample sizes to arrive at the respective confidence that could have been attained in a marginal independence test. Zhang et al. [2012] circumvent this density estimation problem by using a kernel-based method to estimate if X and Y are conditionally independent given a set Z. The test statistic is based on conditional cross-covariance operators, for which they work out the asymptotic distribution under the null-hypothesis $X \perp\!\!\!\perp Y|Z$. Like other kernel-based methods however, it requires the manual tuning of a set of hyperparameters or reliance on heuristics, with potentially strong effects on type 1 and type 2 errors, especially when the dimensionality of the conditioning set is large. Additionally, they found that the performance of their method decreases in the dimensions of the conditioning set, additionally constraining the set of problems for which the test is viable for.
In the copula realm, the task of testing for conditional independence has been attempted through the use of partial copulas [Bergsma, 2011]. As for other copula-based approaches, the methods suffers from the need for manually choosing an appropriate copula. Additionally, strong assumptions are made on the type of relationship between the variables that are to be tested for independence and the conditioning set (it is assumed to be linear).

2.3 Two sample tests

Two-sample testing, that is, testing the null-hypothesis that two samples stem from the same underlying distribution, is another classical statistical problem. It has many applications, such as in clinical trials (testing a sample taken from a treatment group against one taken from a control group) or in data integration, where one would like to increase the sample size by merging samples from two difference sources, and thus needs to test if the two samples stem from the same underlying process. In a well-defined sense, two-sample testing is a generalization of independence testing, since the test if X and Y are independent is equivalent to testing if $P(X \cap Y) = P(X)P(Y)$. Equivalently, to conduct a two-sample test with the null $X_1 \overset{d}{=} X_2$, where $\overset{d}{=}$ indicates equality in underlying distribution, one might alternatively construct a test assessing if the joint set $X \equiv \{X_1, X_2\}$ is independent from the vector Y containing information if an instance in X stems from X_1 or X_2.

Univariate

There exist many different powerful univariate two-sample tests for all types of data. They mainly divide into parametric (where the data is assumed to stem from a certain distribution) and non-parametric (”distribution-free”) tests. Stronger assumptions result in more powerful tests (if the assumptions are true). For the parametric case, the t-test in its various forms is commonly used, whereas the Wilcoxon tests are usually implemented in the non-parametric case. These tests usually come with a version for tied samples for equal sample sizes (where it is assumed that observations of the same index in both samples are in some sense related, e.g. as a result of being calculated on the same data set), which generally results in a more powerful test than for unpaired tests (since, again, stronger assumptions lead to more powerful tests). Two of these univariate tests are implemented in Section 3.3.

Multivariate

Multivariate two-sample testing is a much more complex problem, both with respect to data requirements and computational complexity, and many different attempts at deriving a feasible test for multivariate samples have been made.

Baringhaus and Franz [2004] derive a test based on the euclidian interpoint distances between the two samples and derives the asymptotic distribution under the null. The complexity of calculating the test statistic is $\mathrm{O}(p \times q \times d)$, where p and q are the number of instances in the samples and d is the dimensionality of the data, and it thus scales badly for large data sets. A different attempt at the same test statistics with the additional use of characteristic functions was made by Fernández et al. [2008]. While the difficulty of density estimation is reduced when using the empirical distributions, the data requirements tend to be much larger, while additionally imposing a sometimes taxing computational complexity. Both of these methods can be related to kernel functions, which was picked up by Gretton et al. [2012a], who proposed kernel-based two-sample test by deriving three multivariate tests for assessing the null-hypothesis that the distributions p and q generating two samples $X \sim p$, $Y \sim q$ are equal, $p = q$. The criterion introduced by them is the "Maximum Mean Discrepancy" (MMD) between p and q over a function space $\mathcal{F}$. That is, the maximum difference between the expected function values with respect to the function space (which they choose to be a Reproducing Kernel Hilbert Space).

$$MMD = \sup_{f \in \mathcal{F}} (\mathbb{E}_{X \sim p}[f(X)] - \mathbb{E}_{Y \sim q}[f(Y)])$$

The algorithm runs in quadratic time, O((m+n)^2), however they also propose an approximation for a large sample size that runs in linear time, O(m+n). The main issue, as with other kernel-based approaches to independence testing, is the heuristics driven/manual approach to hyperparameter-tuning as well as the potentially prohibitive cost of deriving the test statistic and it's asymptotic distribution. Heuristics for the problem of optimal kernel choice for the MMD criterion have been outlined in Gretton et al. [2012b]. More recently, Fourier transform-based sampling approaches to two-sample tests with linear time complexity in the number of data, relying on mean embeddings (whose distance in the RKHS can shown to be equivalent to the MMD criterion) and analytical representations of probability measures, have been developed by Chwialkowski et al. [2015].

In the copula realm, two sample testing has largely remain unaddressed, since it's application in the financial sector are less relevant. Rémillard and Scaillet [2009] describe a two sample test for the estimated copulas, and hence, the independence structures, but the test does not extend to a comparison of the joint distributions as a whole.

Some of the problems with two-sample testing have been addressed by a novel approach based on binary classifiers introduced by Lopez-Paz and Oquab [2016]. Again, for 2 sets of samples X and Y, $X \sim p$, $Y \sim q$, the null-hypothesis p=q is assessed. The idea is, to assign each data point in X the label 0, and each data point in Y the label 1, and to then train a classifier (classification function) that can predict if an instance is a 1 or a 0. If the null-hypothesis about the samples being drawn from the same distribution is true, this classifier should not fare significantly better than chance on an unseen test set. They define the test statistic t to be the fraction of correct classification, which is expected to behave like the average of independent Bernoulli($\frac{1}{2}$) random variable under the null. While the method is pretty straightforward, it might suffer from a lack of power for small sample sizes. Additionally, knowledge about paired data cannot be leveraged in a straightforward manner to derive a more powerful test.

2.4 Shortcomings of the existing methods

The shortcomings of the state-of-the-art methods across different fields vary, but are largely consistent within the different methodologies:

- **Density estimation**: As aforementioned, while the probability density function is in some sense the optimal for measuring dependence, since it contains all the information about the random variable, it's estimation is a difficult (and due to the curse of dimensionality for more than 3 dimensions generally even intractable) task, which requires either strong assumptions are large amounts of data (or both).

- **Copulas**: Leaving aside issues made by practitioners misunderstanding the method (which had a strong contribution to the 2007/2008 financial crisis), copula-based independence testing is a heuristics driven field, requiring many subjective manual choices for the estimation. Above all, copula methods requires a user to subjectively choose an appropriate copula from a variety of options, such as the survival copula, the multivariate Gaussian and the multivariate Student's t copula [Cherubini et al., 2004]. Additionally, two sample testing is to date largely unaddressed in the field.

- **Kernels**: Like for the copula-based methods, kernels require many subjective and manual choices, such as the choice of kernel function, and its hyperparameters. While in theoretical setting this can be addressed by hyperparameter tuning through the usage of the *ground truth*, artificial data, in practice it is difficult to tune the hyperparameters or make statements about the confidence in results.

3 Predictive independence testing

This section will explore the relationship between predictability and dependence in a set of variables. It will be shown that one can use supervised learning methodology to conduct marginal and conditional independence tests. This distinction between marginal and conditional dependence is not theoretically necessary (since the marginal case can be achieved by setting the conditioning set to the empty set), but is made to highlight specific properties of the two approaches. First, equivalence of independence statements and a specific supervised learning scenario will be shown. After, a routine leveraging this equivalence to test for conditional independence will be proposed. The theoretical setup is adapted from Fránz Kiraly (personal communication and unpublished draft manuscript) and as follows

Mathematical setting

The following tests will be based on a supervised learning routine. In supervised learning, the task is to find a function f of a variable X, s.t. $f(X)$ well approximates a target variable Y, where "well" is defined with respect to a loss function L (see figure 1) which is to be minimized in the expectation. That is, one seeks an f s.t.

$$f = \arg\min_g \mathbb{E}[L(g(X), Y)],$$

where $\mathbb{E}[L(g(X), Y)]$ is the expected generalization loss of g under the convex loss function L. The distribution of X and Y is unknown, but one has access to samples of X and Y that are assumed to be independent and identically distributed, $(X_i, Y_i) \sim (X, Y)$, for $i \in \{1, ..., N\}$. To test for over-fitting (choosing an f that performs well on the data set that it is estimated on, but badly for unseen data), one splits up the data into a training set, where f is trained on, and a test set, where the performance of f is measured on. In the following sections, when estimating the generalization error, it will be assumed that f is independent of the test data set.

A loss function is a function $L : Y \times Y \to \mathbb{R}$, that assigns a value to each possible configuration of prediction $\hat{Y} = f(X)$ and outcome Y. In Supervised Learning, the task is usually to find a function f that minimizes the expected generalization error $\epsilon(f) = \mathbb{E}(L(f(X), Y))$. To avoid having to estimate the joint density over X and Y, one usually considers the empirical risk $\hat{\epsilon}(f)$ for a prediction functional f on a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where the joint distribution over $\{X, Y\}$ is approximated by the empirical distribution observed in the training/test data, $\hat{\epsilon}(f) = \frac{1}{n}\sum_{i=1}^n L(f(x_i), y_i)$.

Figure 1: Loss function in supervised learning, empirical risk

3.1 Marginal independence

Let $P \in \mathcal{P}$ be a probability distribution over Y and $T : P \to \mathbb{R}^q$ be a functional that assigns a set of values to each P. Examples of $T(P)$ could be the mean or quantiles of Y.

Definition 3 (Gneiting [2011]). *A loss function $L(y, y^*)$ elicits a functional T if, $\forall t \in T(P)$,*

$$\mathbb{E}[L(t, Y)] < \mathbb{E}[L(x, Y)],\ \forall x \in \mathbb{R}^q \setminus T(P).$$

Following the examples above, the mean of a function is elicitable (it is elicited by the squared loss), under some regularity conditions on $\mathcal{P}$ (finite second moment).
Now consider the univariate quantile loss function L_α,

$$L_\alpha(y, y^*) = (\mathbb{I}(y \geq y^*) - \alpha)(y - y^*). \tag{1}$$

Example 1. *For continuous Y, the α–quantile of the (predictive) distribution $p(y)$ of Y is elicited by L_α.*

For example, $L_{0.5}$ reduces to the absolute loss, which is minimized by the median (the 0.5-quantile), a known property of the absolute loss.

Proof. Let $P(x) = \int_{-\infty}^{x} p(y)dy$, where $p(y) = \frac{\partial P(y)}{\partial y}$.

$$\begin{aligned}
\mathbb{E}_Y[L_\alpha(y^*, Y)] &= \int_{-\infty}^{y^*}(1-\alpha)(y^*-y)p(y)dy - \int_{y^*}^{\infty}(\alpha)(y^*-y)p(y)dy \\
&= y^*(P(y^*)-\alpha) + \alpha\mathbb{E}[Y] - \int_{-\infty}^{y^*} yp(y)dy \\
\frac{\partial \mathbb{E}_Y[L_\alpha(y^*, Y)]}{\partial y^*} &= y^*p(y^*) + P(y^*) - \alpha - y^*p(y^*) = P(y^*) - \alpha \overset{!}{=} 0 \\
&\implies P(y^*) = \alpha \\
\frac{\partial^2 \mathbb{E}_Y[L_\alpha(y^*, Y)]}{\partial (y^*)^2} &= p(y^*) \geq 0
\end{aligned}$$

Hence, the first order condition is a minimum, minimized by the α-quantile of Y, and, by Definition 3, the quantile loss elicits the quantile. □

Theorem 1. *Suppose that X and Y are sets of random variables taking values in $\mathcal{X}$ and $\mathcal{Y}$, Y following some probability distribution P. Then, for all convex loss functions L that elicit a functional T_L, and a set $\mu_L \in T_L(P)$ (the optimal point forecast),*

$$X \perp\!\!\!\perp Y \iff [x \to \mu_L] = \arg\min_f \mathbb{E}_Y[L(f(X), Y)],\ (\forall \mu_L \in T_L(P)).$$

That is, if X and Y are independent, the prediction strategy f minimizing the expected generalization loss is given by the function that predicts any $\mu_L \in T_L(P)$, for some (often times singleton) set $T_L(P)$. Since the choice of μ_L has no impact on the below results, an arbitrary representative of $T_L(P)$ is chosen. In a supervised learning context, μ_L will be referred to as the *uninformed baseline* prediction strategy, since we are testing if adding a variable X improves the prediction.

Proof. "$\implies$"

$$\begin{aligned}
\mathbb{E}_{X|Y}[L(f(X), Y)|Y] &\geq L(\mathbb{E}_{X|Y}[f(X)|Y], Y) && \text{(by Jensen's inequality)} \\
&= L(\mathbb{E}_X[f(X)], Y) && \text{(since } X \perp\!\!\!\perp Y\text{)} \\
&= L(c, Y) && \text{(defining } c := E[f(X)]\text{)} \\
&\geq L(\mu_L, Y) && \text{(by definition 3)}
\end{aligned}$$

By law of total expectation, taking expectations over Y on both sides gives

$$\mathbb{E}_{X,Y}[L(f(X),Y)] \geq \mathbb{E}_Y[L(\mu_L,Y)] = \mathbb{E}_{X,Y}[L(\mu_L,Y)] \quad \Box$$

$$\text{“} \Longleftarrow \text{”}$$

Now assume that

$$[x \to \mu_L] = \arg\min_f \mathbb{E}_{XY}[L_\alpha(f(X),Y)](\forall\alpha).$$

By Example 1, the optimal constant predictor is μ_L s.t. $P(\mu_L) = \alpha$. Thus, the quantiles/level curves of Y are independent of X, and hence the marginal probability density function of Y and its distribution conditional on X are equivalent,

$$p(x,y) \equiv p(y|x)p(x) = p(y)p(x).$$

which, by Definition 2 implies independence between X and Y.

Similarly for discrete Y, the optimal μ_L is given by the estimated probabilities for the occurrence of a class (Appendix A). That means that the marginal probability mass function (pmf) of Y is equivalent to the pmf conditional on X, leading to the same conclusion as in the continuous case. $\Box$

These results hold only for univariate Y, and thus the marginal quantiles, which is sufficient for the methodology outlined in Section 5.1. Additionally, since the null-hypothesis is that $X \perp\!\!\!\perp Y$, failing to detect multivariate dependence (a specific case) only decreases the power of the test, the type 1-error remains unaffected.

3.2 Conditional independence

The last section established a theoretical foundation for marginal independence testing, now this foundation will be expanded by adding a framework for testing conditional independence.

The statement one would like to make thus connects 2 random variables, X and Y, that are conditionally independent given a third variable, Z, and the expected generalization loss from predicting from Z and from the set $\{X, Z\}$.

Theorem 2. *Let X and Y be as defined in Theorem 1, and Z be a random variable taking values in $\mathcal{Z}$. Then, for all convex loss functions L and $\forall g(X,Z)$,*

$$X \perp\!\!\!\perp Y|Z \iff \mathbb{E}[L(f(Z),Y)] \leq \mathbb{E}[L(g(X,Z),Y)].$$

where $f(Z)$ is the function predicting $T_L(\hat{P})$, with $\hat{P}$ being the conditional distribution of $[Y|Z]$ and T_L being the functional elicited by L

Proof. " $\implies$ "

$$\begin{aligned}\mathbb{E}_{X|Y,Z}[L(g(X,Z),Y)|Y,Z] \geq& L(\mathbb{E}_{X|Z}[g(X,Z)|Z],Y) \quad \text{(Jensen's inequality)}\\ \equiv& L(\tilde{f}(Z),Y) \qquad (\tilde{f}(Z) := \mathbb{E}_{X|Z}[g(X,Z)|Z])\end{aligned}$$

By the law of total expectation, taking expectations over Y and Z on both sides, and the fact that $\mathbb{E}[\epsilon(\hat{f})] \geq E[\epsilon(f)]$,

$$\mathbb{E}_{X,Y,Z}[L(g(X,Z),Y)] \geq \mathbb{E}_{Y,Z}[L(f(Z),Y)] = \mathbb{E}_{X,Y,Z}[L(f(Z),Y)].$$

Since g is arbitrary, for all $g(X,Z)$, there exists an $\tilde{f}(Z)$ that leads to a smaller loss. Since $f(Z)$ is the function with the lowest expected generalization error among all prediction functionals predicting Y from Z, its expected generalization error is thus smaller or equal than the respective error for all g and corresponding $\tilde{f}$ □

" $\impliedby$ "

For a continuous random variable Y, consider the quantile loss L_α and

$$\mu_f(z) := \arg\min_{y \in \mathcal{Y}} \mathbb{E}[L_\alpha(y, Y|Z=z)]$$

$$\mu_g(x,z) := \arg\min_{y \in \mathcal{Y}} \mathbb{E}[L_\alpha(y, Y|Z=z, X=x)]$$

Which represent the optimal predictions for $[Y|Z]$ and $[Y|Z,X]$ w.r.t the quantile loss L_α.

By Example 1, these are given by the α-quantiles of the conditional distribution. Since the quantiles are elicited by the quantile loss,

$$\mu_f(Z) = \arg\min_f \mathbb{E}(L_\alpha(f(Z), Y))$$

and

$$\mu_g(X, Z) = \arg\min_f \mathbb{E}(L_\alpha(g(X, Z), Y)).$$

Since μ_f is just a special case of μ_g (conditioning on an additional variable can not decrease the information about the quantiles of Y), μ_g should perform at least as well as μ_f,

$$\mathbb{E}_{Y,Z}[L(\mu_f(Z), Y)] \geq \mathbb{E}_{X,Y,Z}[L(\mu_g(X, Z), Y)].$$

From the assumption that the RHS of Theorem 1 is true,

$$\mathbb{E}_{Y,Z}[L(\mu_f(Z), Y)] \leq \mathbb{E}_{X,Y,Z}[L(\mu_g(X, Z), Y)],$$

follows that the expected quantile loss is equivalent for μ_f and μ_g, implying that, by Example 1, all quantiles of $[Y|Z]$ and $[Y|Z, X]$ are equivalent, since the quantile loss is strictly convex. Hence, $p(Y|X, Z) = p(Y|Z)$ and

$$p(X, Y|Z) = p(Y|X, Z)p(X|Z) = p(Y|Z)p(X|Z),$$

which, by Definition 2 is sufficient to attest that $X \perp\!\!\!\perp Y|Z$.

For discrete Y, μ_f and μ_g are given by the functions predicting the class probabilities conditional on information on Z and $\{X, Z\}$ respectively, which minimize the expected generalization loss under the logistic loss. Conditioning on another variables can not decrease the information about the class probabilities in the conditional probability mass function and hence, by the same logic as for the continuous case, μ_g and μ_f are equivalent. As a result, $p(Y|X, Z) = p(Y|Z)$, which again leads to $X \perp\!\!\!\perp Y|Z$, by factorization and Definition 2. $\square$

The logic used here, attesting that a low assigned probability to the statement "Prediction strategy f has a lower expected generalization error than baseline strategy g" implies dependence between X and Y, conditional on Z, draws from the same type of statistical syllogism as classical statistical hypothesis testing.

Figure 2: Probabilistic inference for predictability and dependence

The same comments about the decrease of power by focusing on the marginal quantiles as for Theorem 1 apply. Unlike in Theorem 1, the right-hand-side of theorem 2 cannot easily be evaluated for a given g, since f is generally intractable, due to the complexity of estimating functionals of conditional distribution functions. Thus, in the algorithm in Section 5.1, the statement will be replaced by a slightly weaker statement, where automatic model selection will determine the optimal $\hat{f}$ as well as $\hat{g}$, without guarantees that the found $\hat{f}$ is the globally optimal f. This is in line with a general task of supervised prediction theory, which is to replace expensive or even intractable functionals with tractable approximations.

3.3 Testing prediction error against baseline

The last section established a theoretical basis for relating statistical independence to predictability of a set of variables from another set of variables, by establishing a baseline loss for the null-hypothesis of conditional independence and a method to determine if the null can be rejected. The following section will expand this by introducing a statistical inference procedure to test if equality between the LHS and the RHS in equation 2 can be rejected for a (potentially uninformed) baseline functional f and an optimal prediction functional g. This concept of comparing outputs from arbitrary prediction functionals has recently increased in relevance as many machine learning algorithms make fewer distributional assumptions than classical statistical methodology, and hence many estimates are produced without associated standard errors, a problem addressed in Nadeau [2003], who proposed the use of a two-sample t-test.

By considering a contraposition (see Figure 2 for a discourse on this) of equation 1, one can derive a strategy to test for dependence between X and Y. Namely, if there exists a prediction strategy g, such that

$$\mathbb{E}_Y[L(g(X), Y)] < \mathbb{E}_Y[L(\mu_L, Y)], \tag{2}$$

where $g(x) \neq \mu_L$ and μ_L is the best constant predictor, then X and Y are not independent. Two remarks:

- To get an estimate of the generalization error, it is important the estimation of g was performed on a data set that is independent from the data set on which the generalization error is estimated. If not, and as a result of the function g being much more flexible than the baseline, overfitting will occur and the best baseline predictor might be unjustifiably discarded in favor of the more flexible predictor g.

- Randomness in the data calls for a significance test for Equation 2. A parametric and a non-parametric possibility for such a test will be explored below.

First, the expectations of the losses in Equation 2 will be replaced by the empirical risk for a sample $\{X_i, Y_i\}_{i=1}^N$, a prediction function g and a loss function L. That is

$$\mathbb{E}_Y[L(Y, g(X))] \equiv \epsilon(g) \approx \frac{1}{N}\sum_{i=1}^N L(g(X_i), Y_i) \equiv \frac{1}{N}\sum_{i=1}^N L_i(g).$$

In order to assess if a prediction function g is significantly better than another prediction function$f$, one has to test if $\frac{1}{N}\sum_{i=1}^N L_i(g) \equiv \bar{L}(g)$ is significantly lower than $\frac{1}{N}\sum_{i=1}^N L_i(f) \equiv \bar{L}(f)$. From the mathematical setup, we know that the test set is independent of the training set, conditional on the prediction functional, and hence the loss residuals are independent, and identically distributed, and by the central limit theorem,

$$\sqrt{N}(\bar{L}(g) - \epsilon(g)) \xrightarrow{d} \mathcal{N}(0, \text{Var}[L(g)]),\ N \to \infty.$$

That is, the empirical mean of the loss residuals for prediction strategy g and loss L is asymptotically normal with mean $\epsilon(g)$ and variance Var$[L(g)]$ / N.

The delta method can be used to calculate an estimate of the standard deviation of the mean of any differentiable monotonous loss function L, here we will provide an estimate under the squared loss.

Let L be the squared loss, that is, $L(x, y) = (x - y)^2$, with mean expected generalization loss $\epsilon \equiv \mathbb{E}[L(x, y)]$ and estimator on an observed data set $\mathcal{D}$,

$$\hat{\epsilon}_{\mathcal{D}} = \frac{1}{N}\sum_{i=1}^{N} L_i \equiv \frac{1}{N}\sum_{i=1}^{N} L(x_i, y_i),$$

where $(x_i, y_i) \in \mathcal{D}$, $i = \{1, ..., n\}$ are observed. We will denote $\hat{\epsilon}_{\mathcal{D}}$ as $\hat{\epsilon}$. Further, define the estimated standard error,

$$\hat{\sigma}_\epsilon = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(L_i - \hat{\epsilon})^2}.$$

Theorem 3. *The RMSE $\sqrt{\epsilon}$ is asymptotically normal with mean $\sqrt{\hat{\epsilon}}$ and standard error $\hat{\sigma}_\epsilon/(2\sqrt{\hat{\epsilon}})$*

Proof. The mean $\sqrt{\hat{\epsilon}}$ follows from the asymptotic distribution of the MSE and the continuous mapping theorem. For $\sqrt{\hat{\epsilon}}$ close to $\sqrt{\epsilon}$, an approximation to $\sqrt{\epsilon}$ is given by the first order Taylor series.

$$\sqrt{\epsilon} \approx \sqrt{\hat{\epsilon}} - \frac{(\epsilon - \hat{\epsilon})}{2\sqrt{\hat{\epsilon}}}$$

$$\text{Var}(\sqrt{\epsilon}) \approx \frac{1}{4}\frac{\text{Var}(\epsilon)}{\hat{\epsilon}}$$

$$\implies \hat{\sigma}_{\sqrt{\epsilon}} \equiv \sqrt{\text{Var}(\sqrt{\epsilon})} \approx \frac{\hat{\sigma}_\epsilon}{2\sqrt{\hat{\epsilon}}}$$

Again considering the Taylor series approximation, for $X_n \sim N(\mu, \frac{\sigma^2}{n})$,

$$\sqrt{X_n} = \sqrt{\mu} + \frac{X_n - \mu}{4\mu^{1/2}} - \frac{(X_n - \mu)^2}{8\mu^{3/2}} + ...$$

From the CLT and the fact that X is an average, we know that, for $n \to \infty$,

$$(X_n - \mu) \sim N(0, \frac{\sigma^2}{n}).$$

Hence, the asymptotic distribution of $(X - \mu)$ is the same as $Y\frac{\sigma}{\sqrt{n}}$, where Y is standard normal, so for $n \to \infty$,

$$\sqrt{X_n} - \sqrt{\mu} \sim \frac{1}{4\mu^{1/2}} Y \frac{\sigma}{\sqrt{n}} - \frac{1}{8\mu^{3/2}} (Y * \frac{\sigma}{\sqrt{n}})^2 + ...$$

Clearly, the terms after the first one converge to 0 faster as $n \to \infty$

$$\lim_{n \to \infty} (\sqrt{X}_n - \sqrt{\mu})\sqrt{n} \sim \mathcal{N}(0, \sigma^{*2})$$

setting $\sigma^* = \frac{\sigma}{4\sqrt{\mu}}$. Hence, since the mean squared error is normal as defined by X, the asymptotic distribution of it's square root, the root mean squared error, is also normal. $\square$

However, since the split into training and test set will be the same for each of the prediction functions that are to be compared, the loss residuals for the prediction functions are inherently tied, and a more powerful procedure can be achieved by considering two-sample tests.
For two prediction strategies f and g, define the difference in the loss residual i, $\delta_i = L_i(g) - L_i(f)$, where $L_i(g) = L(g(x_i), y_i)$ is the loss of prediction functional g on instance i. In general, the hypothesis we are interested in, is the one sided test for the difference δ, $\delta_i \sim \delta$, $\forall i$

$$H_0 : \mathbb{E}(\delta) = 0 \text{ against } H_A : \mathbb{E}(\delta) < 0,$$

where without loss of generality we assume that we test if the incumbent prediction function g can outperform the baseline f, since a low δ_i is associated with larger loss residual for $L_i(f)$ than for $L_i(g)$. The summary for δ in H_0, here the mean, can differ, we might also be interested in the median of δ, or similar.

To assess this null-hypothesis, two simple tests, one parametric and one non-parametric, are implemented.

- **T-test for paired samples [Nadeau, 2003]**: A parametric test assuming that the differences between the loss residuals δ_i are normally distributed with mean 0 and standard deviation σ. Under the null-hypothesis, the normalized empirical mean

  $$\frac{\frac{1}{N}\sum_{i=1}^{N}\delta_i}{\frac{\sigma}{\sqrt{N}}},$$

  follows a t-distribution with degrees of freedom N, where we have used the fact that under H_0, $\delta = 0$.

- **Wilcoxon signed-rank test**: If the δ_i are not normally distributed, an increase in power can be achieved by testing if the rank differences are symmetric around the median. Since the δ_i in question are differences in loss residuals, there is no ex ante reason to believe they might be distributed normally, without making any assumptions on the irreducible error, and hence the Wilcoxon test could offer a better general approach to comparing loss residuals. A detailed approach can be found on pages 38-52 of Corder and Foreman [2009].

The last section established a principled way to test for conditional independence in a set of variables using predictive inference. Section 5.1 will outline the usage of these properties in a conditional testing routine, which will be applied to graphical model structure learning in Section 5.2

4 Graphical Models

This section will first introduce the concepts of graphs and graphical models and outline their respective properties, and then provide on overview of graphical model structure learning algorithms, which will later be tackled in a novel way using the conditional independence test outlined in section 3.

4.1 Overview

A probabilistic graphical model is a graphical (see Figure 3) representation of a probability distribution. Graphical models encode a set of independencies for a probability distribution P over a random variable X, $X \sim P$. Graphical model theory usually defines two main tasks, learning (a model) and inference (on a given model). This section will give a quick overview of different graphical model structures and introduce the state-of-the-art structure learning algorithms. For an extended introduction to graphical models, the reader might refer to Koller and Friedman [2009] or Barber [2012].

There are two different approaches to graphical model structure learning. One can consult experts to manually create a graphical model, by starting with an outcome, and then iteratively backtracking the causes until all the relevant variables are covered by a model (see Figure 4 for an example).

The second approach, which will be the focus of this paper, is concerned with

In graph theory, a graph $\mathcal{G}$ is a pair (V, E) of sets s.t. the elements in V denote the vertices (nodes) of the graph, and the elements in E, which are 2-element subsets of V, denote the edges (links) in the graph [Diestel, 2016, ch. 1]. $\mathcal{G}$ is then visualized by drawing all vertices V and edges between all pairs of vertices in E. They are used for many purposes, such as in optimization theory (computational graphs) or the study of networks, where the interest usually lies in properties of the graphs, rather than the graph itself. For the purposes of this paper, graphs are used to describe properties of the joint distribution over a set of random variables. The vertices denote individual random variables, and the edges express dependence. That is, for a set of random variables $X = [X_1, ..., X_p]$, if a graph consistent with the probability distribution over X shows an edge between the vertices corresponding to the random variables X_i and X_j, there is shared information between them that is not contained in any subset of $X \setminus \{X_i, X_j\}$.

Figure 3: Graphs in graph theory

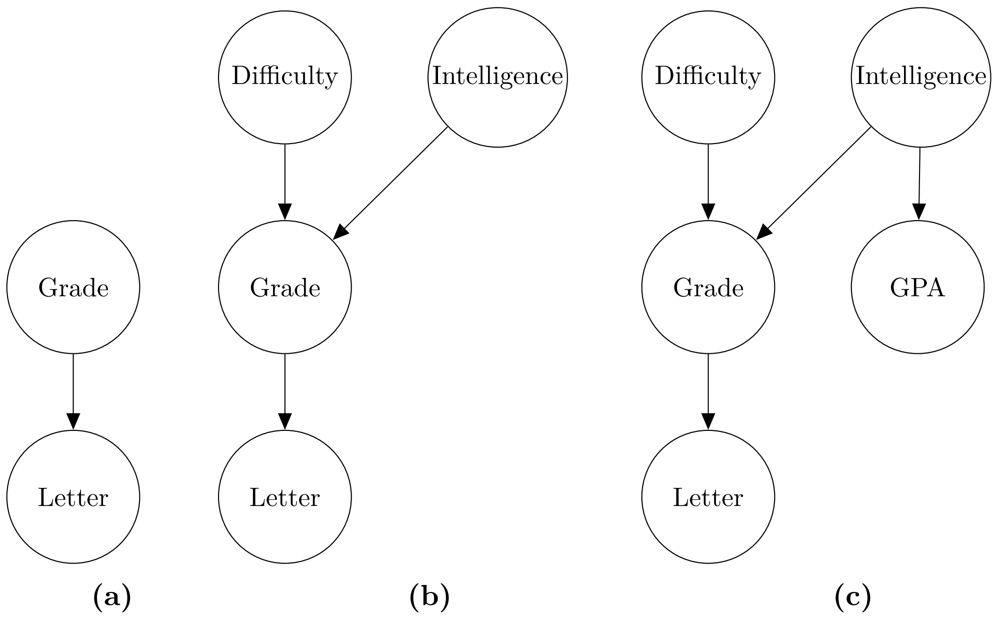


Figure 4: Example of expert based graphical model structure learning (backtracking).
(a) The quality of an academic reference letter is determined by the grade
(b) The students intelligence and course difficulty determine the grade.
(c) Knowing a students GPA gives additional information about the state of intelligence

learning a graphical model from data, without the inclusion of any domain knowledge. Hybrid approaches (combining data and expert inputs) exist, but are outside the scope of this paper.

4.2 Types of graphical models

There are many different types of graphical models, this section will introduce two of the most basic ones, one for directed and one for undirected graphs.

Bayesian Networks

A Bayesian network is a directed acyclic graph over a set of random variables (the nodes on the graph). Each node X_i has a (potentially empty) set of descendants,

Definition 4. *A node X_j is a descendant of X_i in the graph $\mathcal{G}$ if there is a directed path from X_i to X_j, where a path is any connection of links that lead from X_i and X_j.*

That is, if there is a path following the arrows in $\mathcal{G}$, going from X_i to X_j. Since the graph is acyclic, no cycles exist, and following the arrows in the graph, the same variable can never be reached twice. Examples of such a networks are shown in Figure 4.

Bayesian Network graphs arise in a natural way when considering the factorization properties of probability distributions P. Assume we are interested in a probability distribution P over the $Difficulty$ of a course, a students $Intelligence$, the $Grade$ a student achieved in a course, and the quality of a reference $Letter$ received from the tutor of the course, $P(Difficulty, Intelligence, Grade, Letter)$. Further assume the following is true

- The quality of the $Letter$ is solely determined by the $Grade$ a student received in the course. That is, $Letter \perp\!\!\!\perp \{Difficulty, Intelligence\}|Grade$
- The $Grade$ of a student depends on the $Difficulty$ of the course and his $Intelligence$
- $Difficulty$ and $Intelligence$ are not causally influenced by any other variable in the graph

This gives a natural way to order the variables for factorization. $Difficulty$ and $Intelligenc$e are so-called root nodes, hence their order is irrelevant, however $Grade$ depends on both of them and $Letter$ depends on $Grade$, giving the ordering: $\{Letter, Grade, Difficulty, Intelligence\}$, which will now be denoted as $\{L, G, D, I\}$. An ordering $\{X_1, ..., X_n\}$ implies that for $\forall i, j \in \{1, ..., n\}$ and $i < j$, X_j can not be a descendant of X_i. The distribution is then factorized, according to the ordering,

$$P(L, G, D, I) = P(L|G, D, I)P(G, |D, I)P(D|I)P(I),$$

a standard way to factorize distributions. Working in the independence statements gives

$$P(L, G, D, I) = P(L|G)P(G, |D, I)P(D)P(I),$$

since $L \perp\!\!\!\perp \{D, I\}|G$ and $D \perp\!\!\!\perp I$. Returning to Figure 4 (b) shows that this factorized distribution exactly matches the arrows in the graph, which start at the parents (the conditioning variables) and lead to the children (the variable that the conditional distribution is over).

To understand the independence statements encoded in a Bayesian Network, one needs to first distinguish between active and passive trails [Koller and Friedman, 2009, p. 71]. Let a trail be any path between X_i and X_j on $\mathcal{G}$ and a v-structure a structure such that $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, where X_i is a descendant of both X_{i-1} and X_{i+1}.

Definition 5. *A trail between X_i and X_j is active given a conditioning set Z, if for every v-structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ along the trail, X_i or one if it's descendants is in Z and no other variables on the trail are in Z.*

In Figure 4 (c) that means, the trail from *Letter* to *GPA* is active only if conditioned on either the empty set or *Difficulty*, and the trail between *Intelligence* and *Difficulty* is only active if *Grade* or *Letter* is in the conditioning set.

Definition 6. *Barber [2012] If there is no active trail in the graph $\mathcal{G}$ between the nodes X and Y given Z, then $X \perp\!\!\!\perp Y | Z$ in any distribution consistent with the graph $\mathcal{G}$*

Markov Network

A Markov Network, or Markov Random Field, is an undirected graph over a set of random variables. While the Bayesian Network is defined over a probability distribution, the Markov Network arises from a factor product (which can be normalized to result in a valid probability distribution). The scopes of these factors determine the complexity of the system, if the scope of a factor covers all the variables in the graph, the graph is unconstrained, whereas any constraint to smaller scopes decreases its complexity.
Figure 5 shows the Markov Network for the normalized factor product

$$P(a,b,c,d,e,f,g) = \frac{1}{Z}\phi_1(a,b,c)\phi_2(b,c,d)\phi_3(c,e)\phi_4(e,f)\phi_4(f,g)\phi_5(e,g),$$

with the normalizing constant (partition function) Z. Small letters denote realizations of the capital-lettered nodes. If P is a probability mass function,

$$Z = \sum_{a,b,c,d,e,f,g} \phi_1(a,b,c)\phi_2(b,c,d)\phi_3(c,e)\phi_4(e,f)\phi_4(f,g)\phi_5(e,g),$$

where $\phi(s)$ is such that $\phi : S \rightarrow \mathbb{R}$, $\forall s \in S$.
If P is a probability density function over continuous variables, Z would be attained by integrating over the support of the variables in the scopes of P.

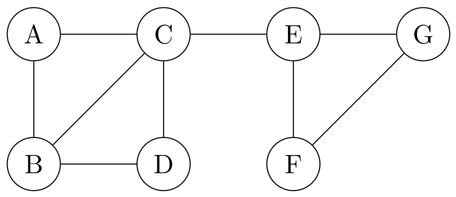


Figure 5: Markov Network

Since the links are undirected, independence statements arising from Markov Networks are much simpler. A path between X_i and X_j, $i \neq j$ is active given Z, if no node on the path is in Z. To determine if X_i and X_j are conditionally independent given Z in all distributions consistent with $\mathcal{G}$, one again considers all paths going from X_i to X_j. $X_i \perp\!\!\!\perp X_j | Z$ if there is no path between X_i and X_j that is active given Z.
So to attain a Markov Network that is consistent with a probability distribution, one has to consider the pairwise conditional independence properties of the variables that the nodes in the graph represent. In general, if in a Markov Network there is no link between a variable X_i and X_j, then $X_i \perp\!\!\!\perp X_j | X \setminus \{X_i, X_j\}$ in any distribution P over X consistent with the graph. These are called the pairwise Markov-independencies of the graph [Friedman et al., 2001, ch. 17.2].

The independencies that can be encoded using Bayesian and Markov Networks differ. Bayesian Networks are natural for causal relationships, distributions where an ordering can be attained and thus have nice factorization properties, while Markov Networks are natural for expressing a set of pairwise Markov independencies. Additionally, Bayesian Networks can be transformed into an undirected graph by a process called "moralization" [Koller and Friedman, 2009, p. 135]. Through this process, information about independencies can however get lost.

Figure 6: Two DAGs equivalent with respect to independence statements

4.3 Structure learning

There are two dominant approaches to structure learning, independence testing-based and score-based methods, however they both suffer to varying degrees from the same underlying problem: combinatorial explosion. Given a probability distribution P (with associated graph $\mathcal{G}$) over a multivariate random variable $X = [X_1, ..., X_p]$, for undirected networks, between any pair of variables X_i and X_j there can be a link or no link in $\mathcal{G}$. Since the links are undirected, there are thus $\frac{p(p-1)}{2}$ potential edges in the graph, and an exponential number of possible graphs. If one wants to test if X_i and X_j are independent given Z, $Z \subset X$, one needs to test if any path between the two is active, leading to a potentially very large number of tests. A graph's complexity can be decreased by upper bounding the number of edges for each node, however, it was shown that, for directed graphs where each node can have d parents, for $d > 1$, the problem a finding an optimal graph is NP-hard [Koller and Friedman, 2009, p. 811]. Generally, the space of graphs is reduced by making structural (what type of relationships can be captured in the graph) and parametric (constraints on the distribution) assumptions. An additional problem is posed when there exists more than one optimum to the structure learning method, as a result of the fact that different graphs can be equivalent (and thus not identifiable) in the respective search space. An example of this is the two graphs shown in Figure 6, which arises from the fact that if $\{X, Y\} \sim P$, P can be decomposed either into $P(X|Y)P(Y)$ or $P(Y|X)P(X)$, which are equivalent. There are different ways to approach these problems, some of which are outlined below.

Score-based methods

Score-based methods attempt to find a (usually local) optimum of a score function in the space of possible graphs. Examples of this can be the Kullback-Leibler divergence scoring function for directed graphical models, which can be used in a convex algorithm to find the optimal Chow-Liu tree graph [Barber, 2012, p.219]. While this is a convex algorithm, finding the global optimum, it does so under the heavy constraint that each node has only one parent at maximum (resulting in a tree-structured graph). For undirected graphs over discrete variables, gradient descent can be used directly on the log likelihood to find the local optimum [Barber, 2012, p. 221], however each gradient evaluation requires calculation of the partition function Z (by summing over all discrete states), which makes this algorithm very expensive. Many of the algorithms for score-based methods are designed for discrete variables, and not applicable when the variables are continuous. Performance is usually evaluated by calculating the likelihood score on a hold-out sample. One other area not mentioned above is *Bayesian model averaging* (e.g. Fragoso and Neto [2015]), which seeks to improve the above methods by averaging over different model structures. It can be viewed as an ensembling method for the score-based methods mentioned above. State-of-the-art score based methods are often times very expensive or make very strong assumptions on the underlying distribution P, such as tree-structure of the graph [Koller and Friedman, 2009, ch. 18].

Independence test-based methods

Unlike the score-based models, independence test-based models conduct independence tests between variables or sets of variables locally and then aggregate the information to produce the graphical model. These sets of independence tests are then used to infer the structure of the graphical model in a straightforward manner, based on the properties of the respective type of graphical model. An example of this is the PC-Algorithm [Barber, 2012, e.g. p. 214] which is used in attempts at causal discovery. There are two main drawbacks to these methods:

1. Since these methods generally require to conduct tests for each pair of variables, it is difficult to avoid the combinatorial explosion, and algorithms get exponentially more expensive as the number of variables increases

2. A large number of hypothesis tests need to be conducted, introducing the usual multiple testing problems, and potentially leading to bad results

These are the main reasons that independence test-based methods are not usually used when trying to estimate graphical model structure. It is hard to regulate the bias-variance trade off that is readily available for score based methods through the use of constraints such as an upper bound on the number of parents. However, the algorithm in Section 5.2 provides a new way to address the above challenges, based on the predictive conditional independence testing routine:

1. While the combinatorial explosion cannot be avoided, one can provide the user with an algorithm to control the trade-off between power of the algorithm and time complexity. This trade-off is readily available when using the conditional independence testing routine described in Section 5.1, since the user can choose estimation methods of varying power/complexity in order to influence the total run time needed by the algorithm

2. Section 5.1.1 will introduce a false-discovery-rate control routine to provide a tool that lets a user control the proportion of false-positives (erroneously found links in the estimated graphical model structure), regardless of the dimensionality of the data.

3. The link to the supervised learning workflow allows for an adoption of the well understood bias-variance trade-off of that field into the conditional independence test, and thus ultimately into the structure learning algorithm

Note on causal inference

Graphical models, and specifically Bayesian Networks, are a natural way to express causality, since the arrows seem to express causation. When learning graphical models from data, causality can however only be inferred by making very strong assumptions on the underlying distribution, or by collecting data in a manner that allows for causal interpretation, namely in experimental setups. All algorithms outlined in this paper are not to be interpreted as expressing causality, but merely as a collection of statements about association.

5 Graphical model structure learning

This section will first introduce the proposed predictive conditional independence testing routine (PCIT) and important subroutines, and then outline an algorithm for learning the structure of an undirected graph.

5.1 Conditional independence test

Algorithm 1 implements the results from Section 3 to test if a set of variables Y is independent of another set of variables X, given a conditioning set Z (optional, if not provided the algorithm tests for marginal independence). It will later be used as a subroutine in the graphical model structure learning, but can also be used in it's own right for tasks such as determining if a subset X would add additional information when trying to predict Y from Z.

Algorithm 1 Predictive conditional independent test (PCIT)

```
Split data into training a test set
for all variables y ∈ Y do
  on training data:
    find optimal functional $f$ for predicting y from Z
    find optimal functional $g$ for predicting y from {X,Z}

  on test data:
    calculate and store p-value for test that gen. loss of $g$ is lower than $f$

end for

if test should be symmetric then
  exchange X and Y, repeat above process
end if

p_values_adjusted ← Apply FDR control to array of all calculated p-values
return p_values_adjusted
```

When the test is used as a marginal independence test, the optimal prediction functional for f is the functional elicited by the loss function, that is, the mean of y for continuous outputs, and the class probabilities of y for the discrete case (Appendix A). The link to supervised learning further allows/forces the user to distinguish between two cases. Independence statements are symmetric, if X is independent of Y, then Y is independent of X, in both the marginal and conditional setting. The same cannot be said in supervised learning, where

adding X to predicting Y from Z might result in a significant improvement, but adding Y does not significantly improve the prediction of X, given Z (as can be seen in the asymmetry of OLS). So if a user is interested in a one-sided statement, one can run the algorithm for a single direction, for example to evaluate if a new set of variables improves a prediction method, and is thus worth collecting for the whole population. If one is interested in making a general statement about independence, the test is "symmetrized" be exchanging X and Y, and thus testing in both directions.
It is important to distinguish between the two types of tests and null-hypotheses in this test. On one hand (in the symmetric case) for each variable in X and Y, it will be assessed, if adding X to the prediction of $y \in Y$ from Z results in an improvement (and vice versa). The null-hypothesis of this "prediction-null" is that no improvement is achieved. After all these p-values are collected, a multiple testing adjustment is applied (Section 5.1.1), after which, the original null-hypothesis, that X and Y are conditionally independent, is assessed. We reject this "independence-null", if any one of the "prediction-nulls" can be rejected after adjustment. The p-value of the "independence-null" hence reduces to the lowest p-value in all the "prediction-nulls". As such, the null of the independence test is that all the "prediction-nulls" are true. False discovery rate control, the chosen multiple testing-adjustment, is appropriate since it controls the family-wise error rate (FWER), the probability of making at least one type 1 error, if all the null-hypotheses are true [Benjamini and Hochberg, 1995].

5.1.1 False-discovery rate control

The account for the multiple testing problem in Algorithm 1, the *Benjamini - Hochberg - Yekutieli procedure* for false-discovery rate (FDR) control is implemented [Benjamini and Yekutieli, 2001]. In their paper, they state that traditional multiple testing adjustments, such as the Bonferroni method, focus on preserving the FWER. That is, they aim to preserve the probability of making any one type 1 error at the chosen confidence level. As a result, tests are usually very conservative, since in many multiple-testing scenarios the p-values are not independently distributed (under the null), and thus the power of these tests can be significantly reduced.

In their 2001 paper, they propose to control the false discovery rate instead, *"[..] the expected proportion of erroneous rejections among all rejections"*, as an alternative to the FWER. The FDR allows for more errors (in absolute terms) when many null-hypothesis are rejected, and less errors when few null-hypotheses are rejected.

Algorithm 2 The Benjamini-Hochberg-Yekuteli Procedure

```
Require: Set $\{p_{(i)}\}_{i=1}^{m}$ s.t. $p_j$: p-value for observing $X_j$ under $H_0^j$
1: Sort the p-values in ascending order, $p_{(1)} \leq ... \leq p_{(m)}$
2: Let $q = \alpha/(\sum_{i=1}^{m} 1/i)$ for chosen confidence level $\alpha$
3: Find the **k** s.t. $k = max(i : p_{(i)} \leq \frac{i}{m}q)$
4: Reject $H_0^j$ for $j \in 1, ..., k$
```

Algorithm 2 shows the procedure outlined in Benjamini and Yekutieli [2001]. They showed, that this procedure always controls the FDR at a level that is proportional to the fraction of true hypotheses. As hinted at before, while this algorithm controls the false-discovery rate, in the special case where all null-hypotheses in the multiple testing task are assumed to be true, it controls the FWER, which then coincides with the FDR. This is especially useful, since both scenarios occur in the graphical model structure learning routine described in Section 5.2.

For the choice of optimal false discovery-rate for an investigation, even more so than in the classical choice of appropriate type 1 error, there is no simple answer for which rate might serve as a good default, and it is highly application dependent. If the goal of the procedure is to gain some insight into the data (without dire consequences for a false-discovery), a user might choose a FDR as high as 20%, meaning that, in the case of graphical model structure learning, one in five of the discovered links is wrong on average, which might still be justifiable when trying to gain insight into clustering in the data. This paper will still set the default rate to 5%, but deviate willingly from this standard whenever deemed appropriate, as should any user of the test.

5.1.2 Improving the prediction functionals

In practice, when assessing the individual "prediction-nulls" in Algorithm 1, the power of the test (when holding the loss function constant) depends on the capability of the the prediction method to find a suitable functional g that outperforms the baseline f. That means, a general implementation of the

	Regression	Classification
Stage 1	ElasticNetCV	BernoulliNB
	GradientBoostingRegressor	MultinomialNB
	RandomForestRegressor	GaussianNB
	SVR	SGDClassifier
		RandomForestClassifier
		SVC
Stage 2	LinearRegression	LogisticRegression

Table 1: Prediction functionals used for Stacking/Multiplexing

independence test needs to include a routine to automatically determine good prediction functionals for g and f. The implementations in the pcit package presented in Section 6 supports two methods for the automatic selection of an optimal prediction functional. Both methods ensemble over a set of estimators, which are shown in Table 1. The prediction functionals refer to the estimator names in sklearn[1]. Some are constrained to specific cases (e.g. *BernouilliNB*, which only applies when the classification problem is binary).

Stacking

Stacking refers to a two-stage model, where in the first stage, a set of prediction function is fit on a training set. In the second stage, another prediction function is fit on the outputs of the first stage. If the prediction function in the second stage is a linear regression, this can be viewed as a simple weighted average of the outputs in the first stage. In theory, stacking allows the user to fit one single stacking predictor instead of having to compare many potential prediction functions based on the model diagnostics, as in the second stage, the optimal methods get more weight (in the expectation). As an additional effect, improvement in the prediction accuracy through ensembling of predictors can take place (see e.g. Section 19.5 of Aggarwal [2014]). The used stacking regressor and classifier can be found in the Python package *Mlxtend* [Raschka, 2016].

Multiplexing

When multiplexing over estimators, the training set is first split into a training and validation set, a common procedure to find optimal hyperparameters.

[1]Details can be found here `http://scikit-learn.org/stable/modules/classes.html`

After, the predictors are fit individually on the training set, and each predictors expected generalization loss is estimated on the validation set. One then proceeds by choosing the predictor with the lowest estimate for the empirical generalization loss, and refits it using the whole training data (including the former validation set), and then uses the fitted estimator for prediction.

5.1.3 Supervised learning for independence testing

Algorithm 1 shows how to leverage supervised learning methodology into a conditional independence test. This has a major advantage over the methods outlined in Section 2, as the supervised prediction workflow is of utmost interest to many areas of science and business, and, as a result, a lot of resources are going into developing and improving the existing methods. By making a link between predictive modelling and independence testing, the power of independence testing will grow in the continuously increasing power of the predictive modelling algorithms.

5.2 Undirected graphical model structure estimation

This section outlines a routine to learn the vertices in a directed graph (the skeleton) for a data set by conducting a range of conditional independence tests with the null-hypothesis of independence. Section 4.2 outlines the conditional independence statements of a Markov network. In a directed graph, if variables X_i and X_j have no direct edge between them, they are conditionally independent given all other variables in the network.

Algorithm 3 Undirected graph structure estimation

```
1: for any combination $X_i$, $X_j$ s.t. $i \neq j$ do
2:     $X_- \leftarrow X \setminus \{X_i, X_j\}$
3:     $p\_val_{i,j} \leftarrow$ p-value for test $X_i \perp\!\!\!\perp X_j | X_-$
4: end for
5: p_val_adj ← Apply FDR control on p_val matrix
6: return p_val_adj
```

Algorithm 3 describes the skeleton learning algorithm for an input set $X = [X_1, ..., X_p]$, by considering all possible pairs of variables in the data set, and testing if they are conditionally independent given all other variables. The output p_val_adj is a symmetric matrix with entries i,j being the p-value for the hypothesis that in the underlying distribution, X_i and X_j are independent, given all other varialbes in X, and hence in the graph $\mathcal{G}$ describing it, there is

no link between the vertices for X_i and X_j. Ultimately, links are drawn where the adjusted p-values are below a predefined threshold.
There are O(p^2) function evaluations in the for-loop, where p is the dimensionality of the data. Section 8 will provide performance statistics for the algorithm and show applications on real data sets.

6 pcit package

The Python package implementing the findings and algorithms of this paper can be found on `https://github.com/SamBurkart/SupervisedGM` and is distributed under the name pcit in the Python Package Index. It has dependencies to the following packages:

- Scipy [Jones et al., 2001]
- Sklearn [Pedregosa et al., 2011]
- Mlxtend [Raschka, 2016]

This section will first provide an overview of the structure and important routines of the package. As this implementation can be thought of as a wrapper for scikit-learn (sklearn) estimators, this section will then describe the sklearn interface and how the package interacts with the sklearn estimators.

6.1 Use cases

The package has two main purposes: conditional independence testing and undirected graph structure learning. The PCIT serves as a readily available tool to test for conditional independence, without the need for much hyperparameter tuning. This benefits users trying to perform conditional independence tests, such as assessing if a data set adds information to existing data in a prediction task. Alternatively, the test could be used for hedging purposes in a multivariate test to determine which financial products are independent from the ones already in the portfolio. The proposed graphical model structure learning routine can be used for tasks such as finding clusters in the data as part of an exploratory data analysis (section 8.2.1) or thoroughly investigating association structures in the data (section 8.2.2).

6.2 API

The package introduces three main routines

MetaEstimator

The MetaEstimator class defines a set of functionalities that, on one hand, automate finding optimal prediction functions for a chosen ensembling method, and on the other hand, adds methods needed for execution of the following algorithms, such as getting the prediction residuals. It is initialized for sensible defaults, which should generally lead to good results, but can be changed for specific tasks (such as more complex base estimators for more powerful, but more computationally demanding, predictions). The ensembling methods used for the estimator are described in Section 5.1.2. For regression, the square loss is used for training and outputting the residuals, for classification, the logistic loss serves as the loss function. The MetaEstimator implements a fit_baseline method, in which case the functionals elicited by the loss function are fitted. On an important note: Some estimators in sklearn demand for scaled data, so if manually chosen estimators are used (as opposed to the defaults), the user should take steps to scale the data to be as expected by the chosen estimator.

PCIT

PCIT implements the conditional independence algorithm described in 1 to test if two samples from from probability distributions over conditionally (or marginally) independent random variables. The MetaEstimator class is used as a prediction functional, and hence the user can trade off between computational complexity and power by adjusting the chosen MetaEstimator. That is, if speed is important, the used MetaEstimator should be a combination of base estimators and ensembling method, that is quick in execution, whereas if computational resources are vast and a more powerful test is needed, the used base-estimators should be highly tuned.
Inputs:

- **X and Y**: Input sets for which (conditional) independence is assessed
- **Z**: Conditioning set (default = $\emptyset$)
- **estimator**: Object of class MetaEstimator
- **parametric**: Type of test for comparison of loss residuals

- **confidence**: Confidence level for test
- **symmetric**: Adds routine for symmetric results if True (= default)

The function assesses if $X \perp\!\!\!\perp Y | Z$, and then returns the following statistics:

- **p_values_adj**: The p-values for the "prediction-nulls" of each $y \in Y$
- **independent**: Tuple, first value indicates if "independence-null" is rejected (0) or not (1), second value is p-value of hypothesis
- **loss_statistics**: Difference in RMSE (see theorem 3) for baseline f and alternative g loss residuals with associated standard deviation, for each $y \in Y$. Only applicable for continuous Y[2].

find_neighbours

find_neighbours implements Algorithm 5.2 to learn the undirected skeleton for an input data set X, and the conditional independence test PCIT. Returns the undirected skeleton for a chosen confidence level.

6.3 Examples

This section will provide some simple examples of how the code is used. For the following it is assumed that data sets X, Y and Z, all of the size $n \times *$, where $*$ is the individual number of dimensions, are loaded. After installing the pcip package, import the relevant functions:

```
from pcip import MetaEstimator, PCIT,
                                    find_neighbours
```

Testing if $X \perp\!\!\!\perp Y | Z$ using the default values:

```
PCIT(X, Y, z = Z)
```

[2]The variance of the difference is estimated by assuming 0 covariance, which generally leads to more conservative confidence intervals for error residuals (due to the irreducible error, prediction residuals for different methods are generally positively correlated)

Testing if $X \perp\!\!\!\perp Y | Z$ with a custom MetaEstimator, multiplexing over a manually chosen set of estimators:

```
from sklearn.linear_model import RidgeCV, LassoCV,
                        SGDClassifier, LogisticRegression

regressors = [RidgeCV(), LassoCV()]
classifiers = [SGDClassifier(), LogisticRegression()]
custom_estim = MetaEstimator(method = 'multiplexing',
                    estimators = (regressors, classifiers))

PCIT(X, Y, z = Z, estimator = custom_estim)
```

Learning the undirected skeleton of X:

```
find_neighbours(X)
```

Concrete outputs will be shown in section 8.

7 Scikit learn workflow

Scikit-learn (sklearn) is a package in the Python programming language (`https://www.python.org/`) that aims to provide a user with a consistent, easy-to-use set of tools to analyze data[Pedregosa et al., 2011]. It is one of the most-used tools in today's supervised learning community, which is why it is chosen as the supervised prediction workflow to build on for the predictive conditional independence test. This section will outline the advantages of basing the test on the sklearn package.

7.1 Sklearn interface

Sklearn is built around estimator objects, which implement a consistent set of methods. Estimators provide a fit and, if applicable, a predict method, in order to be able to fit the estimator to training data and then predict on a test set. Additionally, sklearn provides standardized approaches to model selection and hyperparameter-tuning as well as data transformation and ensembling methods (see Buitinck et al. [2013] for a more thorough discussion). The methods can easily be combined to create more powerful estimators. Defaults are chosen sensibly so that in most cases, an initial fit of a method to data requires little manual parameter specification from the user's side. While the highly automated and simplified approach of sklearn lowers the bar of entry when aiming to generate knowledge from data, it also comes with a downside. For most statistical applications that exceed fitting and predicting from a data set, such as inference on the parameters and hypotheses about prediction accuracies, the relevant subroutines are missing from the API. Relevant statistics can be attained by interfacing it with other packages (such as SciPy), but this usually results in an overly complicated procedure. However, for the purposes of this paper, the functionalities provided by sklearn suffice, the next section extends on how this paper's package interfaces with sklearn.

7.2 Wrapper for Sklearn estimators

As described in Section 6.2, the conditional independence test described in Algorithm 1 uses the newly defined **MetaEstimator** class to automate determining the optimal prediction functional for a given task. It does so, by ensembling over a set of base estimators from sklearn. These are either chosen to be the sensible defaults described in Table 1, or can be passed by the user as a tuple of lists of sklearn base estimators. This is required since regression and classification tasks rely on vastly different prediction functionals, and thus need to be specified separately. As a general rule, the passed regressors need to have a fit and a predict method, whereas the classifiers need to have a fit and a predict_proba method. Requirements might be more stringent for certain types of data or certain estimators, however specifying an unsuitable estimator class will result in an error message as specified by the respective class, allowing the user to either remove the unsuitable class or proceed with the sensible defaults. As mention before, this gives a user a flexible tool to trade off between power and computational complexity. If in need of a fast method, one can use an algorithm that runs in linear time, such as stochastic gradient descent linear regression, whereas if a test with high power is needed, one can pass hyper-tuned estimators to the function, that take longer to run but generalize the data much better.

8 Experiments

This section will first evaluate the performance of the proposed algorithms, and then provide some examples of applications on real world data sets. All performance tests can be found on Github[3].

8.1 Performance tests

This section will conduct performance tests on the algorithms derived in Section 5 to evaluate the performance of the conditional independence testing subroutine. First the power of the the predictive conditional independence routine is bench-marked against current state-of-the-art methodology, then various tests on the directed graph structure learning algorithm are conducted.

8.1.1 Performance of conditional independence test

In this section the conditional independence routine will be bench-marked against the previous research, namely the kernel based approach for conditional independence testing [Zhang et al., 2012]. The code for the test is available on Github[4]. To conduct a test using real world data, as opposed to synthetic data, the UCI Wine Repository data set [Lichman, 2013] is used as follows:

- The columns 'Alcohol', 'Malic Acid' and 'Magnesium' are randomly permuted (to make them independent) and will serve as X, Y and $noise$ arrays respectively

- Create vector Z by individually sampling vectors X', Y' and $noise'$ of size n with replacement from X, Y and the noise vector, and then calculate $Z_i = \log(X_i') \times \exp(Y_i') + u * \sqrt{noise_i'}$, $i \in \{1, ..., n\}$, where u is the sign, uniformly drawn from $\{-1, 1\}$

This results in a scenario, where $X \perp\!\!\!\perp Y$, but $X \not\perp\!\!\!\perp Y | Z$, and the signal to noise ratio is high for small sample sizes. The test will be conducted by increasing the sample size from 100 to 5000, and calculating the run times and power for both approaches. For each sample size, the PCIT is run 500 times, and the KCIT is run 200 times. The only exception is $n = 5000$ for the KCIT, which is run 25 times, since the time complexity is high. Both methods are run for their default values, without additional manual tuning, and at a confidence level of

[3]`https://github.com/SamBurkart/pcit/tree/master/tests`

[4]`https://github.com/devinjacobson/prediction/tree/master/correlations/samplecode/KCI-test`

	n	100	200	500	1000	2000	5000
PCIT	Power	0.020 (0.006)	0.046 (0.009)	0.332 (0.021)	0.672 (0.021)	0.832 (0.017)	0.970 (0.007)
	Time (s)	0.32	0.38	0.49	0.624	1.31	4.79
KCIT	Power	0.050 (0.015)	0.085 (0.019)	0.185 (0.027)	0.325 (0.033)	0.8 (0.028)	1 (*)
	Time (s)	0.57	1.25	9.8	44	383	4758
	Stand. difference	-1.8	-1.78	4.25	8.85	0.97	*

Table 2: Performance statistics for the newly proposed predictive conditional independence test (PCIT) and the kernel based approach (KCIT). The values in brackets show the estimated standard errors. The last row shows the standardized difference between the power estimates, PCIT - KCIT

.

5%. Each time, $\{X', Y', Z\}$ is sampled, and then the conditional independence tests are applied and the run time is recorded. If they reject independence at a 5% level, the round counts as a success, 1, otherwise 0. The power and run times are the calculated by averaging over the results, and standard errors for the power are attained by realizing that the standard error of the power for a sample size of n is the standard error of $\frac{X}{n}$, where $X \sim Bin(n, \theta)$, where θ is the observed power.

The results are shown in Table 2. The power at the higher end of the sample sizes seems to be similar (it is important to note that the power of 1 for the KCIT for $n = 5000$ was achieved on 25 resamples only), where as in the range 500 to 1000 samples, the proposed predictive conditional independence test shows a higher power. For small n, the KCIT seems to fare significantly better, however both approaches have a very low power, and the PCIT especially shows the power levels below the confidence levels, which might indicate a discrepancy between true type 1 error and expected type 1 error. Important to note is the very high computational complexity of the kernel-based approach for a data set of size 5000, with a run time of approximately 80 minutes, while the predictive conditional independence test still has a very low run time of 4.8 seconds. This is to be taken with a grain of salt, since the tests were run in different languages (PCIT in Python, KCIT in Matlab), but it is apparent that PCIT scales much better than KCIT, while the both converge to a power of 1.

8.1.2 Performance of structure learning algorithm

Data

The data used in the performance tests is generated as follows (specifics can be found on Github):

1. Sample a random positive definite, sparse, symmetric precision matrix M making use of diagonal dominance. The size of the entries of this matrix (relative to the diagonal) determine the signal to noise ratio, and are thus thresholded to 0 if below a certain value.

2. Invert M, $M' = inv(\text{M})$ and use M' as covariance matrix to sample from a multivariate Gaussian distribution. This has the effect, that the zero-entries in M express zero-partial correlations [Friedman et al., 2001, ch. 17.3] between respective variables (and hence, lack of a direct link between two nodes in the graph). That is, for a multivariate Gaussian random variable X, $X = [X_1, ..., X_p]$, $M_{i,j} = 0 \implies X_i \perp\!\!\!\perp X_j | X \setminus \{X_i, X_j\}$.

3. Sample $\mathcal{D}$, a data set of size n, from the multivariate normal $P = N(0, M')$. The choice of n will allow to evaluate the algorithms performance for an increasing signal to noise ratio.

4. The graph $\mathcal{G}$ consistent with the probability distribution P is given by M, where edges occur between variables X_i and X_j, if $M_{i,j}$ is non-zero.

Undirected graph structure estimation

Now, Algorithm 3 will be tested by getting an estimate $\hat{\mathcal{G}}$ of the structure of an undirected graph $\mathcal{G}$ induced by the distribution P from which the data set $\mathcal{D}$ was drawn. The performance evaluation will be guided by three metrics:

- False-discovery rate: The FDR is the fraction of type 1 errors (edges in $\hat{\mathcal{G}}$ that are not in $\mathcal{G}$) over the total number of identified links in the learned $\hat{\mathcal{G}}$

- Power: Number of found edges (links in true graph $\mathcal{G}$ that were found by the structure learning algorithm, and are present in $\hat{\mathcal{G}}$) over the total number of links in $\mathcal{G}$

- Time: Run time needed to execute the algorithm

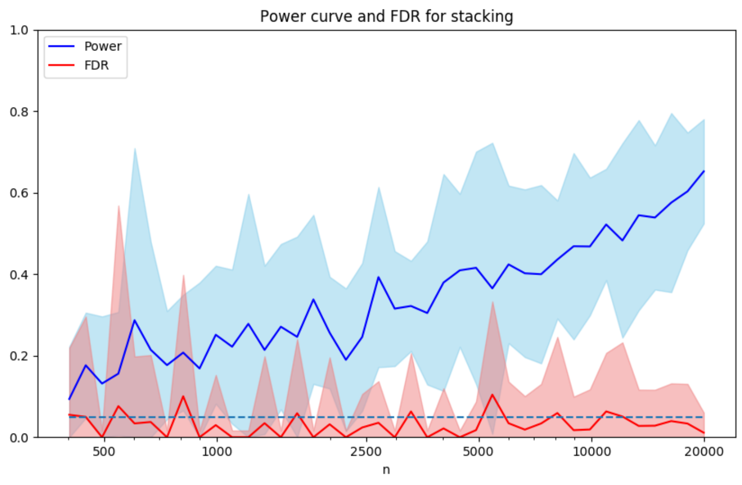
	FDR	Time (sec)
No ensembling	3.09%	30
Stacking	3.03%	450
Multiplexing	2.75%	1000

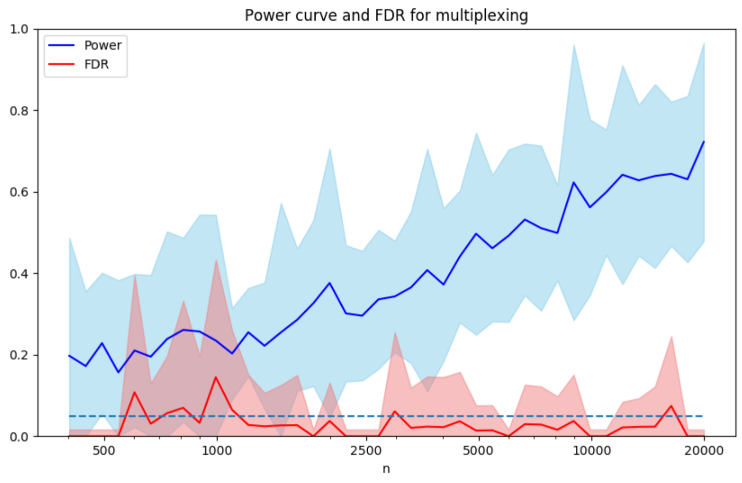
Table 3: False-discovery rates and run times for a data set of 22000 for all used methods

For the test, the number of random variables is chosen to be 10. This means, that each node in the graph has up to 9 neighbours, and the total number of possible undirected links (before inducing sparsity) is 45. The sparsity parameter is chosen in a way that generally between 10 and 15 of those links exist in the true graph induced by P. The size and sparsity of the graph are chosen to produce estimators of the metrics with reasonably low variance, but are otherwise arbitrary. The sample sizes range from approximately 1100 to 22000, increasing in steps of 10%. For each sample size, 10 tests are run to decrease the variance in the estimators. The test is conducted for conditional independence testing using stacking and multiplexing, as well as without using any ensembling method, which, since the data is continuous, means Elastic Net regularized regression is used.

If the algorithms work as expected, the FDR is expected to be at or below 5%. High power-levels indicate a better performance of the algorithm, with respect to this specific task. At the very least, the power level is expected to increase in the number of samples, indicating that asymptotically the routine will find the correct graph.

Table 3 shows the average FDR and run times for each of the three methods. The average FDR seem to be similar across all three methods, whereas the computational complexities differ by a large amount. No ensembling runs very quick, about 15 times faster than stacking, which itself only takes about half as long as multiplexing. This is the case, since multiplexing requires the calculation of performance measures for each used estimator. Figure 7 shows the power and FDR of the algorithm for increasing sample size. The FDR for all 3 methods seem to be slightly higher for small sample sizes than they are for large sample sizes, but they are generally around or below the desired 5% in the expectation (the variances are quite high, as the number of possible reruns is low due to the computational complexity of the multiplexing method). While it might seem surprising that stacking and multiplexing are outperformed by





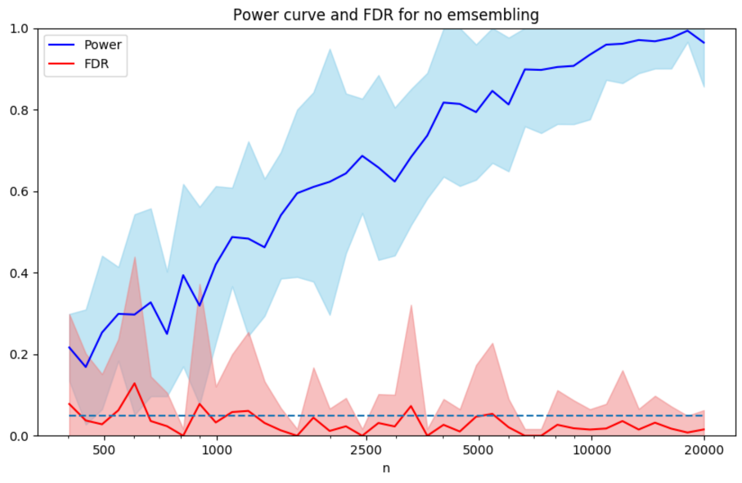


Figure 7: Power (blue lines) and FDR (red lines) for increasing sample size for all 3 methods. The transparent blue and red areas denote the 90% confidence intervals, the dashed line shows the expected FDR (0.05)

the no-ensembling method, one has to remember that the ensembling is used to choose the optimal prediction functional automatically from a set of estimators. However, the data is multivariate Gaussian, for which ridge regression is optimal in terms of generalization performance. While stacking and multiplexing are tasked to find this property from a large set of estimators, the estimator used in the no ensembling case is Elastic Net regularized linear regression, a generalization of ridge regression, and hence fares better since there is less variance in finding the optimal estimator. For all three methods, the power increases roughly linearly in exponentially increasing sample sizes, implying that for a test that recovers the truth with high probability, a large data set or a more powerful set of estimators (see Section 8.2.2) might be needed for that specific task. However asymptotically all three tests seem to recover a graph that is close to the truth, in this specific scenario, unless the power starts to plateau before reaching 1 (which there is no indication for). Since the power for the no ensembling case is biased by the fact that it uses an optimal prediction functional, the power curves for stacking and ensembling provide a better estimate for the performance on a new data set of an unknown distribution family. As graphical model structure learning is an inherently hard problem (due to issues such as multiple testing, combinatorial explosion, as outlined in Sections 2 and 4), it is promising that the algorithm finds an increasing fraction of the links while keeping the ratio of false-discoveries constant.

8.1.3 Model consistency on resampled data sets

As a second performance criterion, this section will assess the consistency of the learned structures on resamples from a data set $\mathcal{D}$. Assuming that all the observation in $\mathcal{D}$ are identically distributed, the structure learning method should arrive at the same conclusions on the resamples, less some variance in the process. The used data set is the Auto MPG Data Set[5] containing various continuous and discrete car performance-attributes, from the UCI machine learning repository Lichman [2013]. The data set contains 398 instances of 8 variables. For the purpose of the experiment, data sets of the same size will be sampled with replacement from the full data set 100 times. On each resample, a graph is estimated using the stacking estimator on a 10% FDR level. Each subsample contains about two thirds of the original data points (with

[5] `https://archive.ics.uci.edu/ml/datasets/auto+mpg`

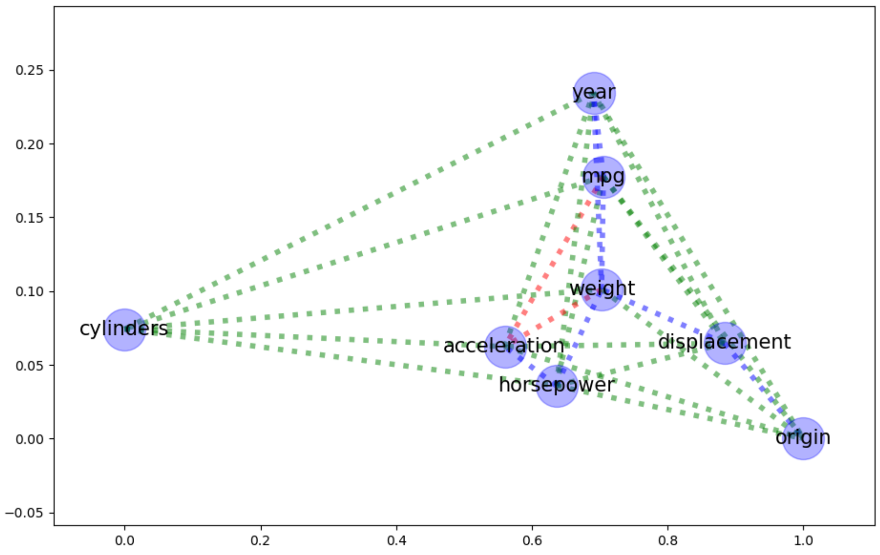


Figure 8: Frequencies of edge occurrence in the learned structure. Green denotes edges that occur in less than 7% (as advocated by FDR) of the models, blue for edges that occur in more than a third of the model, and red everything in between

some instances repeated). This is a commonly used procedure to estimate the sampling distribution of a statistics, and which will here allow us to assess the variance in the learned graph structure. Figure 8 shows the results. On average, there are 6 links in the learned structure, hence the FDR advocates about 0.6 type 1 errors per learned model. The green lines are connections that are found less times than expected by the FDR and the blue lines are connections that are found in a large fraction of the models (over one third of the resamples). The concerning links are the ones in between, for which the null of independence is rejected more than occasionally, but not in a reasonably large fraction of the learned graphs. There are only 2 links in the model for which this occurs, so overall, the variance across learned graphs seems to be reasonably small and the learning process is hence consistent.

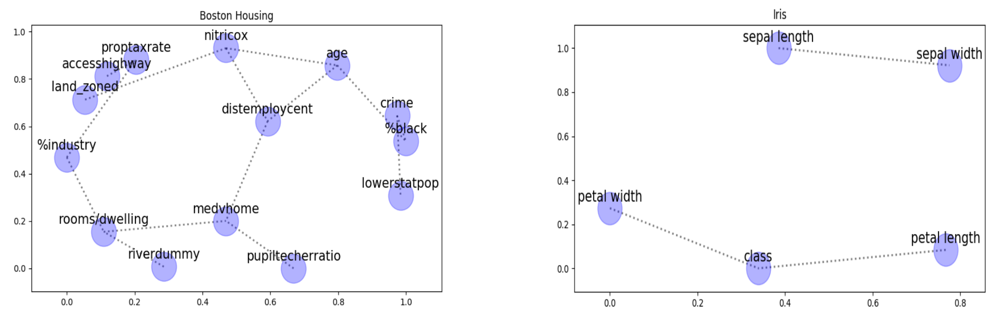


Figure 9: Learned graphical model structures for the Iris and the Boston Housing data set

8.2 Experiments on real data sets

In this section, the outputs of the graphical model structure learning routine are shown on some real world data sets. It will outline some possibilities of the user to trade off between power and computational complexity.

8.2.1 Sklearn data sets: Boston Housing and Iris

Setup: One receives a data set and is interested in initial data exploration. This involves finding related sets of variables, and variables that don't show any obvious relationships with other variables.

Boston Housing and Iris are two standard data sets from sklearn. The housing data set contains 506 samples of 14 variables, originally collected to build a model for predicting house prices (descriptions of the variables can be found online[6]). The Iris data set is a data set for predicting flower species based on petal and sepal measurements with 150 observations in 5 dimensions.
Estimation will take place using the default stacking estimator. Since we are interested in initial exploration, and finding interesting groups of variables, a large FDR (20%) was chosen. Note that, unlike for other (mostly constrained, score-based) structure learning algorithms, the outcome of one experiment (the presence of an edge in the model) does not influence other experiments, and hence, false discoveries do not compromise the estimated structure additionally.

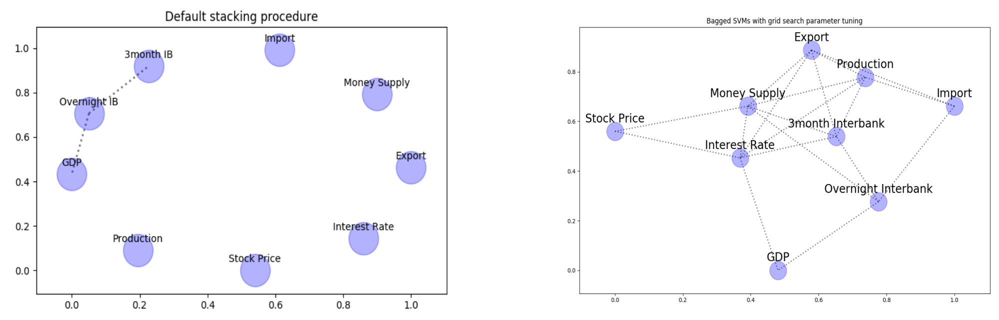
[6]`http://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets`

The results are shown in Figure 9 (graphs drawn with networkx [Hagberg et al., 2008]). For the Boston housing data, seemingly sensible variable groupings occur. In the top left, the variables related to industrialization of a neighbourhood are shown, while on the right, demographic attributes form a cloud of related attributes. For the Iris data set, while length and width of both sepal and petal are related, as expected, it seems that petal size has a higher association with class, and, in fact, width and length are independent given the class.
Both of these analyses require no parameter tuning (the only non-default chosen for this experiment was the adjusted confidence level) and take very little time (less than 15 seconds). The implementation of algorithm 3 thus provides a quick, ready to use tool for initial exploratory data analysis.

8.2.2 Key short-term economic indicators (UK)

Setup: One is interested in finding the relationships within a set of variables and making local conditional independence statements for sets of variables. The focus is on finding associations that we are confident about.

The economic indicator data set contains monthly data of key economic indicators between February 1987 and June 2017 for the UK from the OECD statistics database [OECD, 2017]. The economics indicators are export and import figures, stock price and industrial production levels, overnight and 3 month Interbank interest rates, money supply (M4) and GDP. This rather small data set, around 369 instances of 9 variables, will outline the possibility of a user to trade off between computational complexity and power of the learning algorithm. As for most economic data sets, the signal to noise ratio can is quite low. Figure 10a shows the structure learned by the default approach for confidence (false discovery-rate) level of 5%, with a run time of 15 seconds. While the ground truth is not known (economic theory aside), it is apparent that many links are missed by the fast default approach. This becomes evident when considering a variable like the GDP, which (by definition) includes exports and imports, and hence some connection between the variables should exist. If there is need for a more powerful approach, a more powerful estimator can be chosen. Figure 10b show the learned structure for an estimator that resamples the data 50 times and learns 50 different Support Vector Machines with automatically tuned hyperparameters. The implementation of this



(a) Default Approach **(b)** Bagged Support Vector Machines

Figure 10: Learned graphs for the Economic Indicator data set for the default approach (left) and a more powerful approach using bagged SVM's (right)

is straightforward, since sklearn provides hyperparameter and bagging wrappers, so an estimator can be tuned easily and passed to the MetaEstimator. The graph shows the many edges that were found. While it is impossible to judge the correctness of the graph, it seems that some reasonable groups of variables are found, such as industrial production, exports and imports (in economic theory, countries seek to balance their trade deficit), or the grouping of stock prices and the main drivers of inflation, money supply and interest rates. Additionally, all the variables are connected in some way, which would be expected from a data set of economic indicators. The computational complexity of this approach was rather high, with a run time of about 20 minutes, however this shows how easily the user can trade off between power and complexity,without having a large effect on the false discovery-rate (as shown in Section 8.1).

9 Discussion

This paper has introduced a novel way for multivariate conditional independence testing based on predictive inference and linked to a supervised learning workflow, which addresses many of the current issues in conditional independence testing:

- **No manual hyperparameter-tuning**: By connecting the classical task of independence testing to supervised learning, and it's well-understood hyperparameter tuning capabilities, there is no further need for the heuristics and manual tuning prevalent in current state-of-the-art conditional independence tests
- **Low computational complexity**: By linking independence testing to one of the most-researched fields in data science, predictive modelling, the high level of power in the contemporary prediction methods can be directly benefited from, and any efficiency gains in the latter directly benefits the former.
- **Multivariate random variables**: By connecting the problem with the task of supervised learning, the method easily deals with the multivariate case, largely removing any issues concerning dimensionality of the data

The test was bench-marked against current state-of-the-art conditional independence tests, and showed on equal or better performance in regions where the power of the tests exceeded 10%.
Subsequently, the PCIT was applied in a method for learning the structure of an undirected graph best describing the independence structure of an input data set. The combination of the new conditional independence test and the structure learning algorithm address some of the current issues in graphical model structure learning:

- **Quadratic time complexity**: While current exact algorithms, such as the PC-algorithm, often require exponential time in the number of variables, the proposed algorithm runs in quadratic time
- **Exact algorithm**: Unlike many scored-based methods, the algorithm does not make any constraints on the underlying distribution and is not subject to local optima

- **False-discovery rate control**: FDR control is added to provide a powerful tool to control the fraction of type 1 errors
- **Complexity trade-off**: The conditional independence test allows for trading off between power of the algorithm and its computational complexity, and thus tailoring it to the task at hand

Performance tests showed that the false-discovery rate is as advertised and the power of the test increases constantly for increasing sample sizes. Additionally, consistency under perturbations in the data set was demonstrated.
The algorithms have been provided in the pcit package giving potential users a simple way to test for conditional independence between sets of variables, as well as perform graphical model structure learning with little need for manual parameter tuning. The implementations are particularly interesting for users

- assessing the value of collecting additional variables for a prediction task
- performing exploratory data analysis
- looking for a visual way to communicate the structure in their data set

There are a few ways in which the current work can be generalized to make for a more powerful test. The power of the conditional independence test is directly linked to the power of the supervised learning algorithm to find the optimal prediction functional and the correct loss. Currently, only the necessary minimum of 2 loss functions is implemented, but this can be generalized by including more losses, and checking if baseline can be beaten with statistical significance in any of the losses. This would improve the argument when reasoning about variables being independent, when the null hypothesis cannot be rejected. What's more, the current methodology connected single-output prediction with FDR control. Alternatively, the feasibility of multi-output predictors, predicting several outputs at once, should be assessed, for appropriate multivariate loss functions.
For performance evaluation, the power of the proposed routine was assessed for two specific cases, for multivariate Gaussian data, and a simple conditional independence test. To get a better idea of the general feasibility of the algorithm, more scenarios need to be analyzed and performance tests conducted. Lastly, the power of the test in the context of alternative graphical model structure learning algorithms should be assessed.

A Optimal constant predictors

Since the following results are only used for univariate predictions, the elicitability is shown in the univariate case.

Squared Loss

Let $L(y, y^*) = (y - y^*)^2$ and let Y be a random variable with density function $p(Y)$. Then the expected generalization loss for predicting the constant t is

$$\begin{aligned}\mathbb{E}[L(Y,t)] &= \int_{-\infty}^{\infty} (y-t)^2 p(y) dy \\ &\propto -2t \int yp(y)dy + t^2 \int p(y)dy \quad \text{(where proportionality is w.r.t. t)} \\ &= t^2 - 2t\mathbb{E}[Y]\end{aligned}$$

The first and second derivatives are

$$\begin{aligned}\frac{\partial \mathbb{E}[L(Y,t)]}{\partial t} &= 2t - 2\mathbb{E}[Y] \\ \frac{\partial^2 \mathbb{E}[L(Y,t)]}{\partial t^2} &= 2\end{aligned}$$

The extremal point is at $t = E[Y]$, a minimum, since the second derivative is positive, assuming $E[Y]$ exists. By Definition 3, the mean is thus elicited by the squared loss function.

Logarithmic Loss

Let Y be a random variable with probability mass function $p(Y)$ taking a finite set of discrete values. Without loss of generality, let Y take values in $\{1, ..., q\}$. Since $p(Y)$ is a probability distribution, $\sum_{i=1}^{q} p(y = i) = 1$. Consider the logarithmic loss (also known as log loss or cross-entropy loss)

$$L(y, q) = -\sum_{i=1}^{q} log(q_i)\mathbb{I}(y = i)$$

Where $\mathbb{I}$ is an indicator function, 1 if $y = i$, 0 otherwise and q_i is the i-th element in the vector q.

The expected generalization error for predicting the constant (vector) t is then

$$\begin{aligned}\mathbb{E}(L(y,t)) &= -\sum_{j=1}^{q}[\sum_{i=1}^{q} log(t_i)\mathbb{I}(y=i)]p(y=j)\\ &= -\sum_{i=1}^{q} log(t_i)[\sum_{j=1}^{q}\mathbb{I}(y=i)]p(y=j)]\\ &= -\sum_{i=1}^{q} log(t_i)p(y=i)\end{aligned}$$

Differentiating with respect to t under the additional constraint that $\sum_{i=1}^{p} t_i = 1$, since the prediction is a probability,

$$L^*(y,t,\lambda) = -\sum_{i=1}^{q} log(t_i)p(y=i) - \lambda(1-\sum_{i=1}^{p} t_i)$$

The first derivatives are

$$\frac{\partial L^*(y,t,\lambda)}{\partial t_i} = -\frac{p(y=i)}{t_i} + \lambda \overset{!}{=} 0,\ \forall i \in \{1,...,p\}$$

$$\frac{\partial L^*(y,t,\lambda)}{\partial \lambda} = 1 - \sum_{i=1}^{p} t_i$$

Since the proportions $\frac{p(y=i)}{t_i}$ are constant and $\sum_{i=1}^{p} t_i = \sum_{i=1}^{p} p(y=i) = 1$, $t_i = p(y=i)$, $\forall i$. The second derivative is

$$\frac{\partial^2 L^*(y,t,\lambda)}{\partial t_i t_j} = \begin{cases}\frac{p(y=i)}{t_i^2} > 0 & (\forall i = j)\\ 0 & (\forall i \neq j)\end{cases}$$

which, by diagonal dominance, is a positive definite matrix, and the constant predictor minimizing the log loss is the one predicting the class probabilities.

References

C.C. Aggarwal. *Data Classification: Algorithms and Applications.* Chapman & Hall/CRC, 1st edition, 2014.

D. Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

L. Baringhaus and C. Franz. On a new multivariate two-sample test. *Journal of Multivariate Analysis*, 88:190–206, 2004.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57:289–300, 1995.

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29:1165–1188, 2001.

W. Bergsma. Nonparametric testing of conditional independence by means of the partial copula. *ArXiv e-prints: 1101.4607*, 2011.

W.P. Bergsma. *Testing conditional independence for continuous random variables.* Eurandom, 2004.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. API design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

U. Cherubini, E. Luciano, and W. Vecchiato. *Copula methods in finance.* John Wiley & Sons, 2004.

K.P. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. Fast two-sample testing with analytic representations of probability measures. In *Advances in Neural Information Processing Systems 28*, pages 1981–1989. Nips, 2015.

G.W. Corder and D.I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach.* John Wiley & Sons, Inc., 2009.

R. Diestel. *Graph Theory.* Springer, 5th edition, 2016.

A. Dionisio, R Menezes, and D.A. Mendes. Entropy-based independence test. *Nonlinear Dynamics*, 44:351–357, 2006.

V.A. Fernández, M.D.J. Gamero, and J.M. García. A test for the two-sample problem based on empirical characteristic functions. *Computational Statistics & Data Analysis*, 52:3730–3748, 2008.

T.M. Fragoso and F.L. Neto. Bayesian model averaging: A systematic review and conceptual classification. *arXiv preprint arXiv:1509.08864*, 2015.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

C. Genest and B. Rémillard. Test of independence and randomness based on the empirical copula process. *Test*, 13:335–369, 2004.

T. Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106:746–762, 2011.

A. Gretton, O. Bousquet, A. Smola, and B. Scholkopf. Measuring statistical dependence with hilbert-schmidt norms. In *ALT*, volume 16, pages 63–78. Springer, 2005.

A. Gretton, K. Fukumizu, C.H. Teo, L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In *Advances in neural information processing systems*, pages 585–592, 2008.

A Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012a.

A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B.K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems 25*, pages 1205–1213. Nips, 2012b.

A.A. Hagberg, D.A. Schult, and P.J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, 2008.

W. Hoeffding. A non-parametric test of independence. *The Annals of Mathematical Statistics*, 19:546–557, 1948.

E. Jones, T. Oliphant, and P. Peterson. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

D. Lopez-Paz and M. Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Y. Nadeau, C.and Bengio. Inference for the generalization error. *Machine Learning*, 52:239–281, 2003.

OECD. OECD statistics: Key short-term economic indicators. `http://stats.oecd.org/`, 2017. Accessed: 2017-08-06.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

S. Raschka. Mlxtend, 2016. URL `http://dx.doi.org/10.5281/zenodo.594432`.

B. Rémillard and O. Scaillet. Testing for equality between two copulas. *Journal of Multivariate Analysis*, 100:377–386, 2009.

B. Schweizer and E. F. Wolff. On nonparametric measures of dependence for random variables. *The Annals of Statistics*, 9:879–885, 1981.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. *arXiv preprint arXiv:1202.3775*, 2012.