Proceedings of the 4th International Conference on Statistics: Theory and Applications (ICSTA'22)
Prague, Czech Republic – July 28- 30, 2022
Paper No. 113
DOI: 10.11159/icsta22.113

A Structural Learning Method for Graphical Models

Benjamin Szili[1], Mu Niu[2], Tereza Neocleous[2]
[1]University of Glasgow
Glasgow, United Kingdom
b.szili.1@research.gla.ac.uk; Mu.Niu@gla.ac.uk, Tereza.Neocleous@gla.ac.uk,
[2]University of Glasgow
Glasgow, United Kingdom

Abstract – This work is centred on investigating dependencies and representing learned structures as graphs. While there are a number of methods available for discrete and Gaussian random variables, there is no such method readily available for continuous variables that are non-Gaussian. For such methods to be reliable, it is necessary to have a way to measure pairwise and more importantly, conditional independence. In this work, an algorithm is created that uses both mutual information and a kernel method together to account for these dependencies and yield a graph that represents them. This method is then demonstrated through a simulation setting, comparing the results to an algorithm often used in Gaussian settings, additionally discussing future steps regarding this project.

Keywords: structural learning, mutual information, kernel methods, independence

1. Introduction

Structural learning is part of the process of fitting a Bayesian Network (BN) [1] - representing probabilistic dependencies between a set of random variables. Essentially, it is a model selection process to find a graph structure that is appropriate for the set of variables. One of the most common well-known methods is the PC algorithm [2], which performs very well in Gaussian settings. It is of interest whether a network structure can be learned when the random variables are continuous, but not linearly dependant on their parents - for example, the dependence between a variable and its parents is in its variance-, or potentially not normally distributed. When encountering continuous variables that cannot be included in a Gaussian BN, the current general approach is discretization, so that they may be included in a Discrete BN, as this class of networks has structural learning algorithms available. The aim of this work is to develop some methodologies that does not require discretization of such continuous variables to learn the network structure.

The main difficulty in this endeavour is that while there appear to several methods to establish pairwise independence, in order to represent the structure of dependencies it is also necessary to obtain a reliable way to assess conditional independence between non-Gaussian continuous variables – which there appears to be a lack of.

The objective therefore is to enable a learning algorithm to recognise relationships that are not between discrete or Gaussian variables, and to dependencies that are not necessarily between the means. In order to do so, it is first required to find a metric that can find such relationships between variables in cost-effective and accurate manner. For this, Mutual Information and a branch of kernel methods were examined. The next step is to extend it to learn conditional dependencies such that it is still effective, which was done by using Conditional Mutual Information. Then, it should be possible to include in a structural learning algorithm, either as a score in a score based, or as part of a test in a constraint-based algorithm. This was done by enabling both an estimate for Mutual Information and a kernel method to establish pairwise independence, then assessing conditional independence via Conditional Mutual Information estimates. Finally, a simulation setup was used to compare this method with the PC-algorithm, which is widely used to learn networks with Gaussian data.

2. Methodology

2.1 Review on Information Theory

Mutual Information (MI) is a fairly common information-theoretic dependence measure for discrete variables. Given **X** and **Y**, it can be written as:

$$I(\boldsymbol{X};\boldsymbol{Y}) = H(\boldsymbol{X}) - H(\boldsymbol{X}|\boldsymbol{Y}) \quad (1)$$

where H(**X**) is the marginal entropy of **X** and H(**X**|**Y**) is the conditional entropy of **X** given **Y**.

Therefore, mutual information can be interpreted as the reduction in uncertainty about **X** after observing **Y**. Important properties of mutual information are symmetry and non-negativity. It follows then that I(**X;Y**) = 0 if and only if **X** and **Y** are independent (as this is achieved when H(**X**) = H(**X**|**Y**), meaning that observing **Y** does not change the amount of information needed to describe **X**). In addition, once MI has been calculated, it is then also possible to determine the Conditional Mutual Information (CMI), the expected mutual information between random variables **X** and **Y**, given that **Z** has been observed:

$$I(\boldsymbol{X};\boldsymbol{Y}|\boldsymbol{Z}) = H(\boldsymbol{X},\boldsymbol{Z}) + H(\boldsymbol{Y},\boldsymbol{Z}) - H(\boldsymbol{Z}) - H(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}) \quad (2)$$

The use of CMI then appears to be a potentially useful measure of dependence that does not require **X**, **Y** and **Z** to be normally distributed. However, unlike the discrete case, the calculation for continuous variables is less straightforward, as the differential entropy needed to estimate mutual information can become too complex to calculate for non-Gaussian distributions. Therefore, instead of using such calculations, it is of interest to find a reliable and efficient way to estimate Mutual Information through approximation.

2.2 Approximating Mutual Information

A potential way to obtain an estimate of mutual information between two continuous variables is using a nearest neighbour approximation [3]. A popular approach is the use of some version of the non-parametric Kozachenko-Leonenko estimator for entropy [4]. The approximation essentially calculates the average distance from k-th nearest neighbour in the joint space of **X** and **Y**, then compares it to the average number of points within the same distance in the projected marginal spaces, denoting the number of points by n_x, n_y for fixed x. Finally, we can then obtain an estimate for mutual information

$$\hat{I}(\boldsymbol{X};\boldsymbol{Y}) = \psi(k) + \psi(n) - \langle(\psi(n_x + 1) + (\psi(n_y + 1)]\rangle \quad (3)$$

where $\psi(.)$ is the digamma function and $z_i=(x_i,y_i)$.

A property of this estimate is that much like MI, it can also be extended to approximate conditional mutual information between continuous variables [3]

$$\hat{I}(\boldsymbol{X};\boldsymbol{Y}|\boldsymbol{Z}) = \psi(k) - \langle\psi[n_{xz}(i)] + \psi[n_{yz}(i)] - \psi[n_z(i)]\rangle \quad (4)$$

which makes the approximation worth considering as a potential independence measurement for a learning algorithm.

It then appears that the approximations for pairwise and conditional mutual information is fairly straightforward and could be incorporated it into a structural learning algorithm.

2.3 Kernel covariance as a measure of independence

The kernel covariance itself is defined given Reproducing Kernel Hilbert Spaces (RKHS) $\mathcal{F}_\mathcal{X}, \mathcal{F}_\mathcal{Y}$, along with associated kernels $\mathrm{k}(\mathrm{x_i} - \mathrm{x_j}), \mathrm{l}(\mathrm{y_i} - \mathrm{y_j})$ as

$$\mathcal{J}(\mathrm{P_{x,y}}, \mathcal{F}_\mathcal{X}, \mathcal{F}_\mathcal{Y}) = \sup_{\mathrm{f}\in\widetilde{\mathcal{F}_\mathcal{X}}, \mathrm{g}\in\widetilde{\mathcal{F}_\mathcal{Y}}} \left|\mathrm{E_{x,y}}[\mathrm{f(x)g(y)}] - \mathrm{E_x}[\mathrm{f(x)}]\mathrm{E_y}[\mathrm{g(y)}]\right| \quad (5)$$

Where $\widetilde{\mathcal{F}_\mathcal{X}} \coloneqq \{\mathrm{f} \in \mathcal{F}_\mathcal{X} : |\mathrm{f}|_{\mathcal{F}_\mathcal{X}} \le 1\}, \widetilde{\mathcal{F}_\mathcal{Y}} \coloneqq \{\mathrm{g} \in \mathcal{F}_\mathcal{Y} : |\mathrm{g}|_{\mathcal{F}_\mathcal{Y}} \le 1\}$ (6)

Then, given $f \in \sim\mathcal{F}_\mathcal{X}, g \in \sim\mathcal{F}_\mathcal{Y}$ on the bounded sets $\mathcal{X} \subset R^k, \mathcal{Y} \subset R^l$, the kernel correlation is zero if and only if **x** and **y** are independent.

A technical report [5] then generalizes kernel covariance to higher dimensions, which can become very useful for structural learning. However, in higher dimensions kernel covariance only extends to pairwise independence, while ultimately, conditional independence is the goal in order to incorporate it into a learning algorithm. Nevertheless, kernel covariance could potentially be useful as a foundation.

2.4 Constrained Covariance and the Hilbert-Schmidt Independence Criterion

In later work [6] Kernel Covariance has been reworked as Constrained Covariance (COCO), given function classes *F*,*G* in $\mathcal{F}_{\mathcal{X}}, \mathcal{F}_{\mathcal{Y}}$ and probability measure $P_{x,y}$

$$\mathrm{COCO}(\mathrm{P}_{x,y}; \mathcal{F}, \mathcal{G}) = \sup_{f \in \mathcal{F}, g \in \mathcal{G}} [\mathrm{cov}(f(x), g(y))] \tag{7}$$

Given independent observations $\mathbf{z} = (x_1, y_1), \dots, (x_n, y_n)$, and *F*, *G* being unit balls in their vector spaces, an empirical estimate can also be obtained.

To obtain a more robust independence measure, the entire spectrum of the cross-covariance operator can be used [7], giving the sum of squared singular values of the cross-covariance operator, which was named the Hilbert-Schmidt Independence Criterion (HSIC)

$$\mathrm{HSIC}(\mathrm{P}_{x,y}; \mathcal{F}, \mathcal{G}) = \left|\left| \mathrm{E}_{x,y}[f(x)g(y)] - \mathrm{E}_x[f(x)]\mathrm{E}_y[g(y)] \right|\right|_{\mathrm{HS}}^2 \tag{8}$$

where $|| \cdot ||_{\mathrm{HS}}^2$ is the squared Hilbert-Schmidt norm. This version of the method has reduced bias, and the empirical estimates are computationally cheaper, as one can obtain an estimate simply using the trace of centred Gram matrices **K** and **L**

$$\mathrm{HSIC}(Z, \mathcal{F}, \mathcal{G}) = \frac{1}{n^2} \mathrm{tr}\mathbf{KL} \tag{9}$$

Where **K** is the outer product of **x** and **L** is the outer product of **y**. Therefore, while a conditional version of HSIC is not generally available, it appears that some variant of these kernel methods can measure independence.

2.5 The algorithm

The above blocks to assess pairwise independence and a way to obtain and estimate of a measure that can indicate conditional independence to some extent were then used together. In order to reflect the dependencies in an overall graph structure, the initial algorithm was developed and is shown in Table 1:

Table 1: Pseudo-code for the algorithm to establish graph structure

1.Take random variables X_i and assign set of vertices V_i representing them, specify k and a threshold value
2. For each pair (X_i,X_j), calculate MI estimate $m_{ij}=\widehat{MI}(X_i, X_j)$, where i≠j
3. If $m_{ij} = \widehat{MI}(X_i, X_j)$ is greater than the threshold value, assign edge e_{ij} between V_i and V_j
4. Obtain cliques of order three.
5. For each of these cliques, using the triplet (X_i, X_j, X_k), calculate and sort CMI estimates $c_{ijk} = \widehat{\mathrm{CMI}}(X_i, X_j, X_k) = \hat{I}(X_i; X_j \| X_k)$ as well as $c_{ikj}$and c_{jki}, where i ≠ j ≠ k
6. If the lowest CMI estimate $\hat{I}(X_i; X_j \| X_k)$ is close to zero, remove edge between V_i and V_j.

Following simulations using a threshold for MI, it was of interest to see the effect of using a formal test for independence. While conditional independence tests are not widely available for non-linear distributions, a pairwise test using the Hilbert-Schimdt Independence Criterion (HSIC) was used [8]. While the results are slightly more conservative, the test indicating independence generally agrees with results obtained using a threshold for MI and has higher calculation speed. Therefore, the edges to be removed were decided by checking pairwise independence using this the test instead of

MI (although the option to calculate pairwise MI was retained), while using CMI to remove edges related to conditional independence, using an initial threshold of 0.25.

3. Results

3.1 The simulation setup

The Bellot setup [9] is being used for the investigation of the algorithm as follows: using the post non-linear noise model, the structure is defined under two hypotheses: a common cause and a common effect.

$$H_0: X = f(A_f Z + \epsilon_f), Y = g\left(A_g Z + \epsilon_g\right); H_1: Y = h(A_h Z + X + \epsilon_h) \quad (10)$$

The functions *f, g, h* are specified as part of the setup, as well as the distributions of X, Y, Z, effectively enforcing a specific structure.

The matrix dimensions of A are such that X,Y are univariate and the noise variables $\epsilon_f, \epsilon_g, \epsilon_h$ are 0 on average with variance 0.025. The distributions of X,Y,Z , and the complexity of dependencies via *f, g* and *h* can then be tuned to make performance comparisons in different settings. The original settings are Multivariate Gaussian - where *f, g* and *h* were fixed to be the identity functions - , and then some arbitrary distributions by randomly sampling *f, g* and *h* from $[x^3, \tanh x, \exp(-x)]$, while sampling Z from a Gaussian distribution. The sample size for each of these cases was initially set to 100 observations, each containing a realization from Z, as well as X and Y under H_0 (X0,Y0) and under H_1 (X1,Y1) . Then ideally, our enforced structure would look like Figure 1

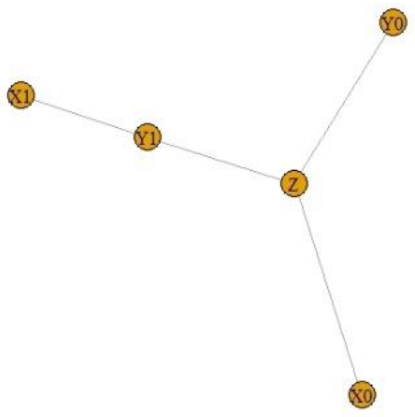

Figure 1: Ideal structure enforced by the setup

As comparison to the output of the algorithm, the PC-algorithm is used on each configuration as well.

3.2 Multivariate Gaussian trials

For this part of the setup, the functions *f, g* and *h* are set to be the identity function. We set

$$Z \sim N(0,1), X_1 \sim N(0,1), \epsilon_{(.)} \sim N(0,0.025) \quad (11)$$

Then, we have

$$X_0 = Z + \epsilon_f, Y_0 = Z + \epsilon_g, Y_1 = Z + X_1 + \epsilon_h \quad (12)$$

Using a sample size of 100, Figure 2 shows the structure learned by the PC algorithm. Then, Figure 3 shows the learned structure by using the new method.

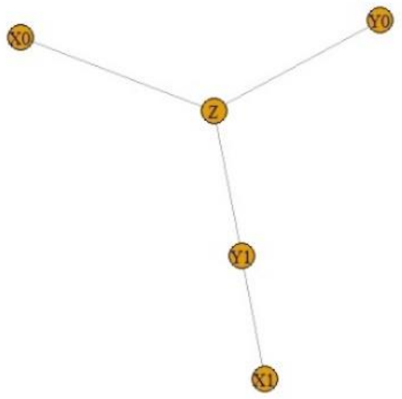

Figure 2: Structure learned by PC-algorithm

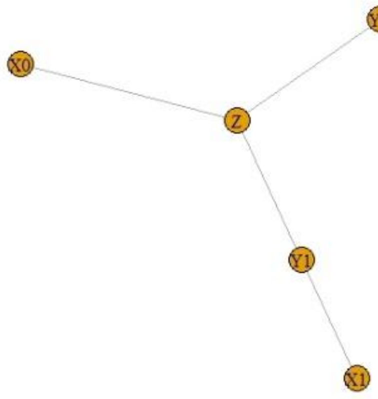

Figure 3: Structure learned by the new method

As expected, since the PC-algorithm is often used on Gaussian variables, it successfully identified the structure enforced by the setup. However, the new method also managed to learn the same structure. It is therefore of interest to move on to move on to more complex relationships.

3.3 Trials with arbitrary, more complex relationships

In the next configuration we still have $\epsilon_{(.)} \sim \mathrm{N}(0,0.025)$. We return to sample Z and X_1 from (11) and so we still have (12).

The main difference in this section is that f, g and h are selected from $[x^3, \tanh x\,, exp(-x)]$. For demonstration, Figures 4 and 5 show results for the PC-algorithm and the new method, when f, g and h are fixed to be $\tanh$ x.

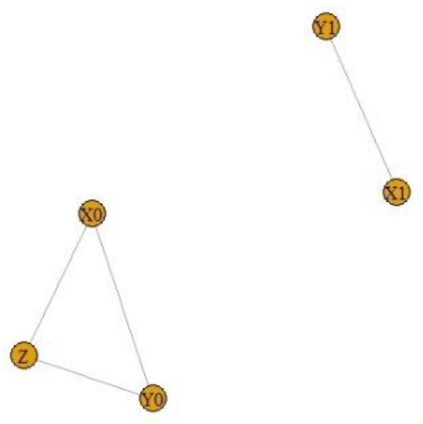

Figure 4: Structure learned by PC-algorithm

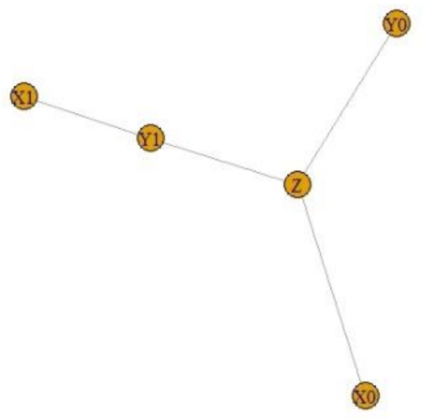

Figure 5: Structure learned by the new method

In this case, the PC-algorithm was not able to establish the conditional independence of X_0 and Y_0 given Z and missed the link between Z and Y_1. On the other hand, the new algorithm did manage to learn the structure enforced by the setup.

As a further step the algorithm is also run ten times in succession, using different samples from the same distributions. In each instance, the remaining edges were recorded, then an overall graph would be created, indicating the number of times an edge remained. While this often does not give a notion of each individual structure, it does highlight overall patterns across trials using different configurations for functions *f*, *g* and *h*. In addition, different variances for noise variables $\epsilon_{(.)}$ are also examined.

4. Conclusion

This work was aimed to develop a method for structural learning that can be used for continuous variables that are non-Gaussian. The HSIC independence test along with the CMI approximation was used to develop this algorithm, creating graphs that reflect the observed dependencies. While the new method does not always completely identify the structure enforced, apart from the Gaussian settings, it appears to find the links between variables and indicate conditional independence more frequently than the PC-algorithm, producing more accurate results on the simulation configurations in general.

For future steps, it is crucial to extend the scope that the method is tested on, using different distributions, non-linear relationships, including real data as well. In addition, it is of interest to use a metric assessing how well the method performs by assigning weights to edges.

Acknowledgements

Many thanks to the School of Mathematics & Statistics at the University of Glasgow for the funding of this research. Mu Niu acknowledges support for this paper from EPSRC grants EP/W021595/1.

References

[1] R Nagarajan, M Scutari S. Lebre (2013) Bayesian Networks in R: With Applications in Systems Biology. Springer.
[2] P Spirtes, C Gylmour, R Scheines (1991). Causal inference. Erkenntnis (35) :151-189
[3] A Tsimpiris, I Vlachos, D Kugiumtzis (2012). Nearest neighbor estimate of conditional mutual information in feature selection. Expert System with Applications (39):12697-12708
[4] A Kraskov, H Stögbauer, and P Grassberger (2004), Estimating mutual information. Phys. Rev. E 69, 066138.
[5] A Gretton, R Herbrich, and A Smola (2003). The Kernel Mutual
Information. Max Planck Institute for Biological Cybernetics
[6] A Gretton, R Herbrich, A Smola, O Bousquet, B Schölkopf (2005) Kernel methods for measuring independence. Journal of Machine Learning Research (6):2075-2129.
[7] A Gretton, K Fukumizu, C H Teo, L Song, B Schölkopf, A Smola (2008) A kernel statistical test of independence. In Advances in Neural Information Processing Systems 20 (NIPS).
[8] J Peters, J M Mooij, D Janzing, B Schölkopf (2014) Causal Discovery with Continuous Additive Noise Models. Journal of Machine Learning Research (15):2009-2053.
[9] A Bellot, M Schaar (2019). Conditional Independence Testing using Generative Adversarial Networks. NeurIPS