The Interdiciplinary Center, Herzliya
Efi Arazi School of Computer Science
M.Sc. Program - Research Track

Discovering Discrete Hidden Variables in Mixed Networks: A Statistical Hypothesis-Testing Approach

by
Aviv Peled

M.Sc. dissertation, submitted in partial fulfillment of the requirements
for the M.Sc. degree, research track, School of Computer Science
The Interdisciplinary Center, Herzliya

September, 2020

This work was carried out under the supervision of Dr. Shai Fine from the Data Science Institute, The Interdiciplinary Center, Herzliya.

Acknowledgements

I would like to thank my supervisor Dr. Shai Fine for guiding me through this process. Thank you for many hours of discussions, many hours of late-night work and for encouraging and empowering me. I would not have been able to do it without your help.

I would also like to thank Tali, my soon to be wife, my family and my friends for encouraging me and supporting me through the emotional roller coaster of these couple of years.

Abstract

Latent variables pose a challenge for accurate modelling, experimental design, and inference, since they may cause non-adjustable bias in the estimation of effects. While most of the research regarding latent variables revolves around accounting for their presence and learning how they interact with other variables in the experiment, their bare existence is assumed to be deduced based on domain expertise. In this work we focus on the discovery of such latent variables, utilizing statistical hypothesis testing methods and Bayesian Networks learning. Specifically, we present a novel method for detecting discrete latent factors which affect continuous observed outcomes, in mixed discrete/continuous observed data, and device a structure learning algorithm that adds the detected latent factors to a fully observed Bayesian Network. Finally, we demonstrate the utility of our method with a set of experiments, in both controlled and real-life settings

Contents

1 Introduction

1.1 Motivation

Consider a researcher attempting to assess the effectiveness of a drug X from a population data, where drug usage is the patient's choice. Furthermore, consider a scenario where gender (Z) differences influence the patient's choice to use the drug (i.e. compliance) as well as their chances of recovery (Y). In this scenario, Z (gender) confounds the relation between X and Y, since Z affects both X and Y. The effect of a confounder on an experiment is that it may lead to biased estimates. For example (c.f Figure 1), if the drug has a better effect on women than it has on men, and the test group consists mostly of women and only a handful of men, the statistical test may falsely indicate that the drug is more effective than it really is.

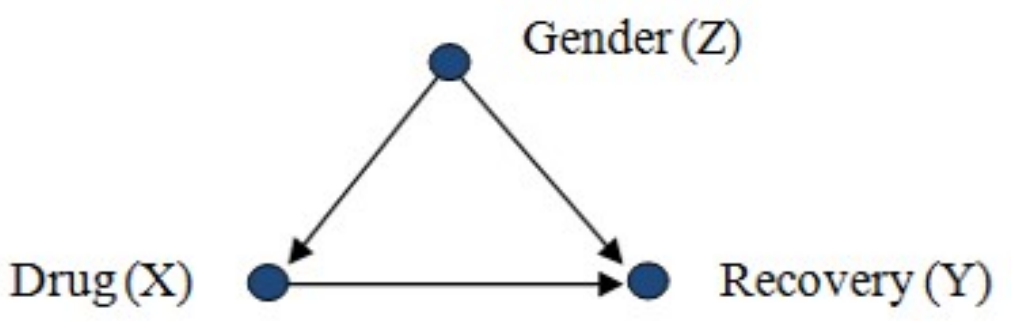


Figure 1: The gender variable is a confounder in the drug experiment

Another interesting effect a latent variable may have is when the relation between the latent confounding factor and the observed explanatory variable is reversed. For example, a clinical study aimed at testing the efficacy of a new painkiller drug, which has a rare paradoxical side effect (such as headaches, mus-

cle aches, and nausea) that interferes with pain levels assessed in patients as part of the clinical study (c.f. Figure 2). Discovering such an anomaly, modelling it and accounting for it, would lead to a more accurate estimate of the drug's effectiveness, as well as a better model that suggests explanations and insights into side effects, adverse events, and possibly also drug-drug-interaction (DDI), if the latent factor is correlated with additional drug administration.

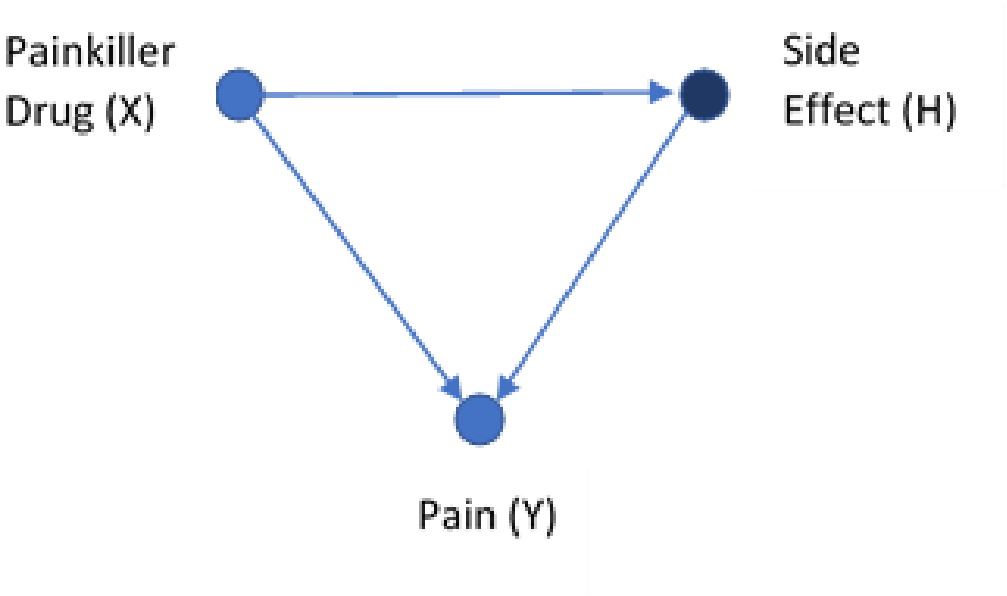


Figure 2: The side effect latent variable is effected by the painkiller drug administration

Thus, confounding factors are an important issue in experimental design, and researchers invest many efforts in controlling for all known confounding factors, e.g. to reduce selection bias. However, once we observe the outcome of an experiment, how can we be sure that there are no additional factors that we did not

include in our design which may effect the observed outcomes?

The problem of detecting and accounting for latent effects is manifested in many domains and real-life settings. Another notable example is in social networks analysis, where researchers have been trying to figure out a way to measure "Social Contagion" or "Social Influence" - The transmission or transfer of deviant behavior from one person to another. A well studied implication is the design of a marketing campaign: In a *seeding campaign*, a new product is provided in a discount price (or free of charge) to influential *key opinion leaders* (KOL) that hopefully will recommend the product, thus contributing to a positive sentiment and increased sales. In order to have a more effective campaign, a seeding strategy (namely to which KOL should the product be provided) must be devised. Such a strategy usually employs a social influence measurement, by which individuals are ranked, simulations of the diffusion process generated by "gifting" the top influencers, and predictive models to assess how much value will this seeding campaign produce. Social influence or social contagion is often measured by a statistical hypothesis testing - The test group consists of people who own the product and the control group consists of people who do not own that product. Then, the group of friends of the people in the test group and of the people in the control group are examined to see if they buy the product (or present the contagious behavior in question). If we see a difference in the number of friends who buy the product between the test and the control group (i.e. the effect size), we may conclude that social influence is responsible for that. But here is where we meet the latent confounders in the form of homophily, which is the tendency of individuals

to associate and bond with similar others, or as the known saying goes “Birds of a feather flock together”. Homophily provides an alternative causal explanation - Maybe the friends of the people who bought the product (the test group) had a higher chance to buy that product to begin with, since they all like the same products, and this is why they are friends. Figure 3 demonstrates this phenomenon with a concrete example: Ian and Joey are friends and also gadget enthusiasts. Ian bought the newest smartphone model, and soon after, Joey bought the same smartphone. Is it because Ian influenced Joey, or maybe Ian was just quicker and Joey would have bought the smartphone anyway?

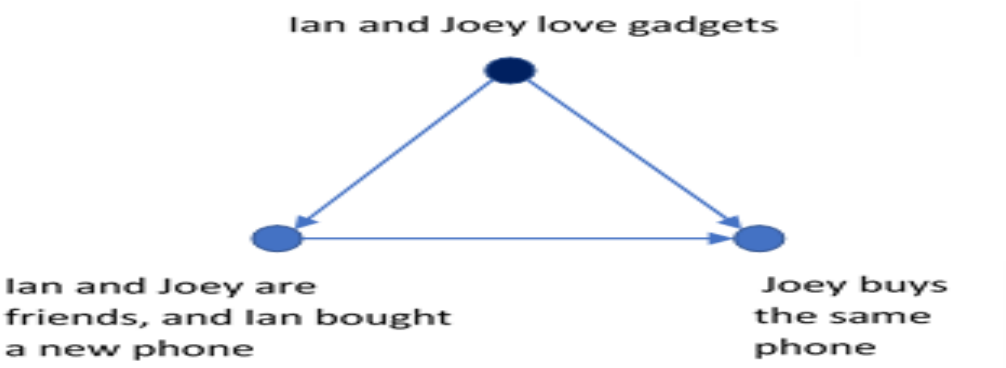


Figure 3: Homophily - Did Ian influence Joey, or maybe Ian was just quicker and Joey would have bought the smartphone anyway?

In this work we focus on the discovery of such latent variables that can provide a better explanation to the observed outcomes. In the current study we focus attention on settings where the observed cause and the latent variable are both discrete, and the observed effect is continuous.

Our solution scheme follows: Given the data, we start by devising a Bayesian

Network to model the relations between the observed variables. Next, we pose the detection of additional latent factors as a hypothesis testing problem, and use Hartigans' *dip test* (Hartigan and Hartigan, 1985) to decide whether or not there's enough evidence to suggest that latent variables exist. Finally, we provide an algorithm that uses d-separation to decide if and how to enhance the network structure with latent variables, what will their cardinalities be, and how to connect the added latent variables to the existing network.

The rest of the dissertation is organized as follows: In Section 2 we give some definitions and essential information on Hartigans' *dip test*, on Bayesian Networks, and on Conditional Gaussian Bayesian Networks. Section 3 reviews some related works. In section 4.1 we describe our dip test based approach for detecting latent variables. This section includes (controlled) demonstration for the detection of variables in simulated and real-life settings. Section 4.2 presents our structure learning scheme. Section 5 describes the results of applying the complete process, namely latent variables discovery and structure learning, in a series of experiments. We conclude in Section 6 with a few insights and an open question for future research.

2 Background

2.1 Statistical Hypothesis Testing

In statistics, hypothesis testing is a method for testing a claim or hypothesis about a population, using data measured in a sample assumed to be drawn from that population. In hypothesis testing a researcher begins by stating two mutually exclusive and exhaustive statistical hypotheses, the null H_0 and the alternative H_1. The alternative hypothesis typically reflects the researcher's guess or prediction which requires testing, while the null hypothesis is the negation of the alternative hypothesis, that is assumed to be true until proved otherwise. The researcher now performs a procedure which involves measuring how surprising the value of an observed test statistic (which is relevant to the hypotheses) is when H_0 is assumed to be true. This assessment is carried out by calculating a probability, called a p-value, so that small values of the p-value indicate that the observed test statistic value is surprising. The researcher needs to set the level of significance α for the test, that is the criterion upon which a decision is made regarding the null hypothesis. To make a decision, the p-value is compared to the level of significance, and if it is lower than it, the null hypothesis is rejected with significance.

For example, take a medical researcher who is claiming that the mean weight of Israeli men aged between 30 and 39 has increased since the 80s. Based on official data from that period, the mean weight for Israeli men of the age group in question was $\mu = 77kg$ with standard deviation $\sigma = 19kg$. To prove his claim, the researcher devices a hypothesis test as follows: He states the null hypothesis based

on the official records $H_0 : \mu = 77kg$ which implies that there has been no change in the mean weight over time, and the alternative hypothesis (which he is trying to prove) $H_1 : \mu > 77kg$ (the mean weight has increased). The researcher then sets a significance level 0.05, and decides to make use of an appropriate statistical hypothesis test based on the problem setting - the one sided Z-test, which is applicable for normally distributed data, and "greater than" claims. The researcher collects a sample $\bar{x}$ of $n = 64$ weight records, computes the sample mean $\mu_{\bar{x}} = 83kg$, and uses the known σ and n to compute the standard error $s_{\bar{x}} = \frac{\sigma}{\sqrt{n}} = 2.375$, finally computing the *Z-statistic* $Z = \frac{\mu_x - \mu_0}{s_x} = \frac{83-77}{2.375} = 2.652$ which is the normalized sample mean assuming H_0 was true. To complete the test, the researcher now has to make a decision based on the significance level he set earlier. For that, he computes the *p-value* - the likelihood of the Z-statistic given that the null hypothesis H_0 was true, that is the area under the curve of a normal distribution beyond the value of Z (one sided test) which for $Z = 2.652$ is 0.109 (see Fig 4). Since the p-value is larger than the significance level ($0.109 > 0.05$) the researcher accepts the null hypothesis H_0 since he does not have enough evidence to reject it, which sadly for him means that the increased weight he witnessed in his test is probably attributed to sampling, and not to a change in the population parameters.

2.1.1 Dip Test

In our work, we make use of a specific statistical hypothesis test named *"Dip Test"* (Hartigan and Hartigan, 1985) which allows us to decide whether a sample in question is from a uni-modal distribution or not. The test's hypotheses

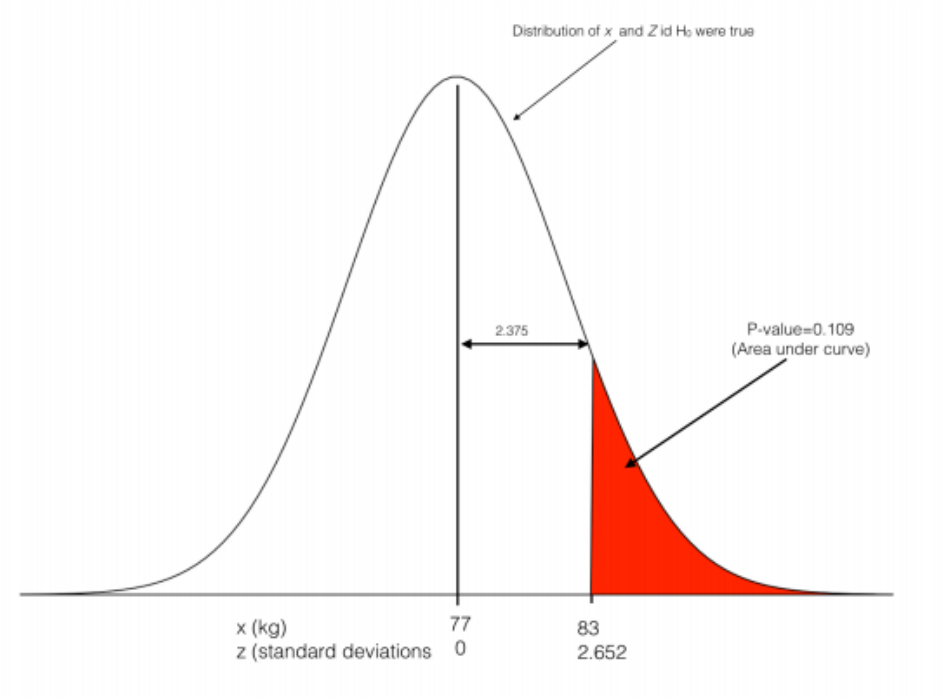


Figure 4: The p-value of the statistic in the one-sided test is the area under the curve beyond the value of the statistic

are: H_0: *The sample was drawn from a uni-modal distribution* and H_1: *The sample was drawn from a multi-modal distribution*. The Dip test statistic is the maximum difference between the observed distribution of the data and a uni-modal distribution that is chosen to minimize this maximum difference: $D(F) = min_{u \in U} max_x |F(x) - u(x)|$ where U is the set of all uni-modal distributions. In his paper, Hartigan suggests an efficient method of computing his test statistic, linear in the size of the input data. Hartigan provided an empirical table with probabilities for different dip test statistic values based on repetitive computation of the test statistic on several samples with varying sample sizes from the uniform

distribution. Based on the table, it is possible to compute p-values for any Dip test statistic using interpolation. Hartigan chose the uniform distribution as the reference null distribution in his power calculations, since, in his words, it is the least favorable uni-modal distribution, considering its Dip test statistic values are stochastically larger than other unimodal distributions, such as those having exponentially decreasing tails. Hartigan suggested to use a significance level 0.05 for his multi-modality test, that is if the p-value for the dip statistic is lower than 0.05, there is enough evidence to reject the null hypothesis and conjecture that the sample is from a multi-modal distribution with high significance, otherwise, we accept the null hypothesis and say that the sample is from a uni-modal distribution.

The Dip test has many advantages: It is non-parametric and works with all kinds of uni-modal distributions such as Gaussian, beta, uniform and so on. It is invariant to scaling and shifting, robust to noise, and finally, it is deterministic and fast to compute.

The Dip test is used a lot as a statistical tool in many life sciences works, such as (Freeman and Dale, 2013) that make use of the test to detect multi-modality in psychology data, for research on dual-cognitive processes, or (Fulton et al., 2017) which used dip test to detect a multi-modality in radius measurements of planets, collected from the California-Kepler Survey. While not wide-spread in the AI community, an example of its usefulness can be seen in (Maurus and Plant, 2016), that presents a noise invariant clustering method based on the Dip test.

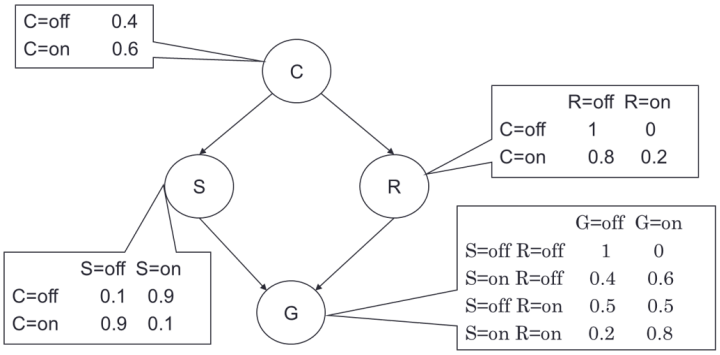
2.2 Bayesian Networks

Bayesian networks (BNs) are probabilistic graphical models that represent a set of variables and their conditional (in)dependencies using a directed acyclic graph (DAG). Their generative nature allows using them for a wide spectrum of modelling problems, such as prediction, classification, diagnosis, etc. In our work we rely heavily on Bayesian networks for our modelling and reasoning purposes. We start by describing the dependencies in the data with a Bayesian network, we then detect the existence of latent variables in the network, and finally we present the new variables and their interactions with the observed data in a Bayesian network.

2.2.1 Definition

The structure of a Bayesian network is defined by two sets: the set of vertices V and the set of directed edges E. The nodes represent random variables and the edges represent direct dependence among the variables, such that an edge from node X_i to node X_j represents a statistical dependence between the corresponding variables and informally implies that X_i "influences" X_j. From graph theory, X_i is the parent of X_j (and X_j is the child of X_i). The parent-child terminology allows the definition of sets of "descendants" and "ancestors" for the different nodes in the network's graph. The Bayesian network's graph should be acyclic, making it a directed acyclic graph (DAG). Other than the structural component of the model (the DAG), the quantitive parameters of the Bayesian network should also be stated: the conditional probability distributions (CPDs) for each node, given all

of its parents in the network's structure. Using the DAG structure and the aforementioned CPDs, the Bayesian network reflects a simple and very useful conditional independence statement - each variable is independent of its non-descendants in the graph given its parents. This property is called the local Markov property and is used to reduce the number of parameters that are required to characterize the joint probability distribution of the variables via factorization (Pearl, 1988). This is one of the most important qualities of the Bayesian network since it provides an efficient way to perform Bayesian inference on the different variables. More



R	C	S	G	P(C,S,R,G)
off	off	off	off	0.04
off	off	off	on	0
off	off	on	off	0.144
off	off	on	on	0.216
off	on	off	off	0.432
off	on	off	on	0
off	on	on	off	0.0192
off	on	on	on	0.0288
on	off	off	off	0
on	off	off	on	0
on	off	on	off	0
on	off	on	on	0
on	on	off	off	0.054
on	on	off	on	0.054
on	on	on	off	0.0024
on	on	on	on	0.0096

Figure 5: Left: An example of a Bayesian network including its graph structure and parameters. Right: The joint distribution encoded in the network without factorization requires more parameters.

formally, a Bayesian network $B = (G, \theta)$, where $G = (V, E)$ is a DAG, represents a joint probability distribution P over the set of random variables V using the CPDs encoded in θ and the conditional independence statements encoded in E, such that each variable X_i is independent of its non-descendants given its parents in G. θ contains the set of CPDs $\theta_i = P(X_i|\pi_{X_i})$ where π_{X_i} are the parents of the node X_i in G. The joint probability distribution P as described by the Bayesian network B

is factorized as such: $P(X_1, X2, ..., Xn) = \prod_{i=1}^{n} P(X_i|\pi_{X_i}) = \prod_{i=1}^{n} \theta_i$. (Pearl, 1988; Friedman et al., 1997)

2.2.2 Conditional Independence and d-separation

Conditional independence between variables is very useful as it allows knowledge discovery, better reasoning, and as we've seen before, the use of simpler, more efficient models since there are less dependencies to model. The Markov property stated above says that in a Bayesian network, a node is conditionally independent of all its non-descendants given its parents, but are these all of the conditional independence statements we can make about the variables in the network? The answer is no. Expanding on the Markov property stated above and using local conditional independence rules, we can define *d-separation* - a practical graphical criteria which allows easy exploration of the different conditional independence statements encoded in a Bayesian network. Let us explore the concept of d-separation.

Definition 2.1. Conditional Independence

Let P be a joint probability distribution over the set of random variables V. Let $a, b \in V$ be random variables and $A, B, C \subseteq V$ be sets of random variables

1. Variables a and b are conditionally independent given C if $P(a, b|C) = P(a|C)P(b|C)$.

2. A is conditionally independent of B given C if for every $a \in A$ and $b \in B$, a and b are conditionally independent given C. We will notate it as

$Ind(A,B|C)$.

Suppose we have a DAG $G = (V,E)$, and a set of nodes $\{X_1, X_2, ..., X_k\}$, where $k \geq 2$, such that $(X_{i-1}, X_i) \in E$ or $(X_i, X_{i-1}) \in E$ for $2 \leq i \leq k$. We call the set of edges connecting the k nodes a chain between X_1 and X_k. A chain containing two nodes, such as $X - Y$, is called a link. A directed link, such as $X \rightarrow Y$, represents an edge. Given the edge $X \rightarrow Y$, we say the tail of the edge is at X and the head of the edge is Y. We also say the following:

- A chain $X \rightarrow Z \rightarrow Y$ is a head-to-tail meeting, the edges meet head-to-tail at Z, and Z is a head-to-tail node on the chain.
- A chain $X \leftarrow Z \rightarrow Y$ is a tail-to-tail meeting, the edges meet tail-to-tail at Z, and Z is a tail-to-tail node on the chain.
- A chain $X \rightarrow Z \leftarrow Y$ is a head-to-head meeting, the edges meet head-to-head at Z, and Z is a head-to-head node on the chain.

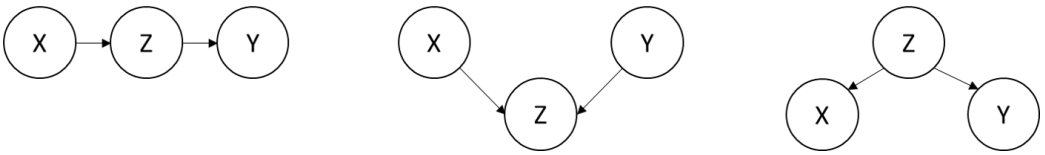


Figure 6: Left: head-to-tail meeting at Z. Center: head-to-head meeting at Z. Right: tail-to-tail meeting at Z.

Definition 2.2. Blocked Chain

Let $G = (V,E)$ be a DAG, $A \subseteq V$ a set of evidence variables, $X, Y \in V \setminus A$ two nodes, and ρ be a chain between X and Y. Then ρ is blocked by A if one of the following holds:

1. There is evidence on a node $Z \in A$ in the chain ρ, and the edges incident to Z in ρ meet head-to-tail at Z.

2. There is evidence on a node $Z \in A$ in the chain ρ, and the edges incident to Z in ρ meet tail-to-tail at Z.

3. There is a node Z in the chain ρ, such that Z and all of its descendants are not in A (there's no evidence on them), and the edges incident to Z in ρ meet head-to-head at Z.

To provide some intuition for the following definition and theorem, notice that for any joint PDF that complies to the conditional independence statements implied by the structures stated in figure 6, using the factorization implied by the network structure it is easy to see that if we're given no evidence whatsoever, then in the left and right structures, the variables X and Y are dependent (through Z), and in the center structure they are marginally independent. Given the value of Z as evidence, in the left and right structures, X and Y are conditionally independent given Z, while in the center structure, conditioning on Z introduces a dependence between them (explaining away). We can now define d-separation:

Definition 2.3. d-separation

1. Let $G = (V,E)$ be a DAG, $A \subseteq V$, and $X,Y \in V \setminus A$. We say X and Y are d-separated by A in G if every chain between X and Y is blocked by A.

2. Let $G = (V,E)$ be a DAG, and A, B and C be mutually disjoint subsets of V. We say A and B are d-separated by C in G if for every $X \in A$ and $Y \in B$,

X and Y are d-separated by C.

We worked through all these definitions to be able to state the powerful d-separation theorem, which without getting too formal or proving it, follows the following lines: Let $G = (V,E)$ be a DAG, A, B and C mutually disjoint subsets of V, and P a distribution function defined on the set of variables V, which is compatible with the conditional independence statements entailed in G. If A and B are d-separated given C in G, then $Ind_P(A,B|C)$. The d-separation theorem

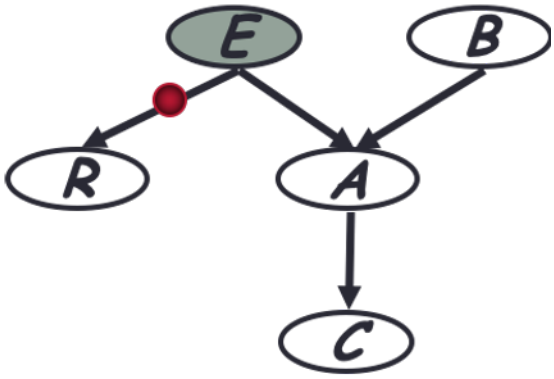


Figure 7: Given E, every chain from R to any other node is blocked

is very useful since it allows the discovery of conditional independence between groups of variables through a simple examination of the graph G - To find out if $Ind_P(A,B|C)$ you simply need to check the paths between A and B and see if they're blocked or active given C. For example, consider the Bayesian network in figure 7. Given the value of node E as an evidence, we can see that every chain from the node R to every other node in the graph is blocked since all these chains have a tail-to-tail meeting on E which is observed, thus R is d-separated from

all the other nodes given E, which thanks to the d-separation theorem means that $Ind(R, \{A, B, C\}|E)$.

2.2.3 Parameters Learning

So far we've described the Bayesian network model, its components, assumptions, usages, and advantages. We did not discuss however how they are constructed. When using Bayesian networks to model real-world problems, in some cases the network (either structure, parameters, or both) is known - that is, designed by an expert who utilized domain knowledge to construct the networks. Eliciting Bayesian networks from experts can be a laborious and difficult procedure in the case of large networks, so researchers developed methods that could learn the network's structure and parameters (conditional probabilities) from data. This problem is known as the BN learning problem, which can be stated informally as follows: Given training data and prior information (domain knowledge), find the network structure and parameters that best fit the data. Learning the structure is considered more difficult than learning its parameters. On top of that, in some cases you have to deal with missing data and hidden nodes, which complicate the task. In general, there are four BN learning cases, to which different learning methods are proposed (Murphy, 1998):

1. Known Structure, Full Data - Maximum Likelihood Estimation (MLE)

2. Known Structure, Partial/Hidden Data - Expectation Maximization (EM)

3. Unknown Structure, Full Data - Searching through model space - Score and

constraint based approaches

4. Unknown Structure, Partial/Hidden Data - EM + Search through the model space

We will present some of these methods in the following sections, starting with parameter learning (known structure).

To discuss the parameters learning process, we will begin by giving an example of some CPDs and their respective parameters, to clarify what is it that needs to be learned in this step.

The CPD of a node in a BN corresponds to its type (discrete/continuous etc.), its parent nodes types, and the connection between them. For example, in our work we primarily used a couple of CPDs:

1. For discrete nodes, we used the categorical distribution, which means that given the values of all of the node's discrete parents (we did not allow continuous parents for discrete nodes), each state of the variable has some probability which sums to 1 over all states, given a certain set of parent values. Parameters of this sort are represented using a Conditional Probability Table (CPT), which notes the probability for each state given the different combinations of values that the node's parents can take - $P(X_i = j|\pi_{X_i})$

2. For continuous nodes, we used a conditional linear Gaussian (see 2.3) CPD which allows discrete and continuous parents, such that the discrete parents affect the node as in a mixture of Gaussians connection, and the continuous

parents affect the node as in a multivariate gaussian connection. The parameters for such a node X_i, given its discrete parents Q and continuous parents Y are the mean μ_Q, covariance σ_Q and regression coefficients matrix W_Q so that $X_i|Q,Y \sim \mathcal{N}(\mu_Q + W_Q \times Y, \sigma_Q)$ (Murphy, 1998).

When learning parameters using full data, the goal is to find the parameters for each CPD in the BN, that maximize the log-likelihood of the training data. Say the dataset contains m cases (which are often assumed to be independent). Given training dataset $D = \{d_1, ..., d_m\}$, where $d_i = (d_{i1}, ..., d_{in})$, the set of parameters $\Theta = (\theta_1, ..., \theta_n)$, where θ_i is the vector of parameters for the CPD of variable X_i, the log-likelihood of the training dataset is a sum of terms: $\log L(\Theta|D) = \sum_{l=1}^{m} \sum_{i=1}^{n} \log P(d_{li}|\pi_i, \theta_i)$. Notice that using the BN conditional independence assumptions, the log-likelihood function decomposes nicely. We can now optimize the parameters Θ to maximize the log-likelihood, and return the optimal parameters as the result of the learning procedure. In the case of learning parameters with partial observability, that is either there are latent variables in the model, or randomly missing values for some of the variables in some of the examples, an accepted method of dealing with the missing data is using the Expectation-Maximization(EM) algorithm to find a locally optimal maximum likelihood estimate for the parameters. The EM algorithm is an iterative algorithm which alternates between an Expectation (E) step which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to

determine the distribution of the latent variables in the next E step. The EM algorithm converges to a local optima of the log-likelihood function, and as with many different optimization algorithms, it is sensitive to the initialization stage - the initial model parameters in our case. Some solutions to the initialization sensitivity include trying several starting parameters (either randomly or not), and choosing the best run as the final model parameters, or using clustering to get good starting parameters for the optimization. A common initialization method when dealing with mixture models is to initialize the parameters based on the results of a k-means clustering algorithm run on the data, such that the k-means centroids will serve as the initial means, and the initial covariance will be based on the distances from the centroids.

2.2.4 Structure Learning

The third case of Bayesian networks learning deals with unknown structure and full observability. The goal in this case is to learn a DAG that best explains the available data. This problem is called Structure Learning and it is considered to be more difficult than parameter learning. In fact, in the general case, structure learning is an NP-hard problem (Chickering, 1996), but it can be solved efficiently in specific cases, for instance, the Chow-Liu algorithm is an efficient structure learning algorithm for BNs with a tree structure (Chow and Liu, 1968). There are several approaches to structure learning: constrained-based methods, which revolve around discovering conditional independencies in the data and constructing a network that models them, and score-based methods which define a score

that evaluates the fit of a network to the data, and search for a network that has the best score given the training data. (Neapolitan et al., 2004) The first scoring function that comes to mind for this use case is the log-likelihood of the data given the model, but one has to be careful when using it in a structure learning procedure, since the log-likelihood score will only improve the more edges (complexity) are added to the model and so an optimization algorithm based on log-likelihood alone might return an overfitting model. A scoring function that tries to overcome the aformentioned problem, is the Bayesian information criteria (BIC) score: $BIC = \ln(n)k - 2ln(\hat{L})$ (Schwarz, 1978). The BIC score, is based on the log-likelihood of the data given the parameters, but takes into consideration a regularization term which penalizes complex models. This second factor keeps the optimization algorithms in check, and helps produce models with better generalization. As for the fourth case (structure learning with missing data), an interesting approach is suggested in (Friedman, 1998) to conduct local search steps inside of the M step of the EM algorithm. This method is known as structural EM, and presumably converges to a local maximum of the BIC score. In our work we made use of the Bayesian information criterion (BIC) score for different model selection tasks, such as deciding whether it is beneficial to add a latent variable to the model, and deciding on its optimal size, all while penalizing the extra added compelxity.

2.3 Conditional Gaussian Bayesian Networks

While the BN model accommodates both discrete and continuous variables, BNs are more commonly used with discrete variables. Inference and learning algorithms have been optimized with discrete variables in mind, and some popular software packages for BNs don't support continuous variables. Because of this, a popular method that researchers employ when analyzing a mixture of continuous and discrete data is to discretize the continuous data, which results in some loss of information. The Conditional Gaussian Bayesian Network (CGBN) is a type of BN in which continuous and discrete variables are mixed, under the restriction that continuous nodes have Gaussian distributions linearly dependent on their continuous parents with parameters conditioned on the values of their discrete parents. In this kind of BN, edges from continuous nodes to discrete nodes are disallowed, thus a discrete node cannot be modeled as statistically dependent upon continuous nodes (Lauritzen, 1992; Heckerman and Geiger, 1995; McGeachie M. J., 2014). The CGBN structure is a graph, in which the set of variables V is partitioned to $D \cup C$ where D is the set of discrete variables and C the set of continuous variables. In the CGBN model, only the following directed edges and their respective connection models are allowed: $continuous \rightarrow continuous$ for linear multivariate Gaussian, $discrete \rightarrow discrete$ for correlated discrete variables, and $discrete \rightarrow continuous$ for a mixture of Gaussians. Following these rules, the joint distribution of the continuous variables given the discrete is assumed to be multivariate Gaussian: $P(Y|I=i) = \mathcal{N}(\mu_i, \Sigma_i)$, where Y denotes the continuous variables, $I=i$ is a specific combination of the discrete variables, μ_i and Σ_i the

means vector and covariance matrix given the $i-th$ combination. The CGBN assumptions and restrictions enable the use of efficient inference algorithms that are based on exact local computation schemes designed specifically for linear conditional Gaussian models (on which the CGBN model is based)(Lauritzen and Jensen, 2001). The CGBN model remains viable in many fields despite its restrictions. For example, in genomics, researchers generally model continuous gene expression values as being dependent upon discrete genetic polymorphisms (McGeachie M. J., 2014), and in classification tasks, the popular Gaussian Naive Bayes classifier can be thought of as a simple CGBN. In our work we used the CGBayesNets package (McGeachie M. J., 2014) for learning and inference on CGBN models. For inference tasks, the CGBayesNets package uses different algorithms on the discrete and continuous portions of the network - the Cowell algorithm for inference in conditional Gaussian network nodes (Cowell et al., 2006), and a simple variable elimination algorithm for inference between discrete nodes in the network (Koller and Friedman, 2009). For structure learning, we used the hill-climbing algorithm implemented in CGBayesNets, which is a greedy, exhaustive, search algorithm that starts with an empty network and adds the best edge in each step, based on the improvement it introduces to the likelihood of the training data. The algorithm is exhaustive in that it considers all possible legal edges, between any two nodes. It may run with or without backtracking, which if enabled will also consider the removal of any existing edge, if that removal results in the greatest increase in likelihood. This algorithm is slower than the other options available in CGBayesNets, but sometimes provides better results.

3 Previous Work

The impact of latent variables is well studied in many domains. Of a particular interest is the impact of including latent variables in studies aimed at estimating the effects of social contagion. A well-known and somewhat controversial study by (Christakis and Fowler, 2007) claimed that obesity is contagious. Critics of that article pointed out that unaccounted latent homophily may provide an alternative explanation - obese people have more obese friends. This debate initiated an extensive study aimed at distinguishing between social contagion and homophily effects. A significant progress was made by Aral et al. (Aral et al., 2009), who were able to bound the effect of homophily on the estimation of social contagion. They provided an upper bound for social contagion effects, and showed that it drops dramatically when adjusting for about 50 covariates (confounders), but it doesn't diminish. This enables the inclusion of latent homophily as an additional source for the observed effects. Finally, Shalizi and Thomas (Shalizi and Thomas, 2011) used graphical models arguments to claim that social contagion and latent homophily can be indistinguishable, and it is difficult to differentiate the effect sizes, respectively.

Latent structure discovery and analysis is the subject of intensive research efforts in statistics and machine learning realms for many years. Factor analysis is one of the oldest structural models, originally conceived by Spearman in 1904. The fundamental assumption in Factor analysis is that dependency between observed variables can be explained solely by one or more latent variables. This

is known as the *local independence* assumption. Thus, the observed variables are conditionally independent given the latent factors. Figure 8 depicts a (latent) factor analysis model. Notice that there are no direct connections between the observed variables.

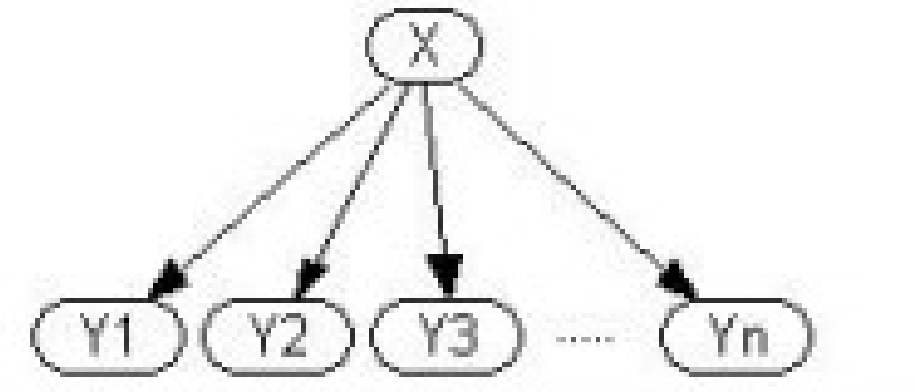


Figure 8: A factor analysis graphical model. Figure adapted from (Zhang, 2004)

Focusing attention on the special case where only two discrete variables exist, Gilula (Gilula, 1979) suggests a sufficient and necessary condition based on the Singular Value Decomposition (SVD) of the matrix of deviations from statistical independence, for the existence of a binary discrete latent variable affecting both of the observed variables. While this method deals with the discovery of latent variables, it has several limitations: Gilula's requires that both the variables be of size 3 or more (usually discrete variables in Bayesian networks are binary), it only detects latent variables of size 2, and it does not work with more than 2 discrete variables that interact with each other.

Hierarchical Latent Class models (Zhang, 2004), suggest some form of relaxation to the the local independence assumption, where connections between observed variables can be explained given a hierarchy of latent variables, c.f. Figure 9

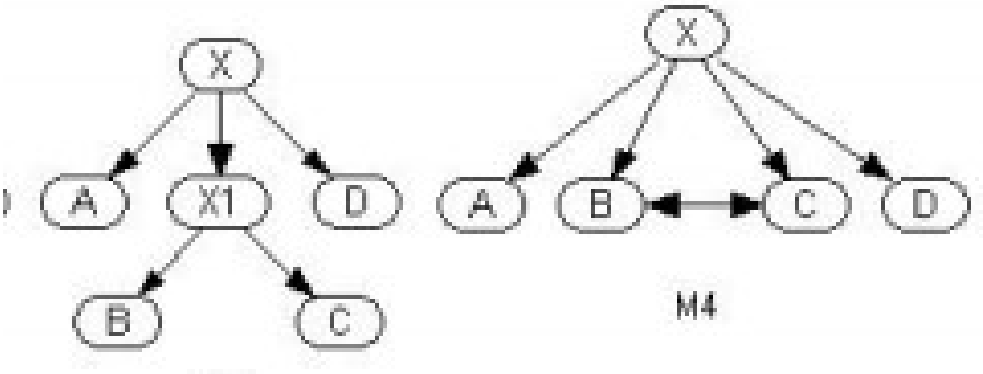


Figure 9: Modelling a direct connection with an added level of latent variable. Figure adapted from (Zhang, 2004)

Our method expands beyond the independence assumption, by preserving the direct connection between observed variables *given* the latent variable.

Furthermore, Factor Analysis and more broadly (Hierarchical) Latent class models require that the number of latent factors and the number of categories that they represent (the cardinality) be specified in advance. To this end, it is very common to iterate over various options and use measures (such as BIC score) to balance between accuracy and model complexity (i.e. number latent factors and the cardinality). Our method discovers the existence of the latent variables and their cardinality, rather than specifying them in advance.

Most related to our study is the line of work concerning Bayesian networks

structure learning with mixed observed and latent variables, and specifically Elidan et al. work on latent variable discovery (Elidan et al., 2001; Elidan and Friedman, 2005; Elidan et al., 2007).

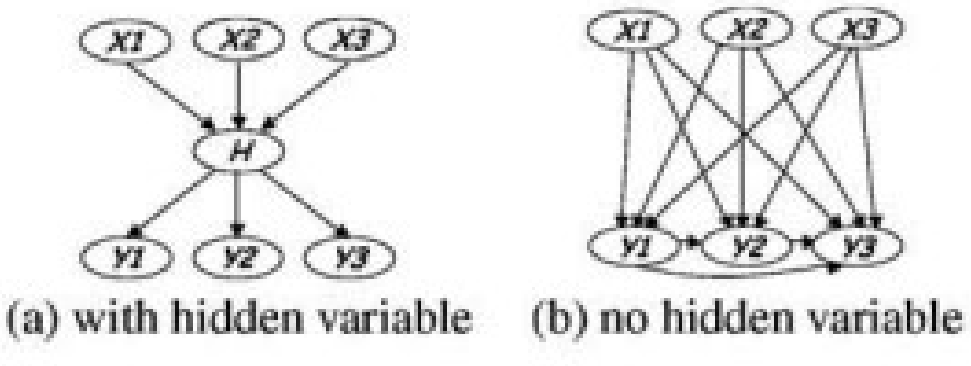


Figure 10: Latent variables reduce complexity. Figure adapted from (Elidan et al., 2001)

In (Elidan et al., 2001), Elidan et al. were motivated to discover latent variables that decrease the complexity of the models, which in turn accelerates learning and improves the fit to the data. Consider the network in Figure 10. The model with latent variable (10(a) on the left) uses only 17 parameters (assuming all of the nodes are discrete binary) while the one with only observed variables (10(b) on the right) uses 59 parameters. Elidan et al. developed an approach based on the graph structure of the models, to identify exactly such dependency patterns as in figure 10(b), and explain (simplify) them using a latent variable. Once such a "structural signature" of a latent variable has been detected, a latent variable candidate is inserted to the network. The cardinality of the latent variable candidate is learned (Elidan and Friedman, 2001), and the latent variable candidate stays in

the network if the BIC score of the network has improved. Their main structure learning vehicle is the Structural EM algorithm (Friedman, 1998) and its extensions. In (Elidan and Friedman, 2005), Elidan et al. introduce the Information-Bottleneck EM (IB-EM) algorithm which utilizes a tradeoff between minimizing the information in the hidden variables about the identity of the training instances and maximizing the information in the hidden variables about the observed data. They combined their method with SEM to deal with model selection with hidden variables allowing SEM to use a step of hidden variable addition and a step of alteration of the cardinality of hidden variables based on information cues in the data. In (Elidan et al., 2007), Elidan et al. describe the "Ideal Parent" method for speeding up the structure learning of continuous variable networks, while also suggesting possible hidden variables and incorporating the most promising ones. In this method, for every variable, Elidan et al. construct an ideal parent profile of a new hypothetical parent that would lead to the best possible prediction of the variable. A candidate parent of a variable will only be considered if it is similar to the ideal parent. If no candidate is similar to the ideal parent, that might be a cue to add a latent variable (and construct it to be similar to the ideal parent).

Our method cast the discovery of latent variables in mixed networks in a statistical hypothesis testing setting, seeking for sufficient evidence for the inclusion of latent variables, rather than aiming to reduce model complexity. The structure learning scheme that we suggest becomes much simpler than Structural EM. For parameter estimation (learning) though, we keep using EM.

4 Methodology

In this section we will describe our latent variable detection method, discuss its assumptions and limitations, and present a practical use for our method by presenting a structure learning algorithm for Conditional Gaussian Bayesian Networks (CGBNs) which is capable of discovering and modelling latent variables.

4.1 Discovering Latent Variables

Our latent variable detection method is based on the assumption that given observations on all known discrete causes of a continuous variable, its (conditional) distribution is uni-modal. We call this assumption the conditional uni-modality assumption. Under this assumption, we use Hartigan's dip test (see 2.1.1) to decide if multi-modalities exist in subsets of the data where we expect to see uni-modality due to our assumption. If we find unexplained multi-modalities, we report the existence of a potential latent variable. First, we assume that the input data contains a mixture of continuous and discrete variables. Then, given a BN that models the data, we make the assumption that fixing observations on all the discrete ancestors of a continuous variable in the network, the subset of its samples which matches the ancestors' state is uni-modal. In other words, conditioning on its discrete ancestors', the continuous variable has a uni-modal distribution. This should be true for all possible conditioning states. Thus, detecting a conditioning state which leads to a multi-modal distribution of the continuous variable, is an evidence to the existence of an accountable latent variable. For the

detection of multi-modal distributions in the data we chose Hartigan's dip test. An interesting analysis was made by Freeman et al. (Freeman and Dale, 2013), who compared dip test's bi-modality detection capabilities to two other bi-modality tests (bi-modality Coefficient and the difference in Akaike's information criterion between one-component and two-component distribution models) based on their sensitivity to different parameters of the generating distribution, such as distance between the modes, skew, kurtosis, etc. They found that the Dip test outperforms the other tests, having the highest sensitivity to bi-modality and lowest false positive rate. Before utilizing dip test to our needs, we first verified that the Dip test performs well in the multi-modal case (as opposed to bi-modal). We extended the aforementioned sensitivity tests and verified the validity of the results above in the detection of multi-modal distributions (See section 7.1). Next, we used the Dip test to detect latent variables in a BN containing a mixture of continuous and discrete variables, where continuous variables cannot be ancestors of a discrete variable. We then perform the following test: for every continuous variable in the BN, we condition over all the possible combinations of its discrete ancestors and use the Dip test on the resulting subsets of samples. If the p-value is lower than 0.05 for any of the combinations (one is enough), we declare it as multi-modal and due to the conditional uni-modality assumption, conclude that there is an additional discrete latent variable affecting the continuous variable in question. If all the p-values are higher than 0.05, then we conclude that the continuous variable's different modes are fully explained by its discrete ancestors. Note that our detection sensitivity is increased since it requires only one unexplained multi-modality

to add a latent variable. We demonstrate the ability of this procedure to detect hidden variables in two controlled settings, one using synthetic data and the other using real-life data. In both cases, we ”hide” one of the discrete variables, i.e. remove its measurements from the data, and check that our process is able to detect its presence.

In the synthetic setting, we demonstrate our detection capabilities on a very small network. The data generation process is outlined in figure 11. This data, and the DGP structure, were used to construct and train a fully observed BN. The marginal distribution of B is multi-modal by construction (c.f. Figure 12),

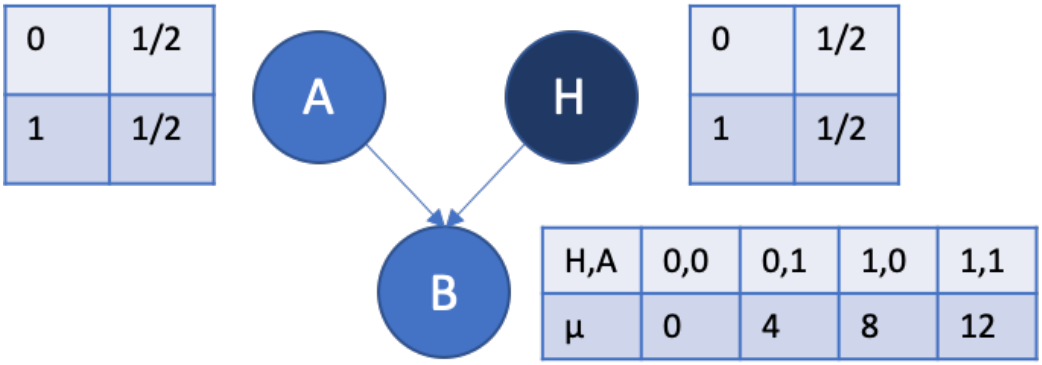


Figure 11: Data generating process for the synthetic setting. A, H are discrete, and B is Gaussian with conditional means (equal variance $\sigma = 1$)

however all conditioned distributions of the continuous variable B are uni-modal (since all the explanatory variables are observed). Now let's remove the (discrete) H variable from the BN. The distribution of B cannot be fully explained by the remaining A variable. For example, conditioning B on $A = 2$ (Figure 12, top right) results in a bi-modal distribution, which the dip test detected with p-value ≤ 0.00001. This bi-modality is attributed to the H variable that was hidden.

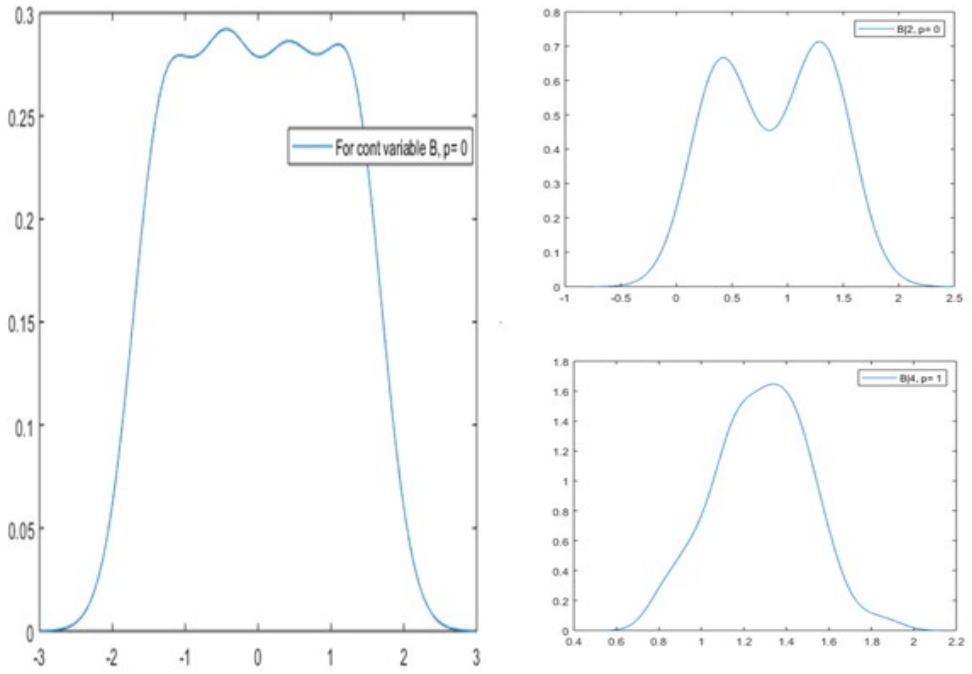


Figure 12: **On the left:** The distribution of B is clearly multi-modal (p-value ≤ 0.00001). **Bottom Right:** Given A and H we get one of the modes (p-value 1). **Top Right:** Hiding the H variable and conditioning only on A results in a bi-modal distribution (p-value ≤ 0.00001)

Next, we turn to detect latent variables in a controlled manner in real-life data, where data generation (and hence BN structure) is not fully understood, and the parametric form of the distribution of the variables is not known and can only be approximated. To this end, we used a dataset of train ticket prices of the Spanish high-speed train company "renfe" collected by a system which scrapes tickets pricing periodically. The dataset was collected by Pedro Muñoz and David

Cañones and is publicly available from Kaggle [1]. The full dataset contains information about 1 million train tickets prices, and has information such as the origin and destination, the dates of the trip, train type, ticket type (2nd class/1st class etc.) and ticket fare type (elder/student/regular etc.). We selected only the train tickets from Madrid to Sevilia and only 2 fare types (flexible and promo+). Thus, we were left with 3 categorical variables which affect the ticket price and fully explain it's modes – train type, ticket type, and ticket fare type. The price variable is ordinal in nature, so we used kernel density estimation to smooth the data in order to achieve stable mode detection results. We then fed our algorithm with all the data, and a BN with all 3 discrete variables as parents of the price variable. As expected, the algorithm reported that there are no conditioned multi-modalities in the data (Figure 13, left plate), namely no hidden variables were detected. Next, we hid the ticket fare type variable and fed the remaining observed data to the algorithm. The algorithm detected the bi-modality caused by different values of ticket fare type hidden in the data and reported the existence of a latent factor with high confidence (p-value ≤ 0.00001) (Figure 13, center plate).

Finally, we fed our algorithm with the full data of ticket prices from Madrid to Sevilia, which includes all fair types (rather than only two fair types), and used all the observed variables. This is not a controlled setting anymore, and any detected hidden variable is new. Indeed our algorithm detected an unexplained multi-modality, which occurs whenever the fare type variable takes the value *Promo*. A suggested explanation might be that the Promo ticket fare type is a promotion

[1] *https://www.kaggle.com/thegurusteam/spanish-high-speed-rail-system-ticket-pricing*

ticket whose price is subject to change due to supply and demand changes (Figure 13, right plate).

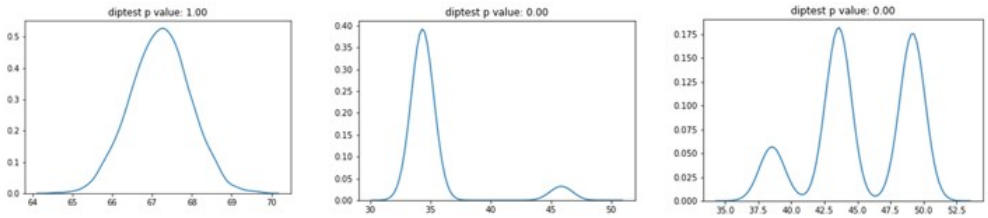


Figure 13: **Left:** Using only Flexible and Promo+ fare types, every combination was reported as uni-modal (p-value 1) **Center:** Hiding the fare type variable creates multi-modality (p-value ≤ 0.00001) **Right:** A real latent effect is reported when considering the full data (p-value ≤ 0.00001)

4.2 Structure Learning in the Presence of Latent Variables

Our starting point is to model the observed discrete/continuous data with a Conditional Gaussian Bayesian Networks (CGBN). We will now show that the CGBN model adheres to our conditional uni-modality assumption.

Lemma 4.1. Let $G = (V, E)$ be a CGBN, $c_n \in V$ a continuous variable, and $DA(c_n)$ the set of discrete ancestors of c_n. It holds that $P(c_n|DA(c_n))$ is Gaussian.

Proof. Let $C \subset V$ be the set of continuous variables, and $D \subset V$ the set of discrete variables.

From the properties of CGBN, $P(C|D)$ is Multivariate Gaussian (Lauritzen, 1992), hence $P(c_n|D)$ is Gaussian.

From the CGBN structure properties and following d-separation rules, it holds

that c_n is d-separated from $D/DA(c_n)$, given $DA(c_n)$. Thus, c_n and $D/DA(c_n)$ are conditionally independent given $DA(c_n)$.
Therefore $P(c_n|DA(c_n)) = P(c_n|D)$ is Gaussian □

Furthermore, we make the observation that not all of the discrete ancestors of a continuous node take part in setting its modality. This is important since as discussed in the previous section, we use all the different combinations of the discrete ancestors of a continuous node in order to detect if its affected by any latent variables. Reducing the number of ancestors that should be accounted for when testing for multi-modality reduces the run time of our algorithm dramatically.

Definition 4.1. Effective Discrete Ancestors Let $G = (V,E)$ be a DAG for a CGBN, thus G is a graph with the set of discrete nodes D, and the set of continuous nodes C, and there are no edges from C to D. We say that $d \in D$ is an Effective Discrete Ancestor of $c \in C$ if there exists a path π from d to c such that there are no discrete nodes in π other than d.

We will call the set of all the effective discrete ancestors of a continuous node c the Effective Discrete Ancestors Set (EDAS)

Consider figure 14 for clarification. From left to right: In the first case D_1 is in the EDAS because of the path $D_1 \rightarrow C_2 \rightarrow C_1$. In the second case D_1 is not in the EDAS since its effect is "blocked" by D_2 and D_3. In the third case, D_1 is not in the EDAS because of D_2. In the fourth case, D_1 is in the EDAS since the intermediate node is continuous.

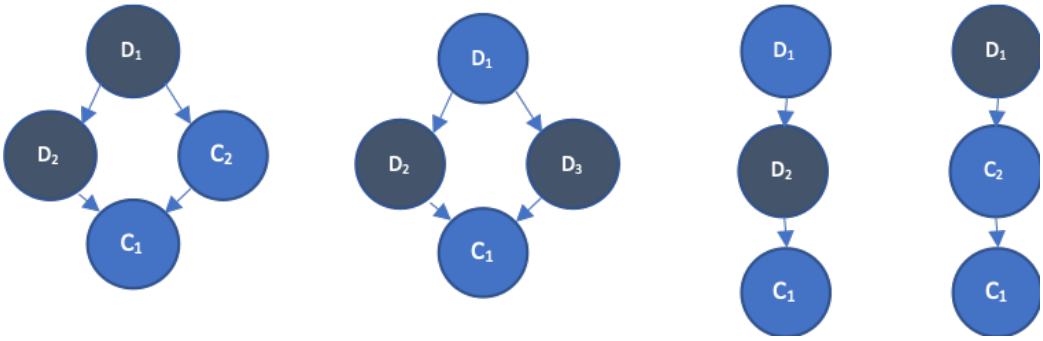


Figure 14: EDAS Examples. Shaded nodes are in the EDAS of continuous variable C_1.

Lemma 4.2. Let c be a continuous node in a CGBN, $DA(c)$ be the set of all the discrete ancestors of c and $EDAS(c)$ be the effective discrete ancestors set of c.

Then $P(c|EDAS(c)) = P(c|DA(c))$.

Proof. By construction $EDAS(c) \subseteq DA(c)$. Consider the set of nodes $\overline{EDAS(c)} = DA(c)/EDAS(c)$. For any node $d \in \overline{EDAS(c)}$, in any path π from d to c, there is an intermediate discrete variable $d^* \in EDAS(c)$ between d and c. From d-separation rules, for the chain structure that π represents, conditioning on d^* blocks the path π. Thus, conditioning on $EDAS(c)$, we get that $\overline{EDAS(c)}$ is d-separated from c, making them conditionally independent given $EDAS(c)$, and we get $P(c|EDAS(c)) = P(c|EDAS(c), \overline{EDAS(c)}) = P(c|DA(c))$ □

The construction of the EDAS of a continuous node can be done efficiently using a DFS algorithm from the node and up the ancestry paths, collecting the first occurrences of discrete nodes in each of the run's branches. It is important to note that conditioning on the immediate discrete parents of the continuous node in question is not enough to detect unexplained multi-modalities. Claiming that a

latent variable should be added based on such a test might not be accurate since indirect discrete ancestors can propagate their multi-modal effect through a chain of descendants. In such a case the multi-modality wouldn't be unexplained, but perfectly reasonable when accounting for the non-direct discrete ancestors and that is why we have to condition on the entire EDAS when testing for latent variables. Consider the experiment portrayed in figure 15 which demonstrates this effect: We created a network where a discrete ancestor created bi-modality in its immediate child node and sampled randomly from the network. We presented both a scatter of the joint distribution of X and Y, and a histogram of the marginal distribution of the indirect descendant Y which clearly show that the bi-modality that was generated in X by D was passed to Y. The phenomena can also be witnessed in the factorization implied by the CGBN: Marginalizing to inspect the distribution of Y gives us: $P(Y) = \int_X \sum_D P(D)P(X|D)P(Y|X) = \sum_D P(D) \int_X P(X|D)P(Y|X)$ which resembles the mixture of Gaussians distribution and hints at a modal nature.

Our structure learning algorithm method is as follows: given a data set of mixed variables (discrete and continuous) we start by applying the greedy hill-climbing structure learning algorithm for CGBNs implemented in the CGBayesNets package. Next, we employ the dip test based detection method to identify potential latent factors. In order to incorporate a detected hidden variable in the BN structure, a few decisions should be made: how to connect it to the relevant variables, estimate its cardinality and learn its parameters. We base our decisions regarding the cardinality and connection type on the Bayesian Information Criterion (BIC) (Schwarz, 1978), as follows: Start with the simplest model - a binary

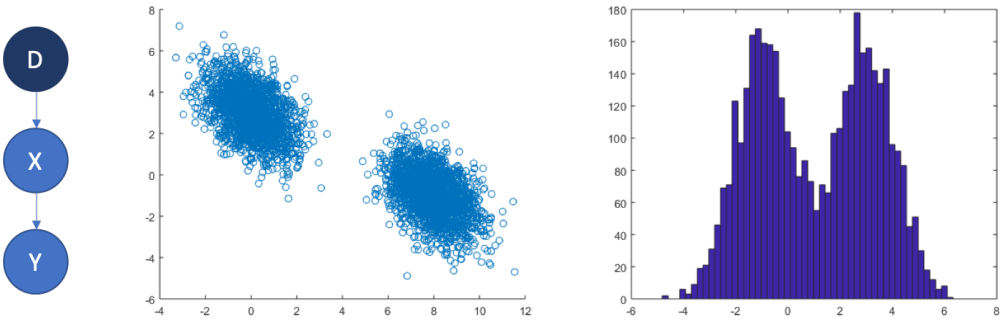


Figure 15: An ancestor can create modes in its non direct descendants. **Left:** The model **Middle:** A scatter of random sample of the nodes X and Y from the model on the left **Right:** The distribution of a random sample of node Y from the model on the left. Notice the multi-modality caused by the non-direct discrete ancestor D.

latent covariate connected only to the continuous variable in question; Apply the EM algorithm to estimate the model parameters and record the updated BIC score. We use a standard EM procedure broadly used in estimation of Mixture of Gaussians models. We have tried both random parameters initialization with multiple repetitions, and parameter initialization based on K-means which provided better results. We use a standard stopping criteria for EM based on the convergence of the log-likelihood of the training data. We continue the process by increasing the cardinality of the potential latent variable, thus raising the model's complexity as long as we see improvement in the BIC score. We then select the connection type and cardinality that provided the best BIC score, and introduce the new latent variable to the model based on them. We keep iterating on all the continuous variables in this fashion, detecting and introducing new latent variables to the network. Notice that the decision whether or not to add the latent variables is based solely on

dip test's detection of unexplained multi-modalities, but once that has been decided, the cardinality and connections are determined using BIC score. Because of that, it is possible that we will detect and add a latent variable to the model, even if its addition to the model decreases the overall BIC score of the network. BIC based decisions regarding the addition of latent variables are dependant on the paramaters learning stage, and more specifically, on the convergence of the EM algorithm, since the increase in likelihood attributed to the addition of the latent variable might not be enough to compensate for the added structural complexity in BIC's penalty term, for example in the cases where EM converges to non-ideal prarameters due to a bad initialization. Consider the simple example in figure 16: We generated a random sample from the network depicted on the right (a mixture of Gaussians with means at 0 and 2.5, and standard deviation of 1), hid the discrete variable from the data, and computed a dip test p-value of 0.048 which suggests multi-modality. Using our method, we would choose to add a latent variable since there's an unexplained multi-modality in the data. Checking the BIC scores of the different possible networks show the following: the BIC score of the original network with the sampled data is -7478, the BIC score of the single node network with parameters learned from the sampled data is -7588, and the BIC score of the network with a latent variable learned from the sampled data is -7601 (using a non-optimal random initialization of EM). Had we used BIC score to make a decision regarding the existence of the latent variable in this case, we would have chosen the single node network and declared that no latent variables exist. Our dip test based method is more stable, and even though we use EM and

BIC to estimate the parameters in our method, which may not converge correctly in some of the cases, we gain an evidence that the latent variable exists (under the conditional uni-modality assumption). Another interesting point to note lies

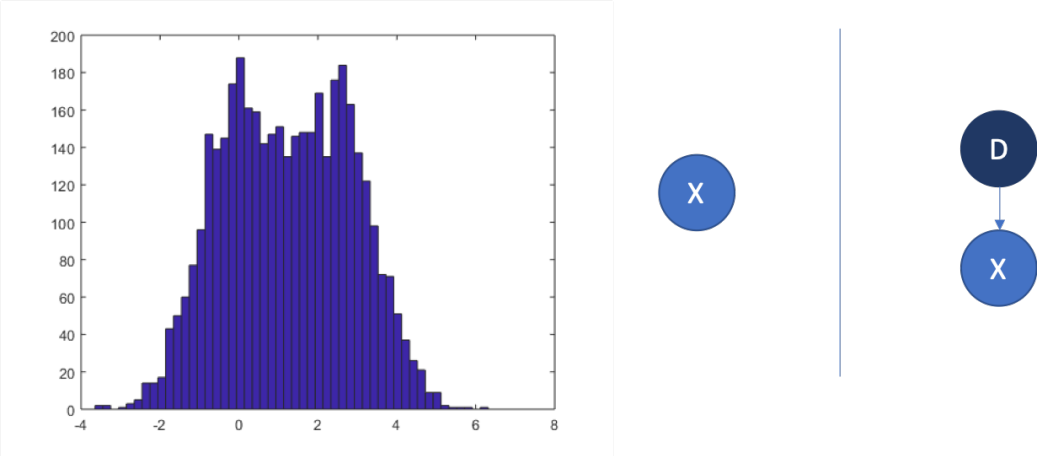


Figure 16: **Left:** A distribution of a random sample of the continuous variable X from the network on the right **Middle:** One possible model for the data, a uni-modal Gaussian **Right:** The original network.

in the local nature of the decisions that our algorithm takes. On the one hand, the local decisions are very advantageous: they simplify the problem and make the algorithm more efficient computationally, but on the other hand, they can result in wasteful addition of variables to the model. Consider figure 17 - running our algorithm on data generated from the network on the left (with the discrete parent hidden), results in the network on the right. The algorithm discovers an anomaly in each of the continuous variables, and adds the new latent variables D_1 and D_2, while a simpler, more accurate model which represents the knowledge better could be achieved by combining the latent variables to a single common parent as in the original network. This can be achieved by applying a post processing step

that checks the resulting network on a global scope. One method we suggest is to generate most probable explanations for the newly added latent variables, and to compute mutual information for all pairs of latent variables to try and find latent variables that could be combined. We leave this problem for future research.

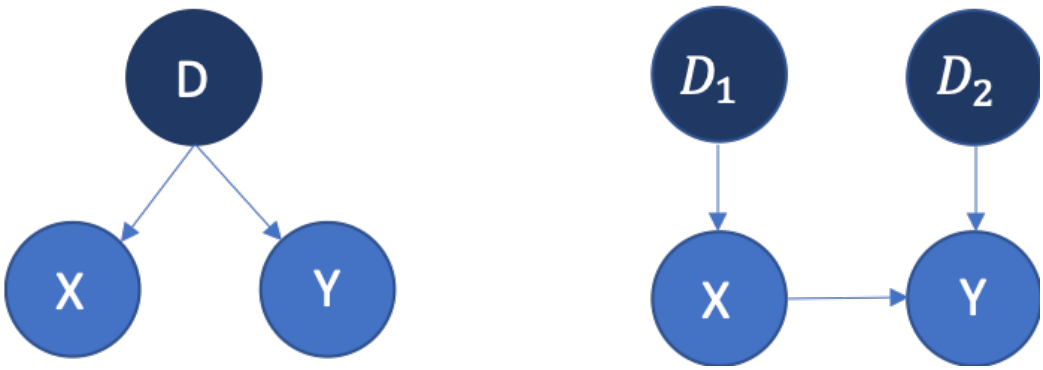


Figure 17: **Left:** The original model, a CGBN with a common discrete parent of two continuous variables **Right:** The result of our structure learning with latent variables algorithm on data generated from the model on the left.

The following is a detailed pseudo-code describing the flow of our algorithm.

Algorithm 1 LearnCGBNStructureWithLatentVariables(data)

```
InitialCGBN ← LearnCGBNStructureAndParams(data)
SuggestedNetwork ← InitialCGBN
C ← GetContinuousVariables(InitialCGBN)
for each c in C do
    Multimodal ← False
    EDAS ← FindEffectiveDiscreteAncestors(c, InitialCGBN)
    Combinations ← ListAllValueCombinations(EDAS)
    for each combination in Combinations do
        CoherentSamples ← FindCoherentSamples(D, c, combination)
        p ← DipTest(CoherentSamples)
        if p < 0.05 then
            Multimodal ← True
            Break
        end if
    end for
    if Multimodal then
        CurrentBIC ← CalculateBICScore(SuggestedNetwork)
        ConnectionTypesScores ← InitializeDictionary()
        ConnectionTypesSuggestedNetworks ← InitializeDictionary()
        for ConnectionType in {Covariate, Confounder, SideEffect} do
            UpdatedNetwork ← SuggestedNetwork
            LatentSize ← 2
            while LatentSize < MaxLatentSize do
                UpdateNetAndLearnParams(UpdatedNetwork, D, c, LatentSize, ConnectionType)
                if BICScoreImproved(UpdatedNetwork, D) then
                    Score ← BICScoreBN(UpdatedNetwork, D)
                    ConnectionTypeScore[ConnectionType] ← Score
                    ConnectionTypeSuggestedNetwork ← UpdatedNetwor
                    LatentSize++
                    Continue
                else Break
                end if
            end while
        end for
        BestScoreIndex ← Argmin(ConnectionTypesScores)
        SuggestedNetwork ← ConnectionTypesSuggestedNetworks[BestScoreIndex]
    end if
end for
Return SuggestedNetwork
```

5 Empirical Results

We performed a set of experiments to evaluate the success of our procedure in detecting and modelling hidden variables. The following experiments were conducted using real life data that contains a mixture of continuous and discrete observations. The experiments were aimed to explore the usefulness of our method for detecting and including latent variables to better explain the data, gain insights and discover knowledge, as well as improving prediction capabilities. In all of the following experiments, we compare the performance of our method to the performance of the fully observed network, and to that of a "baseline" latent network that is constructed by adding a single discrete latent parent to all of the observed network's nodes. The different networks are compared by: BIC score on the training data, likelihood of the test data, and when applicable - prediction related metrics. The "baseline" latent network is constructed in the following manner: A discrete latent variable is added to the fully observed network and connected as a parent of every other node in the network. Notice that no cycles are being created this way. We then use the training data to choose the best cardinality for the new variable. We try different cardinalities between 2 and 10, for each cardinality we learn the best parameters for the network using several runs of EM with random initialization, and measure the BIC score of the resulting network on the training data. The latent network with the best BIC score is chosen to be used in the respective experiment as a "baseline" latent network to compare to the latent network obtained using our method.

5.1 Health Insurance

The *Health Insurance* dataset (available from Kaggle [2]) includes 1300 records of health insurance charges, along with several discrete and ordinal covariates such as *age*, *gender*, *BMI* etc. We started by splitting the data to a training dataset and test dataset using a 80-20 split. Then, given the training data, our algorithm first learns the structure of an initial fully observed CGBN model that fits the data (c.f. Figure 19 the white nodes represent the observed network). Next, it applies dip test on the variables *age*, *bmi* and *charges* to detect potential latent variables affecting them given their discrete ancestors. For the detected hidden variables, the algorithm sets the optimal cardinalities, learns their parameters and decides how the detected hidden variables should be added to the model.

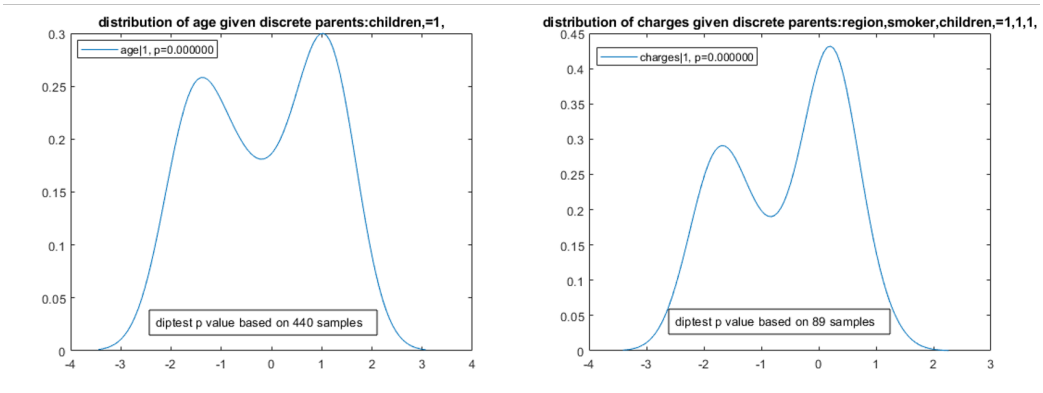


Figure 18: Left: Unexplained bi-modality for the age of people who don't have children: dip test p-value ≤ 0.00001. Right: Unexplained multi-modality in the distribution of the *charges* variable given its discrete ancestors: dip test p-value ≤ 0.00001

[2] https://www.kaggle.com/bmarco/health-insurance-data

Our algorithm detected a latent covariate of size 2 affecting the *age* variable, and added it to the model. The observed bi-modality happens when fixing *children=0* (Figure 18 on the left), thus indicating that there are two sub-populations of people who don't have children, where one group is significantly younger than the other. Thus, the hidden variable is indicative of the two sub-populations - a piece of information we could not deduce directly from the given features (the observed variables).

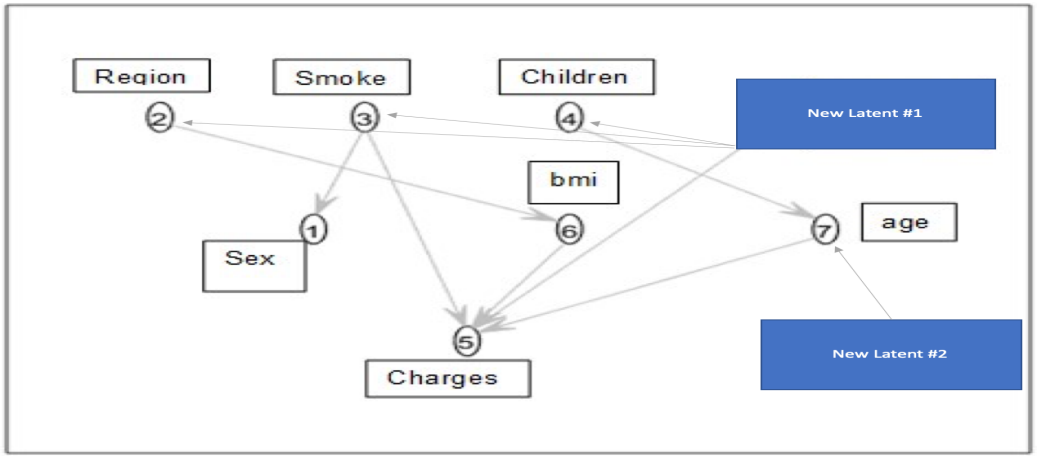


Figure 19: The Insurance Bayesian Network with two added hidden variables

In addition to that, our algorithm detected several unexplained multi-modalities that occur in the distribution of the *charges* variable, given its discrete ancestors (Figure 18 on the right). Our algorithm chose to connect this node to the network as a confounder of size 4 (Figure 19). Adding the detected hidden variables to the model improved the BIC score on the training data, and the likelihood of the test data both compared to the fully observed network and to the baseline latent network (Figure 20). The results are presented in table 1. These results show

Table 1: The performance of our method compared to the fully observed network and the baseline latent network in the insurance experiment

Network	Training BIC Score	Test Likelihood
Fully Observed	-7.7792e+03	-2.0703e+03
Baseline Latent	-7.1013e+03	-1.8969e+03
Our method	-7.0353e+03	-1.8503e+03

that the addition of latent variables enhance the fit to the data significantly even when taking into consideration the added complexity, as shown by the improved BIC score which penalizes complexity, and the increased likelihood of the held-out test data on both the latent networks. The latent network obtained using our method includes 2 new latent variables that are connected locally only to the nodes that require the higher complexity. Our latent network outperforms the baseline latent network which uses a single global latent variable. We attribute this to the fact that we use significantly less edges than the baseline method, resulting in a simpler and more focused model. In this case, this fact allowed our model to use a confounder of size 4, while the baseline latent model selected a latent parent of size 3 since increasing its cardinality any further would increase the complexity so much so that it'll outweigh the increase in likelihood (since it affects the parameters of all the other nodes).

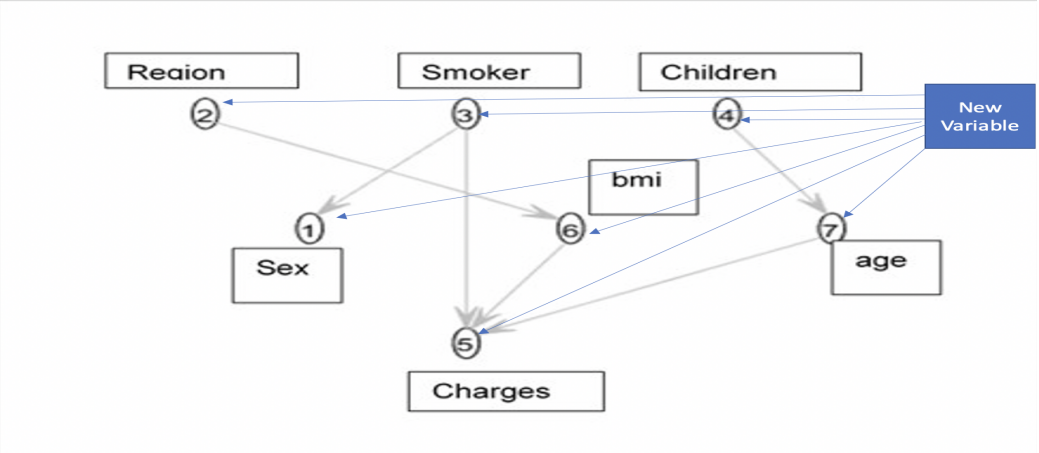


Figure 20: The baseline latent Bayesian network with a single discrete latent parent connected to every node

5.2 Activity Recognition in-the-Wild

The *ExtraSensory* dataset was collected and made publicly available by Vaizman et al. for research purposes related to behavioral context recognition in-the-wild from mobile sensors (Vaizman et al., 2017). It holds data from 60 users, each identified with a universally unique identifier. It contains thousands of examples for every user where each example includes multiple sensors measurements from the user's smartphone and smartwatch. Most examples also have self-reported labels which describe the user's activity at the time. The data comes from many different sensors such as: accelerometer, gyroscope, magnetometer, location services, audio, compass, phone state indicators and more. The Data was collected using the *ExtraSensory* mobile application which periodically records the sensory data and asks the user for an activity label.

We focused attention on the audio sensors data which is composed of 26

features that represent the means and standard deviations of 13 Mel-frequency cepstral coefficients (MFCC) that were recorded over a 30 seconds window. We wanted to test the effect of our structure learning with latent variables algorithm on the predictive power of a classifier which uses the sensor data to predict the different labels. We selected to model the problem with a Gaussian Naïve Bayes (NB) classifier due to its simplicity, d-separability (Pearl, 1988) and the fact that it contains edges only from discrete variables to continuous variables, which fits our algorithm's prerequisite. We selected labels that were potentially predictable with the NB model: Trained an NB classifier on pairs of activity label and an MFCC feature and tested the balanced accuracy ($BA = \frac{1}{2}(Sensitivity + Specificity) = \frac{1}{2}(\frac{TP}{P} + \frac{TN}{N})$) performance on held-out data. Two activity labels that had a sufficient number of samples and presented a high BA score were selected: *sleeping* and *lying down*.

To evaluate our algorithm's performance, we used the 5-folds cross validation setting defined at (Vaizman et al., 2017), which partitions the data according to user IDs. This ensures that there is no leakage of information from the train set to the test set. For each fold, and each of the two labels, we trained an NB model on the training data with a set of selected audio features. We used the NB model as the initial fully observed structure model for our algorithm, which seeks to enhance it by detecting and adding hidden variables. We used the dip test to discover potential hidden variables, added them to the model using two edges – from the label node to the new hidden variable, and from the hidden variable node to the corresponding MFCC feature (c.f. Figure 21). We chose this type of connection

to account for the fact that the discovered multi-modalities occurred in only one of the states of the label.

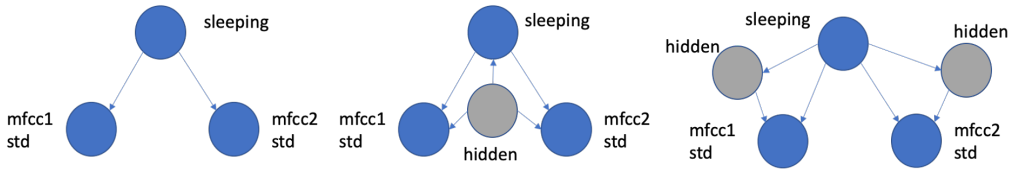


Figure 21: **Left:** An initial Gaussian NB model for the *sleeping* label **Center:** The baseline latent model **Right:** The NB model enhanced with latent variables

To evaluate the predictive performance of our model, we followed (Vaizman et al., 2017) and averaged the BA scores of the enhanced models over the defined 5 folds and compared them to the BA scores of the corresponding initial NB models, and baseline latent networks. To evaluate the goodness of fit of our model, we took note of the BIC score on the training data, and the likelihood of the held-out test data in each of the folds. Table 2 compares the performance of the different models averaged over the 5 folds and the 2 labels.

Table 2: The performance of our method compared to the fully observed network and the baseline latent network in the activity experiment. Results averaged over the 5 folds and the 2 labels

Network	Training BIC Score	Test Likelihood	BA
Fully Observed	-3.67307e+05	-8.95478e+04	0.720
Baseline Latent	-2.02964e+05	-4.86670e+04	0.753
Our method	-2.26232e+05	-5.99941e+04	0.755

In this experiment, both the latent models outperform their fully observed counterparts, but while our method slightly outperforms the baseline latent model in balanced accuracy, the baseline model presents better goodness of fit to the

data. We attribute this to the fact that the fully observed networks were relatively small, and contained a majority of variables with unexplained multi-modalities, causing our method to add a large amount of hidden variables that could have been merged.

5.3 COVID19 Test Results Prediction

The COVID19 Clinical Data Repository (Health and Health, 2020) by Carbon Health and Braid Health is an effort to compile a repository of the clinical characteristics of patients who have taken a COVID-19 test. While relatively small with 1610 rows (at the time of writing), the dataset is of very high quality, clean, informative, and compliant with HIPAA Privacy Rule's De-Identification Standard. Each row in the dataset contains information about a patient's COVID19 test result, including further detail regarding the testing procedure such as swab type and the kind of test taken, his reported symptoms such as cough, fever and sob, some measurements taken from him such as temperature, pulse and lung oxygen saturation levels, his comorbidities, and several epidemiological factors such as whether he has been in contact with infected people, his age etc. This dataset can be used to train classifiers that will predict whether a person is infected or not based on a questionnaire and tests that are cheaper and more available than the current COVID19 tests. In this experiment, we present how the hidden variables detected using our method can be used to improve the performance of widely used classifiers. First, we stratified split the COVID19 dataset to train (80%) and test (20%) sets with respect to the label (the test result). We used the mean of each column to impute missing values in the train and test sets, and normalized the continuous columns. Once the preparations were over, we ran a structure learning algorithm on the train set to get an initial CGBN to feed to our algorithm (c.f. Figure 23), the yellow nodes are the fully observed MB network). We used a structure learning algorithm originally used for phenotype prediction problems

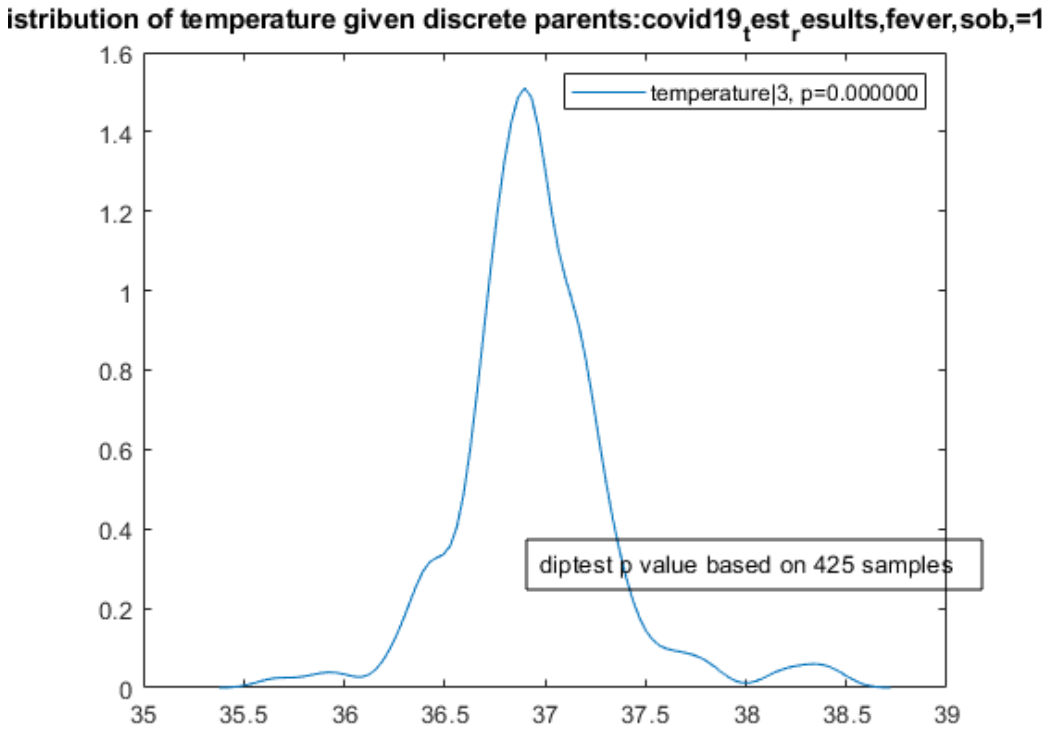


Figure 22: An unexplained multi-modality exists in the distribution of the *temperature* variable given its discrete ancestors

(Chang and McGeachie, 2011) which is better suited for classification problems. The algorithm produces a Markov Blanket (MB) of the target variable which is comprised of its parents, children, and other parents of those children. Based on the conditional independence assumptions of the BN, only these nodes are required to predict the value of the target variable. Thus, for prediction tasks, the full network is not required, and we can get by using a much smaller network. We chose to use this type of structure learning in order to focus our algorithm's attention on variables which have predictive power over the target label - a sort

of dimensionality reduction that will help the optimization steps in our algorithm provide more accurate results. We fed the MB network to our algorithm, which detected a latent variable that had an effect on the *temperature* variable with high significance (dip test p-value ≤ 0.0001) (c.f. Figure 22). and incorporated it into the network as a parent of the temperature variable (c.f. Figure 23).

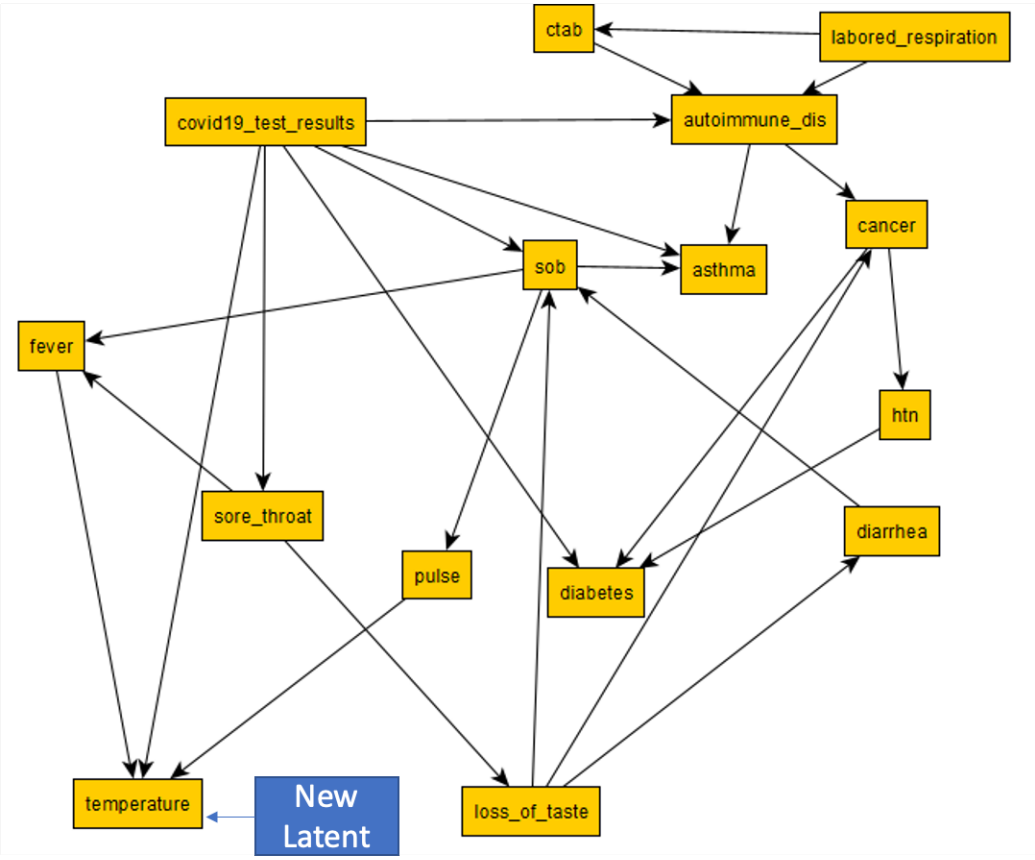


Figure 23: The latent-enhanced network our algorithm learned for the COVID dataset. Yellow nodes are observed.

The BIC score of the network with respect to training data, and the likelihood of the held-out test data improved thanks to the addition of the latent variable, and

despite of the complexity added due to its inclusion. On top of that, our network fit to the data better than the baseline latent model. See a summary of the results in table 3

Table 3: The performance of our method compared to the fully observed network and the baseline latent network in the COVID19 experiment.

Network	Training BIC Score	Test Likelihood
Fully Observed	-1.0356e+04	-2.5208e+03
Baseline Latent	-1.0234e+04	-2.5173e+03
Our method	-1.0164e+04	-2.5005e+03

We then used the latent enhanced network to generate the posterior probability of the hidden variable given the observed variables of each example in both training and test sets. The augmented train and test sets were used to train and test several classifiers, and their performance (accuracy and AUC of the ROC curve) were matched against the same models trained and tested on the original data sets (i.e. without the hidden variable's posterior probabilities) and on the original data sets enhanced with the posterior probability of the latent from the baseline network. The results are depicted in table 4. In all of the cases, the AUC of the models trained on the enhanced dataset (using our method) was better than their counterparts. Some models improved more significantly than others. We attribute this to capacity factors, and the ability of the models to discover the latent information on their own. The improvement of the AUC is more significant due to the highly unbalanced nature of the data.

Table 4: The performance of several models is improved with the addition of the latent information

	Base Dataset		Dataset + Hidden		Dataset + Hidden	
			Our Method		Baseline	
Model	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
kNN	0.928571	0.547103	0.931677	0.561507	0.934783	0.554801
Linear SVM	0.683230	0.696854	0.742236	0.756291	0.686335	0.676325
RBF SVM	0.773292	0.680795	0.776398	0.745530	0.782609	0.685265
Gaussian Process	0.937888	0.671523	0.937888	0.695861	0.937888	0.649338
Random Forest	0.934783	0.734106	0.940994	0.772185	0.940994	0.703311
Neural Net	0.934783	0.668543	0.937888	0.772351	0.934783	0.709768
AdaBoost	0.928571	0.605132	0.922360	0.610017	0.931677	0.552566

6 Summary

In this dissertation we present a novel method for discovering latent variables that interact with observable variables that are directly related, where the observable parent and the latent variables are both discrete, and the observed child is a continuous variable. To this end, we take a statistical hypothesis testing approach, and base our method on Hartigans' dip test, seeking for sufficient evidence for the inclusion of latent variable, rather then a structure-based approach which aimed at reducing model complexity (Elidan et al., 2001).

We demonstrated the utility of our method with a set of experiments, in both controlled and real-life settings. In these experiments we demonstrated how the inclusion of latent variables can better explain the data, reveal additional insights, enhance understanding, as well as enhance the fit to the data and accuracy. We compared our latent enhanced networks to baseline latent networks built from the fully observed networks by adding a single latent discrete parent to all of the

nodes. Our method presented either better predictive power (higher BA in the activity experiment and higher AUC in the COVID experiment) or better fit to the data (in the insurance and COVID experiments) than the baseline network. We think that if we find a way to effectively merge new latent variables with existing ones, we will present better fit to the data in all of the experiments above, but that will be left for future research.

The discovery of latent factors that are added to an already existing direct connection, rather than replacing it (i.e. latent class models), is of particular interest in many domains, as it has the potential capacity to provide an additional insight and explanation to an observed phenomenon. Such is the case in latent homophily vs. social contagion, and also in drug efficacy vs. side effects, placebo, and DDI.

The use of dip test as the main tool for detecting latent factors enables to relax the assumption about the parametric form of the observed continuous child - All we need to assume is that unconditionally it has a uni-modal distribution (unlike many latent models that assume Gaussian distribution). Practically speaking, this allows us to apply kernel density estimation, as a preprocess, to handle ordinal variables as well.

However, we are currently not handling latent factors discovery in parent-child relations between discrete categorical variables. This is a rather difficult problem, which is commonly handled by employing conditional independence assumptions (c.f. (Gilula, 1979; 1983) for necessary and sufficient conditions for such a decomposition). Extending it to handle additional forms of observed and latent variables interaction is left for future research.

7 Appendix

7.1 Dip Test Sensitivity to Multi-Modality

We expanded on the set of experiments performed by Freeman et al (Freeman and Dale, 2013), to test the sensitivity of dip test to multi-modality (as opposed to bi-modality). We reproduced the basic results by showing that without sufficient distance between the generating Gaussian's means (lower than 3-4 standard deviations), dip test fails to identify the multi-modality (p-value > 0.05). This result is consistent with the bi-modality sensitivity tests, and has been reproduced with varying sample sizes (50,500,1000,2000) and different number of modes (3,4,6,8).

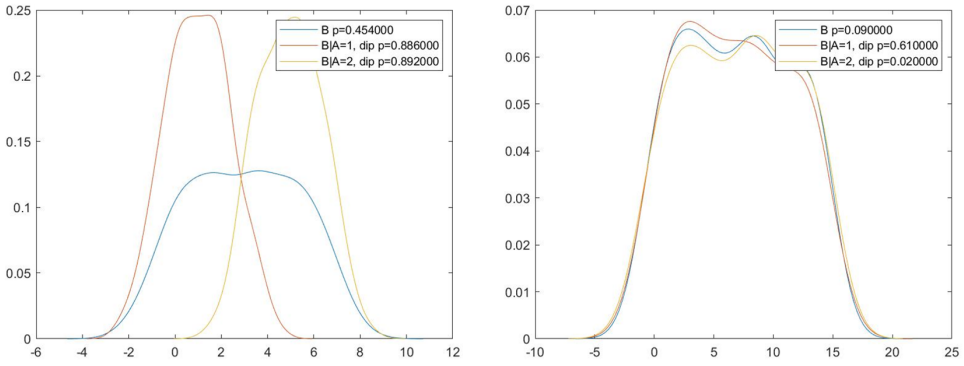


Figure 24: On the left: 4 Gaussians with means: 0,2,4,6, standard deviation is 1.

With sufficient distance between the generating gaussian's means (higher than 3-4 standard deviations), dip test identifies the multi-modality with high significance (p-value < 0.05). This result is consistent with the bi-modality sensitivity

tests, and has been reproduced with varying sample sizes (50,500,1000,2000) and different number of modes (3,4,6,8).

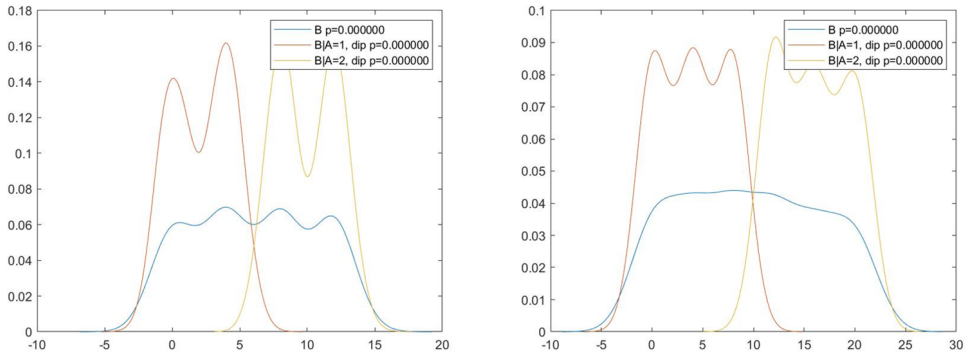


Figure 25: On the left: 4 Gaussians with means: 0,4,8,12. On the right: 6 Gaussians with means: 0,4,8,12,16,20. Standard deviation 1.

An interesting result in the multi-modal case is that introducing a larger gap between two consecutive means in a series of equally spaced and close Gaussians (in terms of standard deviations) creates a bi-modal effect which triggers dip test to report multi-modality with smaller minimal distances than the bi-modal case. (in the figure, small distance between means is 1 standard deviation, the larger gap is 2 standard deviations). This result has been reproduced with varying sample sizes (50,500,1000,2000) and different number of modes (3,4,6,8,10).

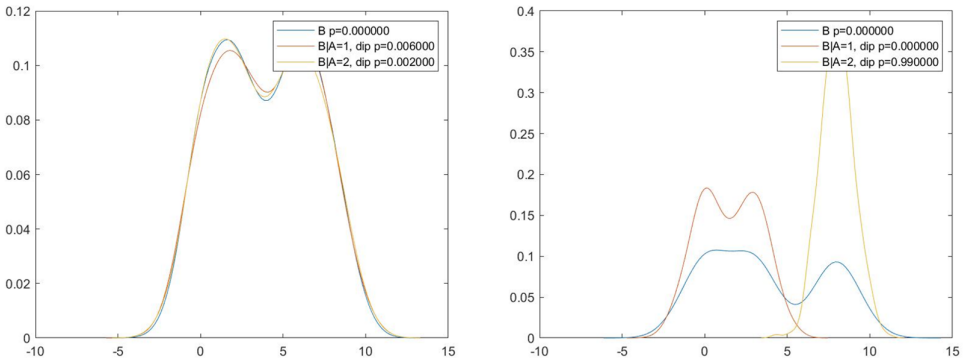


Figure 26: On the left: 8 different Gaussians with means: 0, 1, 2, 3, 5, 6, 7, 8 and standard deviation 1. Notice the larger gap between 3 and 5.

References

S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, 2009.

H.-H. Chang and M. McGeachie. Phenotype prediction by integrative network analysis of snp and gene expression microarrays. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6849–6852. IEEE, 2011.

D. M. Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

N. A. Christakis and J. H. Fowler. The spread of obesity in a large social network over 32 years. *New England journal of medicine*, 357:370–379, 2007.

R. Cowell, S. Lauritzen, and J. Mortera. Maies: A tool for dna mixture analysis. In *Proceedings of the Twenty-Second Conference Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 90–97. AUAI Press, 2006.

G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 144–151, 2001.

G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In *Advances in Neural Information Processing Systems*, pages 479–485, 2001.

G. Elidan, I. Nachman, and N. Friedman. " ideal parent" structure learning for continuous variable bayesian networks. *Journal of Machine Learning Research*, 8(8), 2007.

J. B. Freeman and R. Dale. Assessing bimodality to detect the presence of a dual cognitive process. *Behavior research methods*, 45(1):83–97, 2013.

N. Friedman. The bayesian structural em algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

B. J. Fulton, E. A. Petigura, A. W. Howard, H. Isaacson, G. W. Marcy, P. A. Cargile, L. Hebb, L. M. Weiss, J. A. Johnson, T. D. Morton, et al. The california-kepler survey. iii. a gap in the radius distribution of small planets. *The Astronomical Journal*, 154(3):109, 2017.

Z. Gilula. Singular value decomposition of probability matrices: Probabilistic aspects of latent dichotomous variables. *Biometrika*, 66(2):339–344, 1979.

Z. Gilula. Latent conditional independence in two-way contingency tables: A diagnostic approach. *British Journal of Mathematical and Statistical Psychology*, 36:114–122, 1983.

J. A. Hartigan and P. M. Hartigan. The dip test of unimodality. *The Annals of Statistics*, 13:70–84, 1985.

C. Health and B. Health. Coronavirus disease 2019 (covid-19) clinical data repository. Accessed from `https://covidclinicaldata.org/.`, 2020.

D. Heckerman and D. Geiger. Learning bayesian networks: a unification for discrete and gaussian domains. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 274–284, 1995.

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

S. L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.

S. L. Lauritzen and F. Jensen. Stable local computation with conditional gaussian distributions. *Statistics and Computing*, 11(2):191–203, 2001.

S. Maurus and C. Plant. Skinny-dip: Clustering in a sea of noise. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1055–1064, 2016.

W. S. T. McGeachie M. J., Chang H. H. Cgbayesnets: Conditional gaussian bayesian network learning and inference with mixed discrete and continuous data. *PLoS Computational Biology*, 10, 2014.

K. Murphy. A brief introduction to graphical models and bayesian networks, 1998. *Available electronically at http://www. cs. ubc. ca/ murphyk/Bayes/bnintro. html*, 1998.

R. E. Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 1988.

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6: 461–464, 1978.

C. R. Shalizi and A. C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods and Research*, 40:211–239, 2011.

Y. Vaizman, K. Ellis, and G. Lanckriet. Recognizing detailed human context in-the-wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16: 62–74, 2017.

N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

תקציר

משתנים חבויים מהווים אתגר במשימות שונות של סטטיסטיקה וכריית נתונים, כמו תכנון ניסויים סטטיסטיים, בעיות מידול ובעיות הסקה. רוב המחקר שנעשה בנושא משתנים חבויים עוסק בהכנסתם לאוסף השיקולים של המערכת, ובשערוך ההשפעה שלהם על שאר המשתנים בסביבתם, מתוך הנחת מוצא שהם קיימים במערכת. בעבודה זו אנו מתמקדים בזיהוי של משתנים חבויים שכאלה על בסיס מבחני השערות סטטיסטיים תוך כדי שימוש ברשתות בייסיאניות.
אנחנו מציגים שיטה חדשנית לזיהוי של משתנים חבויים בדידים, אשר משפיעים על משתנים גלויים רציפים, במערכת שמערבת משתנים בדידים ורציפים. אנחנו משלבים את יכולות הזיהוי שלנו באלגוריתם ללמידת מבנה של רשתות בייסיאניות, ומציגים את השימושיות של היכולות הללו בסדרה של ניסוים על מידע סינטטי ומידע אמיתי .

עבודה זו בוצעה בהדרכתו של ד״ר שי פיין מהמכון למדעי הנתונים, המרכז הבינתחומי, הרצליה.

בית ספר אפי ארזי
למדעי המחשב
הבינתחומי
הרצליה

המרכז הבינתחומי בהרצליה
בית-ספר אפי ארזי למדעי המחשב
התכנית לתואר שני (M.Sc.) - מסלול מחקרי

זיהוי משתנים חבויים בדידים ברשתות מעורבות בעזרת מבחנים סטטיסטיים

מאת
אביב פלד

עבודת תזה המוגשת כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.
במסלול המחקרי בבית ספר אפי ארזי למדעי המחשב, המרכז הבינתחומי הרצליה

ספטמבר 2020